Out-of-distribution Generalization in Deep Learning : Classification and Spatiotemporal Forecasting

Matthieu Kirchmeyer

▶ **To cite this version:**

HAL Id: tel-04139066

https://theses.hal.science/tel-04139066

Submitted on 23 Jun 2023

Out-of-Distribution Generalization in Deep Learning: Classification and Spatiotemporal Forecasting

Généralisation hors-distribution en Apprentissage Profond: Classification et Prévision Spatiotemporelle

Présentée par
Matthieu Kirchmeyer

Dirigée par
Patrick Gallinari et Alain Rakotomamonjy

Pour obtenir le grade de
DOCTEUR de SORBONNE UNIVERSITÉ

Abstract

Deep learning has emerged as a powerful approach for modelling static data like images and more recently for modelling dynamical systems like those underlying times series, videos or physical phenomena. Yet, neural networks were observed to not generalize well outside the training distribution, in other words out-of-distribution. This lack of generalization limits the deployment of deep learning in autonomous systems or online production pipelines, which are faced with constantly evolving data. In this thesis, we design new strategies for out-of-distribution generalization. These strategies handle the specific challenges posed by two main application tasks, classification of static data and spatiotemporal dynamics forecasting. The first two parts of this thesis consider the classification problem. We first investigate how we can efficiently leverage some observed training data from a target domain for adaptation. We then explore how to generalize to unobserved domains without access to such data. The last part of this thesis handles various generalization problems specific to spatiotemporal forecasting.

Résumé

L'apprentissage profond a émergé comme une approche puissante pour la modélisation de données statiques comme les images et, plus récemment, pour la modélisation de systèmes dynamiques comme ceux sous-jacents aux séries temporelles, aux vidéos ou aux phénomènes physiques. Cependant, les réseaux neuronaux ne généralisent pas bien en dehors de la distribution d'apprentissage, en d'autres termes, hors-distribution. Ceci limite le déploiement de l'apprentissage profond dans les systèmes autonomes ou les systèmes de production en ligne, qui sont confrontés à des données en constante évolution. Dans cette thèse, nous concevons de nouvelles stratégies d'apprentissage pour la généralisation hors-distribution. Celles-ci tiennent compte des défis spécifiques posés par deux tâches d'application principales, la classification de données statiques et la prévision de dynamiques spatiotemporelles. Les deux premières parties de cette thèse étudient la classification. Nous présentons d'abord comment utiliser des données d'entraînement en quantité limitée d'un domaine cible pour l'adaptation. Nous explorons ensuite comment généraliser à des domaines non observés sans accès à de telles données. La dernière partie de cette thèse présente diverses tâches de généralisation, spécifiques à la prévision spatiotemporelle.

Remerciements

Ma thèse touche à sa fin et je tiens à remercier les nombreuses personnes qui m'ont soutenu et accompagné pendant ces trois années et m'ont permis de mener cette thèse jusqu'au bout.

Tout d'abord, je remercie les membres du jury qui ont accepté d'évaluer ma thèse : Céline Hudelot, Gilles Gasso, Gérard Biau et Jörn-Henrik Jacobsen.

Ma thèse n'aurait pu se concrétiser ni se construire sans l'aide précieuse de mes directeurs de thèse, Patrick et Alain, qui m'ont guidé tout au long de cette recherche et que je remercie. Tout particulièrement, un grand merci, Patrick, de m'avoir accueilli dans ton équipe au MLIA et de m'avoir aidé à construire mes directions de recherche pendant ma thèse ! Je remercie Criteo de m'avoir donné l'opportunité de réaliser cette thèse et de l'avoir financée.

Je remercie aussi l'ensemble de mes co-auteurs qui ont contribué aux travaux présentés dans cette thèse, pour toutes les discussions passionnées et les moments partagés ensemble ; j'ai tellement appris en travaillant à vos côtés ! Merci Yuan, Jean-Yves, Alexandre, Emmanuel, Jérémie, Matthieu, Thibaud et Nicolas.

Merci aussi à tous les doctorants et collègues que j'ai côtoyés ces dernières années au MLIA et chez Criteo ainsi qu'à tous mes amis rencontrés à Ginette, aux Mines et ailleurs, pour les moments de convivialité partagés avec vous durant ces trois années, qui m'ont motivé dans mon travail. Notamment, au MLIA, merci à l'équipe spatiotemporelle et aux membres de mon bureau ; à Criteo, merci à mon équipe ainsi qu'à l'équipe de recherche.

Enfin, je tiens à remercier ma famille et mes proches qui m'ont accompagné et soutenu durant ces trois années. Notamment, merci à mon épouse, Camille et à mes deux petites filles, Charlotte et Marguerite, pour tous les moments de bonheur qu'elles m'ont apporté et leur soutien sans faille. Merci à mes parents, mes frères, Nicolas et Alexandre, et toute la famille pour votre soutien actif. Et finalement, merci à Pierre, Marie-Charlotte, Jean-Pierre et ma belle-famille pour tous les moments de convivialité.

Contents

Acronyms

ADV Adversarial

CE Cross Entropy

CEM Classification Expectation Maximization

CNN Convolutional Neural Network

CTR Click-Through-Rate

CR Conversion Rate

COVS Covariate Shift

DL Deep Learning

DA Domain Adaptation

DG Domain Generalization

ERM Expected Risk Minimization

ENS Ensembles

IVP Initial Value Problem

IBVP Initial Boundary Value Problem

FFT Fast Fourier Transform

GAN Generative Adversarial Network

GP Gaussian Process

GETARS Generalized Target Shift

GNN Graph Neural Network

GBML Gradient-Based Meta Learning

IID Independent and Identically Distributed

ID Identically Distributed

IM Information Maximization

INR Implicit Neural Representation

LP Linear Probing

ML Machine Learning

MCAR Missing Completely at Random

MAPE Mean Absolute Percentage Error

MSE   Mean-Squared Error

MTL   Multi-Task Learning

MMD   Maximum Mean Discrepancy

NN   Neural Network

NTK   Neural Tangent Kernel

ODE   Ordinary Differential Equation

OOD   Out-of-distribution

OT   Optimal Transport

PDE   Partial Differential Equation

SSL   Semi-Supervised Learning

TARS   Target Shift

WA   Weight Averaging

UDA   Unsupervised Domain Adaptation

Part I

MOTIVATION

Chapter 1

Introduction

> Nature does not shuffle the data, so
> we shouldn't either
>
> ———————————————————
> Léon Bottou at ICLR 2019

1.1 Context

Deep Learning (DL) has recently emerged as a powerful tool to learn complex patterns from data. DL models are currently trained using large annotated datasets and have achieved impressive results in domains such as natural language processing (Wu et al. 2016), computer vision (Szegedy et al. 2017), recommender systems (Covington et al. 2016) or games (Silver et al. 2016). They are now widely used for applications such as language translation and text classification, object detection and image classification or content recommendation. Their success has mainly relied on learning better representations of the data at hand, *e.g.* via pretrained models (Ramesh et al. 2022; Brown et al. 2020), trained on a large dataset and used as a starting point for finetuning on new tasks.

There has also been a growing interest in applying DL to model temporal phenomena *e.g.* time series (Franceschi et al. 2019) or videos (Denton et al. 2018). More recently, DL was applied to the physical sciences *e.g.* climate or biology (Willard et al. 2020; Thuerey et al. 2021) in applications such as weather forecasting (Pathak et al. 2022), aerodynamics (Pfaff et al. 2021) or material design (Kirkpatrick et al. 2021). One key problem in these applications is to model complex spatiotemporal dynamical systems. So far, Partial Differential Equation (PDE)s are the main modeling approach and are used at different scales, from molecules to geophysics modeling. PDEs are solved explicitly via numerical solvers (Hairer et al. 2000). DL is being used as a surrogate to these numerical methods or in combination (Yin et al. 2021b; Avila Belbute-Peres et al. 2020; Kochkov et al. 2021), given some simulated or real-world observations for training these models. It was found to

solve several common limitations of purely physical models: namely, it improves simulation speed and cost; it handles high dimensionality or partial observations and it does not require to know the exact underlying physical laws. These advantages can accelerate scientific discoveries *e.g.* by reducing the cost and time of simulation or enabling model discovery.

1.2 Subject

Despite these successes, there are still major challenges for DL. One challenge, is the ability to generalize to distribution shifts; in other words, to generalize on target data that lies outside the train distribution. This is the Out-of-distribution (OOD) generalization problem, handled in this thesis.

OOD generalization is a key requirement for DL models in real-world applications:

- In e-commerce, Click-Through-Rate (CTR) prediction (Kirchmeyer et al. 2021) or federated learning (Fallah et al. 2020) models should handle new users.

- In health, Covid-19 detectors from chest scans should generalize beyond training dataset's biases to be fit for clinical use (DeGrave et al. 2021; Roberts et al. 2021).

- In autonomous driving, Neural Network (NN)-based perception systems should generalize to new countries or weather conditions for world-wide deployment (Michaelis et al. 2019).

- In text, language models should be personalized to different users for better user privacy (Li et al. 2021a).

- In the physical sciences, simulators should handle different contexts *e.g.* new parameters, initial or boundary conditions, spatiotemporal observations.

The standard inference principle to train NNs is Expected Risk Minimization (ERM). ERM shuffles data assuming it is Independent and Identically Distributed (IID). Yet, ERM does not guarantee OOD extrapolation and may be prone to spurious correlations. For instance, image recognition models may fail to correctly classify cows on the beach, when mostly cows on grassy fields were seen at training time (Beery et al. 2018a). They have indeed used the background as a predictive feature. In sciences, physics-based models are not prone to this problem as they come with generalization guarantees as long as the underlying equations are valid.

The limitations of ERM call for new alternative Machine Learning (ML) approaches which are more robust to OOD data.

1.3   Contributions

In this thesis, we propose such alternatives which improve the generalization of NNs in various problems. We introduce different tasks and problems which make different assumptions on the train and test domains. A domain corresponds to a given data acquisition context with a specific probability distribution over inputs and labels. In most work, the train domain consists of multiple different yet related domains, from which we can learn domain invariances and specificities. This direction is motivated by the observation that "nature does not shuffle the data, so we shouldn't either" (Léon Bottou).

We consider representative application problems, namely classification of static data *e.g.* images and forecasting of spatiotemporal physical dynamics. The former is well-studied in the ML community with numerous baselines and standard NN architectures. The latter is a more recent and fast-growing field which still requires defining new models to handle the space and time continuity of such systems.

1.3.1   Domain Adaptation (DA) (Part ii)

DA models adapt a NN to the target domain by leveraging at training time few target data and plenty of data from multiple training domains. The access to target samples helps better model the target labeling function, especially when it differs from the training domain's one. Namely, we consider Unsupervised Domain Adaptation (UDA) for classification, which assumes that some unlabelled target samples are available at training time. The standard UDA approach learns domain-invariant representations; yet it fails under missing data and complex domain shifts. We propose new approaches to handle these two challenging settings, as introduced below.

1. **Missing Data (Chapter 3)**. Motivated by cold start problems in recommendation, we consider that some data components are systematically missing on the target domain without available supervision for imputing them. Standard UDA methods cannot handle this missing information. We propose a new UDA model that attempts at reconstructing this missing information by performing imputation in a domain-invariant latent space with a generative model. It also leverages indirect classification supervision from the complete source domain. We ground our approach theoretically and provide experimental validation on a real-world dataset.

   M. Kirchmeyer, P. Gallinari, A. Rakotomamonjy, and A. Mantrach (2021). "Unsupervised domain adaptation with non-stochastic missing data". In:

2. **Complex Domain Shifts (Chapter 4)**. We then consider the most challenging domain shift, Generalized Target Shift (GeTarS), with both conditional and label shift. This corresponds to a setting where the target labeling function differs from the training one, yet shares the same classes. Existing domain-invariant approaches have practical limitations in this setting, including strong assumptions that may not hold in practice. Instead of constraining representation invariance, we propose to align pretrained representations and circumvent existing drawbacks. Our approach OSTAR learns an Optimal Transport (OT) map, implemented as a NN, which maps fixed source representations onto target ones. OSTAR is flexible and scalable, it preserves the problem's structure and has strong theoretical guarantees under mild assumptions.

1.3.2   Domain Generalization (DG) in Classification (Part iii)

We then investigate DG models for classification that do not leverage data from the target domain at training or test time to generalize. This is an extremely challenging setting as the problem is ill-posed when the target labeling function differs from the training domain's one.

1. **Weight Averaging (WA) for OOD Generalization (Chapter 5)**. We propose a new model, DiWA, to account for domain shift by ensembling multiple predictors. DiWA achieves ensembling by averaging weights obtained from several independent training runs. The main motivation of DiWA is to increase the functional diversity across averaged models. On the competitive DomainBed classification DG benchmark (Gulrajani et al. 2021a), WA of multiple NNs recently consistently outperformed representative OOD baselines. Yet, existing strategies only average models collected along a single run, which are less diverse than models across runs. We motivate the need for diversity by a new bias-variance decomposition of the expected error, exploiting similarities between WA and standard Ensembles (ENS). This decomposition provides some formal conditions for the success of WA OOD.

Experimentally, DiWA sets a new state-of-the-art on DomainBed without any inference overhead.

A. Rame*, M. Kirchmeyer*, T. Rahier, A. Rakotomamonjy, P. Gallinari, and M. Cord (2022). "Diverse Weight Averaging for Out-of-Distribution Generalization". In: *Neural Information Processing Systems (NeurIPS)*

1.3.3   Generalization in Spatiotemporal Forecasting (Part iv)

We then investigate generalization problems in the setting of spatiotemporal forecasting. These include a variety of settings described in Section 2.2.3 among which we consider two.

1. **Context-Informed Dynamics Adaptation (Chapter 6)**. First, we consider the setting where the dynamical system changes at test time *e.g.* when the parameters vary. Our approach, CoDA, performs test-time parameter adaptation by conditioning them on some observed data from a new domain. Conditioning is performed via a linear hypernetwork, which defines implicitly a weight subspace to which adaptation is restricted. CoDA is fast, sample- and data-efficient. We consider the application of CoDA to the problem of adapting neural dynamics models to new physical parameters. CoDA leverages for training different domains, each corresponding to a different physical system. All domains share the same general dynamics but correspond to different physical parameters. CoDA outperforms related meta-learning strategies on representative Ordinary Differential Equation (ODE)/PDE datasets and is grounded theoretically.

   M. Kirchmeyer*, Y. Yin*, J. Dona, N. Baskiotis, A. Rakotomamonjy, and P. Gallinari (17–23 Jul 2022). "Generalizing to New Physical Systems via Context-Informed Dynamics Model". In: *Proceedings of the 39th International Conference on Machine Learning (ICML)*. vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 11283–11301

2. **Spatiotemporal Generalization in PDE Forecasting (Chapter 7)**. Finally, we explore another direction to generalize, by designing NN architectures better suited to the problem at hand. For PDE forecasting, a key problem is to generalize to new spatiotemporal observations. Yet, existing neural PDE surrogates are currently too reliant on the training spatiotemporal discretization and extrapolate poorly beyond the train conditions. To this end, we propose to better account for the continuous nature of physical processes by combining the spatial flexibility of Implicit Neural Representation (INR)s

with the temporal flexibility of neural ODEs in our model DINo. DINo combines the following advantages: it extrapolates at arbitrary spatiotemporal locations; it can learn from sparse irregular grids or manifolds; at test time, it generalizes to new grids or resolutions. DINo outperforms alternative neural PDE forecasters in a variety of challenging generalization scenarios on representative PDE systems.

Y. Yin*, M. Kirchmeyer*, J-Y Franceschi*, A. Rakotomamonjy, and P. Gallinari (2023). "Continuous PDE Dynamics Forecasting with Implicit Neural Representations". In: *International Conference on Learning Representations (ICLR)*

1.3.4 Outline of this Thesis

This document is organized as follows.

- Part i includes the above introduction (Chapter 1) and details the literature and the necessary background for presenting our contributions (Chapter 2).

- Our contributions are presented in Parts ii to iv:

    - Part ii details the proposed methods for UDA: Chapter 3 handles the missing data setting and Chapter 4 the complex domain shift setting.

    - Part iii Chapter 5 details our DG method for classification, based on WA.

    - Part iv details our methods for generalization in spatiotemporal forecasting. Chapter 6 introduces our context-informed test-time adaptation method for handling changing dynamics. Chapter 7 presents our new continuous neural PDE forecaster for spatiotemporal generalization.

- Finally, Part v with Chapter 8, concludes this document with an overview of the contributions and a discussion of their perspectives.

- An appendix in Part vi contains supplementary material for Chapters 3 to 7 in Appendices A to E.

Chapter 2

Background and Related Work

We present in this chapter the background and related work of this thesis. We consider successively our two application domains:

- Section 2.1 covers Chapters 3 to 5, which handle classification.

- Section 2.2 covers Chapters 6 and 7, which handle dynamics forecasting.

These two sections correspond to separate problem settings and models but share similarities in the techniques used for generalization, originally introduced for classification.

2.1 Classification

We present the classification problem in Section 2.1.1. Section 2.1.2 introduces the standard learning strategies for this problem. They were adapted to handle domain shift as reviewed in Section 2.1.3. The contributions in this thesis build upon these learning strategies.

2.1.1 Notations and Problem Setting

Setting We consider multi-class classification. $\mathcal{X}$ is the input space (*e.g.* static images) and $\mathcal{Y} = \{1, ..., K\}$ the label space with K classes. X, Y denote random variables with values in $\mathcal{X}, \mathcal{Y}$. We consider a Neural Network (NN) $h(\cdot, \theta) : \mathcal{X} \to \mathcal{Y}$ with weights θ. h will be independently trained on some given data and tested on some other data. It consists of:

- an encoder $g : \mathcal{X} \to \mathcal{Z}$ from $\mathcal{X}$ to $\mathcal{Z}$; $\mathcal{Z} \subset \mathbb{R}^d$ is the latent space with dimension d and Z is a random variable in the latent space. There are various choices of architectures usually pretrained on ImageNet (Krizhevsky et al. 2012) *e.g.* ResNet-50 (He et al. 2016a) or CLIP (Radford et al. 2021).

- a classifier $f : \mathcal{Z} \to \mathcal{Y}$, usually shallow and retrained from scratch.

Data   In practice, "data" is defined over a domain D, defined by a joint, posterior or marginal distribution of (X, Y), denoted p_D. $\mathcal{D}_D$ corresponds to some training data from domain D. It consists of pairs $(\mathbf{x}_D, y_D)$ of input, label sampled from the domain's joint distribution *i.e.* $(\mathbf{x}_D, y_D) \sim p_D$ or $(\mathbf{x}_D, y_D) \in (\mathcal{X}_D \times \mathcal{Y}_D)$ where $\mathcal{X}_D \triangleq \{\mathbf{x} \in \mathcal{X}/p_D(\mathbf{x}) > 0\}, \mathcal{Y}_D \triangleq \{y \in \mathcal{Y}/p_D(y) > 0\}$. We assume that there is no noise in the data: then, the label on domain D is defined by $f_D : \mathcal{X} \to \mathcal{Y}$, a labeling function via $\forall (\mathbf{x}_D, y_D) \sim p_D, f_D(\mathbf{x}_D) = y_D$.

Risk   Given $\ell : \mathcal{Y}^2 \to \mathbb{R}_+$, we define the risk a.k.a. error of h on D as:

$$\mathcal{E}_D(h, \ell) \triangleq \mathbb{E}_{\mathbf{x} \sim p_D}[\ell(h(\mathbf{x}, \theta), f_D(\mathbf{x}))] \tag{2.1}$$

A standard choice is to take ℓ as the 0-1 loss. This corresponds to the accuracy of the model, usually used to evaluate different classifiers. Then,

$$\mathcal{E}_D(h) \triangleq \mathcal{E}_D(h, \ell_{0-1}) = \mathbb{E}_{\mathbf{x} \sim p_D}[\mathbb{I}(h(\mathbf{x}) \neq f_D(\mathbf{x}))] = \Pr_{\mathbf{x} \sim p_D}(h(\mathbf{x}) \neq f_D(\mathbf{x})) \tag{2.2}$$

Another possible choice is to use the Cross Entropy (CE) loss ℓ_{ce} defined as

$$\ell_{ce}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{c=1}^{K} y_c \log(\hat{y}_c)$$

with $\hat{\boldsymbol{y}} = h(\mathbf{x})$ the predictions of h (after applying a softmax activation) and $\boldsymbol{y}$ a one-hot vector of the label $f_D(\mathbf{x})$, output by f_D.

Loss   We can also define the empirical version of the risk, which we call here loss. Given an architecture h with parameters θ, $\mathcal{L}(\theta, \mathcal{D}_D)$ corresponds to the empirical loss over a given training dataset $\mathcal{D}_D$ in domain D with the standard ℓ_{ce} loss:

$$\mathcal{L}(\theta, \mathcal{D}_D) \triangleq \mathbb{E}_{\mathbf{x} \in \mathcal{D}_D}[\ell_{ce}(h_\theta(\mathbf{x}, \theta), f_D(\mathbf{x}))] \tag{2.3}$$

Learning Setting   In the following, D refers to either a source domain S with distribution p_S or a target domain T with distribution p_T. We will use indifferently the terminologies: domain, task and environment.

There are various learning settings summarized in Table 2.1, Supervised Learning, Semi-Supervised Learning (SSL), Multi-Task Learning (MTL), Unsupervised Domain Adaptation (UDA), Meta-Learning and Domain Generalization (DG). We illustrate major training settings in Figure 2.1.

Table 2.1. – Learning setups. L^i and U^i denote the labeled and unlabeled distributions from the ith domain, D^i. S is the training domain and T the test domain. $d_{tr} \neq 1$ represents the number of available domains for multi-domain problems. Adapted from Gulrajani et al. 2021a.

Setup	Domains		Train inputs		Test inputs	Domain Shift
	S	T	S	T	T	$p_S = p_T$?
Supervised learning	D^1		L^1		$\emptyset$	$p_S = p_T$
SSL	D^1		L^1, U^1		$\emptyset$	$p_S = p_T$
MTL	$\cup_{i=1}^{d_{tr}} D^i$		$\cup_{i=1}^{d_{tr}} L^i$		$\emptyset$	$p_S = p_T$
UDA	$\cup_{i=1}^{d_{tr}} D^i$	$D^{d_{tr}+1}$	$\cup_{i=1}^{d_{tr}} L^i$	$U^{d_{tr}+1}$	$\emptyset$	$p_S \neq p_T$
Meta-Learning	$\cup_{i=1}^{d_{tr}} D^i$	$D^{d_{tr}+1}$	$\cup_{i=1}^{d_{tr}} L^i$	$\emptyset$	$L^{d_{tr}+1}$	$p_S \neq p_T$
DG	$\cup_{i=1}^{d_{tr}} D^i$	$D^{d_{tr}+1}$	$\cup_{i=1}^{d_{tr}} L^i$	$\emptyset$	$\emptyset$	$p_S \neq p_T$

- All aim at learning θ such that the target generalization error $\mathcal{E}_T(\theta)$ in Eq. (2.2) is minimized, *i.e.* $h(\cdot, \theta)$ approximates f_T on $\mathcal{X}_T$.

- They differ by the definition of domains S and T namely:

    - domain shift present *i.e.* $p_S = p_T$ or not *i.e.* $p_S \neq p_T$ (Table 2.1 column 7),

    - single-domain ($d_{tr} = 1$) or multi-domain ($d_{tr} > 1$) problem,

    - unlabelled or labelled T samples available at train, test time or not (columns 5 and 6 in Table 2.1).

2.1.2 Standard Classification Strategies

In the standard scenario, represented by the first three rows in Table 2.1 (supervised learning, SSL, MTL), we observe plenty of train data with distribution p_S and perform inference on new unlabelled samples from the same distribution $p_T = p_S$. In other words, there is no domain shift between train and test.

2.1.2.1 Supervised Learning and Expected Risk Minimization (ERM)

ERM   The most common setting is supervised learning, which leverages some labeled data from a single domain S ($d_{tr} = 1$) for inference on unlabeled sam-
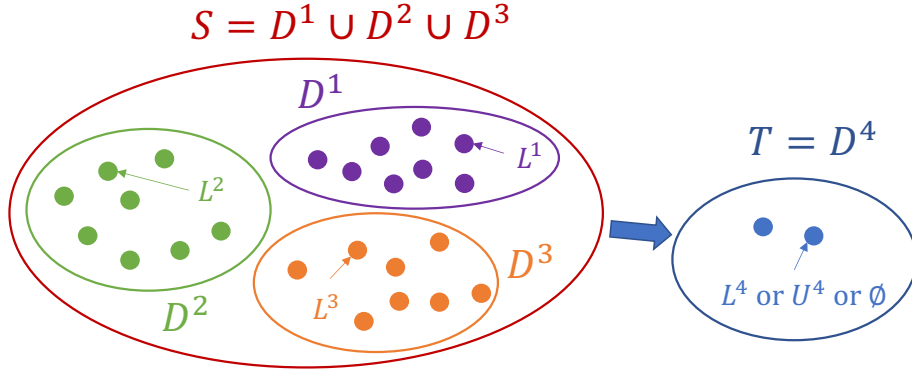
Figure 2.1. – Illustration of various training settings. L^i and U^i denote the labeled and unlabeled distributions from the ith domain, D^i. S is the training domain and T the test domain such that $p_S \neq p_T$. In all settings, we observe samples from S (L^1, L^2, L^3) at training time. In UDA, we observe unlabelled target samples (U^4) at training time. In Meta-Learning, we observe some labelled target samples (L^4) at test-time. In DG, we do not observe target samples ($\emptyset$) at training nor test time.

ples from domain T. The inference setting is ERM that considers that data is Independent and Identically Distributed (IID), where $\mathcal{L}$ is defined in Eq. (2.3):

$$\min_{\theta} \mathcal{L}(\theta, \mathcal{D}_S) \tag{2.4}$$

Regularized ERM   Several other strategies extend Eq. (2.4) by including a regularization term $\mathcal{R}(\cdot, \mathcal{D}_D)$, defined over a well-chosen dataset $\mathcal{D}_D$ over domain D, weighted by a hyperparameter λ:

$$\min_{\theta} \mathcal{L}(\theta, \mathcal{D}_S) + \lambda \mathcal{R}(\theta, \mathcal{D}_D) \tag{2.5}$$

We detail in the following some standard choices of this regularization term and of dataset $\mathcal{D}_D$ over which this term is defined.

2.1.2.2   Self-supervised Learning (SSL)

SSL considers additionally that some unlabeled samples from T are available for training. The standard SSL strategies consider the regularized form in Eq. (2.5), where $\mathcal{R}$ is defined over these unlabeled samples as:

- A classification CE loss with pseudo-labels as labels, as in Discriminant Classification Expectation Maximization (CEM) (Amini et al. 2005).

- A conditional-entropy loss (Grandvalet et al. 2005).

A main assumption for SSL is the cluster assumption (Chapelle et al. 2010), that states that decision boundaries should not cross high-density regions.

2.1.2.3   Multi-task Learning (MTL)

In the following, all strategies relax the IID assumption by considering that the train domain consists of multiple domains *i.e.* $d_{tr} > 1$.

The standard multi-domain strategy, MTL, aims at learning from multiple domains to better generalize to these same domains. In this setting, $S = T = \cup_{i=1}^{d_{tr}} D^i$. MTL does not address adaptation to new domains as inference is limited to the train domains, although some extensions to this challenging problem were proposed (Wang et al. 2021a; Requeima et al. 2019).

A standard approach in MTL is hard-parameter sharing, where earlier layers of the network are shared across domains (Caruana 1997). Several extensions were proposed to learn more efficiently from a set of related domains (Rebuffi et al. 2017; Rebuffi et al. 2018). Argyriou et al. 2006 makes sparsity assumptions w.r.t. model parameters, assuming that only a small set of features is shared across models.

2.1.3   Classification With Domain Shift

In the most challenging setting, we are faced with distribution shifts, a.k.a. Out-of-distribution (OOD) data. This occurs when $p_S(X, Y) \neq p_T(X, Y)$. The main strategies for this setting are listed in the last three rows in Table 2.1 (UDA, Meta-Learning, DG).

We decompose distribution shifts into different settings, investigated in this thesis:

- **Covariate Shift (CovS)** a.k.a. diversity shift (Shimodaira 2000; Ye et al. 2022), when marginal distributions differ (*i.e.* $p_S(X) \neq p_T(X)$)

- **Concept shift** a.k.a. correlation shift (Ye et al. 2022), when posterior distributions differ (*i.e.* $p_S(Y|X) \neq p_T(Y|X)$ and $f_S \neq f_T$).

- **Target Shift (TarS)** (Zhang et al. 2013), when target label proportions differ (*i.e.* $p_S(Y) \neq p_T(Y)$ and $p_S(X|Y) = p_T(X|Y)$).

- **Generalized Target Shift (GeTarS)** (Zhang et al. 2013), when both marginal and posterior distributions differ (*i.e.* $p_S(X) \neq p_T(X), p_S(Y|X) \neq p_T(Y|X)$ or equivalently $p_S(Y) \neq p_T(Y), p_S(X|Y) \neq p_T(X|Y)$)

All methods for handling domain shift leverage different labeled domains for training ($d_{tr} > 1$) and can be applied to multiple target domains. For simplicity, we consider the setting where inference is performed on a single target domain ($T = D^{d_{tr}+1}$). They differ by their assumptions:

- UDA (Section 2.1.3.1) and test-time adaptation methods like meta-learning (Section 2.1.3.2) assume access to some (limited) samples from T (unlabelled respectively labelled). While UDA uses this additional information at training time, meta-learning uses it at test time for fast adaptation. This additional information is necessary for handling concept shift settings as in Chapters 4 and 6 but is also helpful for handling the other shifts *e.g.* CovS in Chapter 3.

- DG (Section 2.1.3.3) assumes that no samples nor labels from domain T are available at training or test time. This problem is attractive as it requires less input, yet, it is ill-posed under concept shift as the target labeling function cannot be recovered. Interestingly, we were able to provide a DG method with theoretical guarantees in the covariate shift setting (Chapter 5).

We illustrate both methods in Figure 2.1 and provide some related work for each of these methods in the following sections.

2.1.3.1   Unsupervised Domain Adaptation (UDA)

In the multi-source UDA setting, we are given unlabelled target samples from domain T. We consider the standard UDA setting (Pan et al. 2010) with only a single train domain *i.e.* $d_{tr} = 1$. The standard strategy is to train a classifier using source labels while handling explicitly distribution shifts, using the available information from T. This strategy was generalized in the multi-source setting (Zhao et al. 2018).

Learning Domain-Invariant Representations   Recent methods proposed to handle distribution shift by aligning the source and target distributions in a joint latent space and minimizing a classification term on source data embeddings (Ganin et al. 2016a; Shen et al. 2018; Long et al. 2015). This is performed by defining $\mathcal{R}$ in Eq. (2.5) as a measure of distance between latent distributions $p_S(Z)$ and $p_T(Z)$ defined by encoder g. There are two main choices for this distance:

- The seminal work of Ganin et al. 2015 sets it to $\mathcal{H}$-divergence, approximated via Adversarial (ADV) training. This work has been extended in several papers (Tzeng et al. 2017; Long et al. 2018a).

- Another direction is to choose an Integral Probability Metrics, *e.g.* Maximum Mean Discrepancy (MMD) (Long et al. 2015) or Wasserstein Distance (Optimal Transport (OT)) (Shen et al. 2018; Damodaran et al. 2018; Courty et al. 2017a; Courty et al. 2017b). Wasserstein distance can be computed in the primal form with linear programing (Courty et al. 2017a; Damodaran et al. 2018) or in the dual-form form with ADV training as in Shen et al. 2018.

Learning to Transfer Representations While domain-invariant approaches are state-of-the-art, they are subject to structure issues posed by the invariance constraint which may degrade target discriminativity (Liu et al. 2019; Chen et al. 2019b). An alternative to avoid this issue is to learn to map fixed source samples to target ones (Courty et al. 2017b; Hoffman et al. 2018). This is achieved with a well-chosen $\mathcal{R}$ in Eq. (2.5). It takes one of the two following forms:

- Courty et al. 2017b leverages linear programing to compute a barycentric mapping in input space.

- Hoffman et al. 2018 leverages CycleGAN mappings, based on Generative Adversarial Network (GAN).

Existing UDA approaches are mostly designed for standard problems and fail under more complex settings *e.g.* missing data or complex domain shift. Chapters 3 and 4 handle these limitations.

2.1.3.2 Meta-Learning

Meta-learning (Thrun et al. 1998) is a general framework for fast adaptation to new domains. It learns to adapt to a new domain T using few observations, instead of learning a domain-invariant function as in Equations (2.4) and (2.5).

A standard meta-learning approach is Gradient-Based Meta Learning (GBML), which learns a model initialization θ^c on a set of training task for fast adaptation to new tasks. The standard GBML method is MAML (Finn et al. 2017) which performs bi-level optimization as summarized in Algorithm 2.1.

At training time, it learns the model initialization θ^c on a set of d_{tr} training domains via the outer-loop:

$$\min_{\theta^c} \sum_{i=1}^{d_{tr}} \mathcal{L}(\theta_i'(\theta^c), \mathcal{D}_{D^i})$$

where $\forall i \in \{1, \cdots, d_{tr}\}$:

$$\theta_i'(\theta^c) = \theta^c - \eta \sum_{k=0}^{K} \nabla_\theta \mathcal{L}(\theta_k, \mathcal{D}_{D^i}) \text{ where } \begin{cases} \theta_{k+1} = \theta_k - \eta \nabla_\theta \mathcal{L}(\theta_k, \mathcal{D}_{D^i}) & k > 0 \\ \theta_0 = \theta^c & k = 0 \end{cases}$$

This is called the inner-loop (which consists of K gradient steps).

At adaptation time for a task $d_{tr} + 1$, only the inner-loop is performed:

$$\theta_{d_{tr}+1}' = \theta^c - \eta \sum_{k=0}^{K} \nabla_\theta \mathcal{L}(\theta_k, \mathcal{D}_{D^{d_{tr}+1}}) \text{ where } \begin{cases} \theta_{k+1} = \theta_k - \eta \nabla_\theta \mathcal{L}(\theta_k, \mathcal{D}_{D^{d_{tr}+1}}) & k > 0 \\ \theta_0 = \theta^c & k = 0 \end{cases}$$

MAML was extended in various work. ANIL (Raghu et al. 2020) restricts MAML to the last layer of the classifier while other work improve adaptation by preconditioning the gradient (Lee et al. 2018; Flennerhag et al. 2020; Park et al. 2019) *e.g.* Meta-SGD (Li et al. 2017b) learns dimension-wise inner-loop learning rates. Contextual meta-learning approaches in Zintgraf et al. (2019) and Garnelo et al. (2018) partition parameters into context parameters adapted on each task, and parameters shared across tasks.

Algorithm 2.1 MAML Pseudo-code

Require: d_{tr} training domains $\{D^i\}_{i=1}^{d_{tr}}$ and test domain $D^{d_{tr}+1}$.

1: *Training:*

2: **loop**

3:     **for** $i = 1$ to d_{tr} **do**

4:         $\theta_i'(\theta^c) = \theta^c - \eta \sum_{k=0}^{K} \nabla_\theta \mathcal{L}(\theta_k, \mathcal{D}_{D^i})$ where $\begin{cases} \theta_{k+1} = \theta_k - \eta \nabla_\theta \mathcal{L}(\theta_k, \mathcal{D}_{D^i}) & k > 0 \\ \theta_0 = \theta^c & k = 0 \end{cases}$

5:     **end for**

6:     $\theta^c \leftarrow \theta^c - \eta \nabla_\theta \sum_{i=1}^{d_{tr}} \mathcal{L}(\theta_i'(\theta), \mathcal{D}_{D^i})$

7: **end loop**

8: *Inference:*

9: $\theta_{d_{tr}+1}' = \theta^c - \eta \sum_{k=0}^{K} \nabla_\theta \mathcal{L}(\theta_k, \mathcal{D}_{D^{d_{tr}+1}})$ where $\begin{cases} \theta_{k+1} = \theta_k - \eta \nabla_\theta \mathcal{L}(\theta_k, \mathcal{D}_{D^{d_{tr}+1}}) & k > 0 \\ \theta_0 = \theta^c & k = 0 \end{cases}$

2.1.3.3 Domain Generalization (DG)

In the DG setting, we do not have acces to target samples. We review two common techniques for DG and refer to the survey (Wang et al. 2021b) for more details.

Learning Domain-Invariants The most standard technique extends the ERM objective to learn domain invariants. The goal is to detect the causal mechanism

rather than memorize correlations. One approach is to use robust optimization (Sagawa et al. 2020) which optimize for worst-case performance to improve robustness to subpopulation shift. Another is to optimize for ERM with some regularization term to account for the non-IID nature of data. This is achieved with the objective in Eq. (2.5), where the first term $\mathcal{L}$ is the loss summed across labelled domains and the second term $\mathcal{R}$ is expressed either on:

Representations as in multi-source extensions of UDA methods, DANN (Ganin et al. 2016b) and CORAL (Sun et al. 2016) (Section 2.1.3.1). $\mathcal{R}$ is set to a distance metric between latent marginal distributions across domains, which is minimized with respect to the encoder's parameters to enforce invariance. The classifier is jointly trained on these invariant representations such that the encoder and classifier minimize $\sum_{i=1}^{d_{tr}} \mathcal{L}(\theta, \mathcal{D}_{D^i})$.

The predictor as in Invariant Risk Minimization (IRM) (Arjovsky et al. 2019b) or Risk Extrapolation (REx) (Krueger et al. 2021). IRM aims at finding a predictor that works well on average across domains, while being also optimal for each individual environment as in Eq. (2.6). With g the encoder and f the classifier, this objective writes as:

$$
\min_{f,g} \sum_{i=1}^{d_{tr}} \mathcal{L}(f \circ g, \mathcal{D}_{D^i})
$$
$$
\text{subject to } f \in \arg\min_{\bar{f}} \mathcal{L}(\bar{f} \circ g, \mathcal{D}_{D^i}) \quad \forall i \in \{1, \cdots, d_{tr}\}
$$
(2.6)

This complex bi-level optimization is simplified in IRMv1 into a regularized ERM objective in Eq. (2.7) which shares the form of Eq. (2.5), where $\mathcal{R}$ is set to the norm of the gradient of the domain's risk w.r.t. a fixed "dummy" classifier (a constant scalar multiplier of 1.0 per output dimension), summed across domains.

$$
\min_{g} \sum_{i=1}^{d_{tr}} \mathcal{L}(g, \mathcal{D}_{D^i}) + \lambda \|\nabla_{f|f=1.0} \mathcal{L}(f \circ g, \mathcal{D}_{D^i})\|^2
$$
(2.7)

REx performs robust optimization of the predictor over a perturbation set of extrapolated domains as in Eq. (2.8), where λ_{min} (which may be negative) controls how much we extrapolate.

$$
\min_{\theta} \max_{\sum_i \lambda_i = 1; \lambda_i \leq \lambda_{min}} \sum_{i=1}^{d_{tr}} \lambda_i \mathcal{L}(\theta, \mathcal{D}_{D^i})
$$
(2.8)

It introduces a simpler variant (V-REx) in Eq. (2.9), which also follows the form in Eq. (2.5) where $\mathcal{R}$ is set to the variance of training risks.

$$\min_{\theta} \sum_{i=1}^{d_{tr}} \mathcal{L}(\theta, \mathcal{D}_{D^i}) + \lambda \mathrm{Var}(\{\mathcal{L}(\theta, \mathcal{D}_{D^i})\}_{i=1}^{d_{tr}}) \tag{2.9}$$

The domain-specific gradients as in Fish (Shi et al. 2021). Fish sets $\mathcal{R}$ as the negative value of the inner product across domains of the expected gradients. The expected gradient for domain i is denoted G_i.

$$\min_{\theta} \sum_{i=1}^{d_{tr}} \mathcal{L}(\theta, \mathcal{D}_{D^i}) - \lambda \frac{2}{d_{tr}(d_{tr}-1)} \sum_{\substack{i,j \in [\![1,d_{tr}]\!]^2}}^{i \neq j} G_i \cdot G_j \tag{2.10}$$

This objective aligns gradient direction across domains.

Ensembling   However, these domain-invariant approaches do not outperform ERM on various benchmarks (Gulrajani et al. 2021b; Ye et al. 2022; Koh et al. 2021). In contrast, ensembling of deep NN (Lakshminarayanan et al. 2017; Hansen et al. 1990; Krogh et al. 1995) consistently increases robustness (Ovadia et al. 2019) and was successfully applied to DG without explicit domain-invariant regularizations (Arpit et al. 2021; Thopalli et al. 2021; Mesbah et al. 2022; Li et al. 2022; Lee et al. 2022; Pagliardini et al. 2022). As highlighted in Ueda et al. 1996, ensembling works due to the diversity among its members. This diversity comes primarily from the randomness of the learning procedure (Lakshminarayanan et al. 2017) and can be increased with different hyperparameters (Wenzel et al. 2020), data (Breiman 1996; Nixon et al. 2020; Yeo et al. 2021), augmentations (Wen et al. 2021; Rame et al. 2021b) or with regularizations (Rame et al. 2021a; Lee et al. 2022; Pagliardini et al. 2022; Pang et al. 2019; Teney et al. 2021).

Recent work (Izmailov et al. 2018; Draxler et al. 2018; Guo et al. 2022; Zhang et al. 2019) combine in weights (rather than in predictions) models collected along a single run. This was shown suboptimal in IID (Ashukha et al. 2020) but successful in OOD (Cha et al. 2021; Arpit et al. 2021). Following the linear mode connectivity (Frankle et al. 2020; Nagarajan et al. 2019) and the fact that many independent models are connectable (Benton et al. 2021), a second group of works average weights with fewer constraints (Wortsman et al. 2022b; Matena et al. 2021; Wortsman et al. 2022a; Gupta et al. 2020; Choshen et al. 2022; Wortsman et al. 2021). To induce greater diversity, Maddox et al. 2019 used a high constant learning rate; Benton et al. 2021 explicitly encouraged the weights to encompass more volume in the weight space; Wortsman et al. 2021 minimized cosine similarity between weights; Izmailov et al. 2019 used a tempered posterior. From a loss landscape

perspective (Fort et al. 2019), these methods aimed at "explor[ing] the set of possible solutions instead of simply converging to a single point", as stated in Maddox et al. 2019. Chapter 5 aims at proposing a new Weight Averaging (WA) approach for OOD generalization by introducing more diversity in the averaged models.

Transition   We reviewed the main strategies to handle OOD data for classification problems. In the following section, we will consider the probelem of spatiotemporal forecasting which presents some specificities over classification.

2.2   Spatiotemporal Forecasting

We now consider the problem of modeling physical processes. In particular, we focus in this work on the problem of forecasting these physical processes.

We handle two types of dynamical systems: in Section 2.2.1, we consider temporal Ordinary Differential Equation (ODE)s and in Section 2.2.2, we consider spatiotemporal Partial Differential Equation (PDE)s. For each dynamical systems, we present some standard (costly) numerical methods and recent Deep Learning (DL) alternatives. Finally, we briefly review in Section 2.2.3 some recent methods for generalizing, inspired from classification.

Preliminary notation and setting   We consider the evolution of a state v defined over a temporal domain $\mathbb{R}$ and denote $\forall t \in \mathbb{R}, v_t = v(t)$ for simplicity. As later descrived, v_t is a vector in ODEs and a function of space in PDEs.

We denote $F(v_t, \boldsymbol{\beta})$ the corresponding dynamics with some parameters $\boldsymbol{\beta}$ *e.g.* physical constants involved in the corresponding differential equation. F defines the state v_t at all times given an initial condition v_0 and includes complex transformations of space *e.g.* spatial derivatives for PDEs.

In all generality, we consider autonomous systems (*i.e.* the dynamics F do not depend on the time variable t) which follow a differential equation of the form:

$$\frac{\partial v_t}{\partial t} = F(v_t, \boldsymbol{\beta}) \tag{2.11}$$

We specify the instantiation of Eq. (2.11) in the ODE and PDE settings.

2.2.1   Ordinary Differential Equations (ODEs)

Given a function $v : \mathbb{R} \to \mathbb{R}^n$, an ODE with parameters $\boldsymbol{\beta}$ is defined by an Initial Value Problem (IVP):

$$
\frac{\mathrm{d}\boldsymbol{v}_t}{\mathrm{d}t} = F(\boldsymbol{v}_t, \boldsymbol{\beta})
$$
$$
\text{(IC) } \boldsymbol{v}_0 \in \mathbb{R}^d
$$

(2.12)

v_0 is called the initial condition (IC) and ensures unicity of a solution to the differential equation in Eq. (2.11) (Picard–Lindelöf theorem).

Such ODEs are involved in many applications *e.g.* disease diffusion, reaction rates in chemistry, stock trends in economics or Newton's second law in mechanics.

2.2.1.1   Numerical method for solving Ordinary Differential Equations

There is usually no explicit formulas for solutions of IVP. They can however be approximated using numerical schemes called ODE solvers (Hairer et al. 2000), with various criteria *e.g.* numerical stability, convergence, consistency and order.

To integrate in time an ODE with a solver, we need to discretize the temporal domain, *i.e.* define $\mathcal{T} = \{t_k\}_{k=1}^n \subset [0, T]$ s.t. $t_0 < t_1 < \cdots < t_n$. The step size is the difference between two consecutive time points *i.e.* $\Delta t_k = t_{k+1} - t_k$. There are two main ways to define the step size.

Fixed-Step   Fixed step solvers set a constant step size $\forall k, \Delta t_k = \Delta t$. The simplest approach is the explicit Euler method, whose update step writes as:

$$
\boldsymbol{v}_{(k+1)\Delta t} = \boldsymbol{v}_{k\Delta t} + \Delta t F(\boldsymbol{v}_{k\Delta t}, \boldsymbol{\beta})
$$

We also consider higher order methods such as Runge-Kutta 4 (RK4), which are more stable as they provide a finer estimation of the temporal derivatives:

$$
\boldsymbol{v}_{(k+1)\Delta t} = \boldsymbol{v}_{k\Delta t} + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \text{ where } k_1 = F(\boldsymbol{v}_{k\Delta t}, \boldsymbol{\beta}),
$$
$$
k_2 = F(\boldsymbol{v}_{k\Delta t} + \Delta t \frac{k_1}{2}, \boldsymbol{\beta}), \ k_3 = F(\boldsymbol{v}_{k\Delta t} + \Delta t \frac{k_2}{2}, \boldsymbol{\beta}), \ k_4 = F(\boldsymbol{v}_{k\Delta t} + \Delta t k_3, \boldsymbol{\beta})
$$

Adaptive   On the other hand, adaptive solvers *e.g.* Runge-Kutta 5 of Dormand-Prince-Shampine a.k.a. dopri5, adjust the step size as follows. It first produces an error estimate of the current step based on multiple different solvers with different precisions (*e.g.* the order). Then, if the error is greater than some tolerance, the step is redone with a smaller step size. This process is repeated until the error is

smaller than the provided tolerance. Adaptive solvers handle better known ODEs, especially stiff ones.

2.2.1.2   Neural ODE

Setting   Recently, NNs were used as an alternative to numerical solvers for forecasting ODEs. They are known as Neural ODE (Chen et al. 2018).

The training data consists of different trajectories *i.e.* different realizations of a same ODE with different initial conditions, obtained via a numerical solver. These trajectories are defined by an initial condition $\boldsymbol{v}_0$ and a fixed time discretization $\mathcal{T} \subset [0, T]$ where $T > 0$. A trajectory can be formally defined as the fonction $t \in \mathcal{T} \mapsto \boldsymbol{v}_t$ where

$$\forall t \in \mathcal{T}, \boldsymbol{v}_t = \boldsymbol{v}_0 + \int_0^t F(\boldsymbol{v}_\tau, \boldsymbol{\beta}) \mathrm{d}\tau \tag{2.13}$$

Eq. (2.13) integrates Eq. (2.12) in time from the initial condition $\boldsymbol{v}_0$.

Learning problem   We propose to learn F in Eq. (2.13) using a NN, f_θ, which should satisfy the following integral formulation:

$$\boldsymbol{v}_t = \boldsymbol{v}_0 + \int_0^t f_\theta(\boldsymbol{v}_\tau) \mathrm{d}\tau \tag{2.14}$$

Note that f_θ which satisfies Eq. (2.14) also satisfies the following differential form:

$$\frac{\mathrm{d}\boldsymbol{v}_t}{\mathrm{d}t} = f_\theta(\boldsymbol{v}_t) \text{ with (IC) } \boldsymbol{v}_0 \tag{2.15}$$

with the advantage that it does not require observing derivatives, which are unknown, but only the state, which is observed.

One way to find θ which satisfies Eq. (2.14) is to relax the equality constraint with a Mean-Squared Error (MSE). The corresponding loss over a set of trajectory $\mathcal{D}$ is then defined as:

$$\min_\theta \mathcal{L}(\theta, \mathcal{D}) \triangleq \min_\theta \sum_{\boldsymbol{v} \in \mathcal{D}} \sum_{t \in \mathcal{T}} \|\boldsymbol{v}_t - \tilde{\boldsymbol{v}}_t\|_2^2 \text{ where } \tilde{\boldsymbol{v}}_t = \boldsymbol{v}_0 + \int_0^t f_\theta(\tilde{\boldsymbol{v}}_\tau) \mathrm{d}\tau \tag{2.16}$$

Choosing the solver for integration   Any solver in Section 2.2.1.1 can be used to perform integration in Eq. (2.16). In practice, we observed that adaptive solvers underperform compared to fixed-step ones as the step size is not correctly estimated at the beginning of training, due to the mismatch between f_θ and F.

Computing gradients   There are two different ways to compute gradients w.r.t. θ in Eq. (2.16) to solve this optimization problem:

- "Differentiate then discretize" (Chen et al. 2018) solves the integration in Eq. (2.16) forward in time with a chosen numerical solver. Then, the gradients w.r.t. θ are computed by solving another differential equation backward in time. This is called the "adjoint state method" (*Mathematical Theory of Optimal Processes* 1962), a time-continuous formulation of backpropagation that leverages the reversibility of ODEs. It is memory efficient as past states can be recomputed on the fly without needs for storing them in memory, but is also prone to numerical errors of ODE solvers *e.g.* associated with backwards ODE solve (Gholaminejad et al. 2019).

- "Discretize then differentiate" obtains these gradients by backpropagating through the steps of the forward pass by a differentiable solver. This is similar to backpropagation in Recurrent Neural Networks. We adopted this formulation as it proved to be more stable on some preliminary experiments. Yet, it presents some limitations, which were not restrictive for us. First, it can be memory intensive as all states need to be stored in memory. Second, the solver should be differentiable unlike the first approach[1].

2.2.2   Temporal Partial Differential Equations (PDEs)

We now consider that v_t is a function of space *i.e.* $v : \mathbb{R} \times \Omega \to \mathbb{R}^n$, where $\Omega \subset \mathbb{R}^p$ is a compact domain of spatial coordinates. In other words, v_t is a spatial function of $\boldsymbol{x} \in \Omega$, with vectorial output $v_t(\boldsymbol{x}) \in \mathbb{R}^n$. A temporal PDE over v, with parameters $\boldsymbol{\beta}$ is defined by the following Initial Boundary Value Problem (IBVP):

$$
\begin{aligned}
&\frac{\partial v_t}{\partial t} = F(v_t, \frac{\partial v_t}{\partial \boldsymbol{x}}, \frac{\partial^2 v_t}{\partial \boldsymbol{x}^2}, \cdots, \boldsymbol{\beta}) \\
&(\text{IC}) \ v_0(\boldsymbol{x}) = v^0(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \Omega \\
&(\text{BC}) \ B[v](\boldsymbol{x}, t) = 0 \quad \forall (\boldsymbol{x}, t) \in \partial\Omega \times \mathbb{R}
\end{aligned}
\tag{2.17}
$$

with some initial and boundary conditions constraints. There are various possible choices of the boundary operator B including:

- Dirichlet: $B_{\mathcal{D}}[v] = v - b_{\mathcal{D}}$ for fixed function $b_{\mathcal{D}}$.

---

1. The adjoint formulation in neural ODE (Chen et al. 2018) is derived with differentiable solvers; yet it is common to obtain the adjoint for non-differentiable solvers as commonly done *e.g.* in CFD applications.

- Neumann: $B_{\mathcal{N}}[v] = n^\top \partial_{\boldsymbol{x}} v - b_{\mathcal{N}}$ for scalar-valued u, where n is an outward facing normal on $\partial\Omega$.

- Periodic: in the 2D case, for a period of p_i along the ith dimension, $B_{\mathcal{P}}[v](\boldsymbol{x}, t) = [v(x_1, x_2, t) - v(x_1 - p_1, x_2, t), v(x_1, x_2, t) - v(x_1, x_2 - p_2, t)]$.

We consider well-posed IBVP, where a unique solution exists which depends continuously on the input. Often, this requires restrictive constraints such as multiple boundary conditions to guarantee unicity.

2.2.2.1 Numerical approaches for solving Partial Differential Equations

Method of Lines   A common technique to solve numerically a PDE is the method of lines (Schiesser 1991). It discretizes domain Ω and solution v respectively into a grid $\mathcal{X} = \{\boldsymbol{x}_0, \cdots, \boldsymbol{x}_d\}$ and vector $\boldsymbol{v}_t = [v_t(\boldsymbol{x}_1), \cdots, v_t(\boldsymbol{x}_d)]^\top \in \mathbb{R}^d$. $\mathcal{X}$ is usually irregular with a higher density of points where the solution and/or its derivatives change rapidly for better modeling. The PDE is transformed into an ODE on each spatial location which can be solved with numerical methods as in Section 2.2.1.

Computing Spatial Derivatives   Each ODE follows the dynamics of F. The temporal derivatives can be formed by approximating spatial derivatives on the grid $\mathcal{X}$; there are four classical techniques for this task which we review briefly below. We refer to Morton et al. 2005 for more details.

- finite difference methods (FDM) uses difference quotients a.k.a. stencils. FDM are simple and efficient but suffer from instability when the spatial and time discretizations are not carefully controlled.

- finite volume methods are more stable and accurate than FDM but can only be applied to conservation form equations.

- pseudospectral methods compute these derivatives in Fourier space, where they take a simpler form.

- finite element methods (FEM) divide the region of interest into smaller subregions, called finite elements. They then approximate the solution by piecewise polynomial functions defined on these elements.

There have been various attemps at replacing (partially) these costly PDE numerical solvers, which we review in the following sections.

2.2.2.2   Solving Partial Differential Equations with Neural Networks

An initial attempt is Physics-Informed Neural Networks (PINN) (Raissi et al. 2019). PINN considers that initial / boundary conditions, the PDE and its parameters β are known. It parameterizes v by a coordinate-based NN a.k.a. Implicit Neural Representation (INR), denoted v_θ, which takes as input the spatial coordinates $x \in \Omega$ and time $t \in \mathbb{R}$. θ is optimized such that v_θ satisfies the given PDE. The spatial and temporal derivatives are computed using automatic differentiation. This approach does not require data but is known to suffer from ill-conditioned gradients (Krishnapriyan et al. 2021).

2.2.2.3   Neural PDE forecasters

An alternative constructs a data-driven estimation of F, via a NN f_θ, to model some observed data without any knowledge of the underlying PDE. Data is obtained using a numerical solver (Section 2.2.2.1) and is discretized into a free-form spatial observation grid $\mathcal{X} \subset \Omega$ and on discrete times $\mathcal{T} \subset [0, T]$ where $T > 0$.

We distinguish three main families of neural PDE forecasters, which we review with their advantages and limitations.

Discretized models   Most models use discretized models, *e.g.* Convolutional Neural Network (CNN) or Graph Neural Network (GNN) to process the observations. CNNs require observations on a regular grid but can be extended to irregular grids through interpolation (Chae et al. 2021). GNNs are more flexible as they handle irregular grids, at an additional memory and computational cost. These encoders were extended to account for multi-scale / global information in UNet-based architectures (Ronneberger et al. 2015). Yet, prediction on new grids $\mathcal{X}' \neq \mathcal{X}$ fails experimentally for all these models, as they are not able to generalize outside the training grid $\mathcal{X}$.

We distinguish two types of temporal models which both extrapolate beyond the train horizon due to their sequential nature.

- Autoregressive models $v_t|_{\mathcal{X}} \mapsto v_{t+\Delta t}|_{\mathcal{X}}$ (Long et al. 2018b; Bézenac et al. 2018a; Pfaff et al. 2021; Brandstetter et al. 2022; Gupta et al. 2023). These models predict the sequence from t only at fixed time increments Δt and not in between.

- Time-continuous extensions using numerical ODE solvers $(v_t|_{\mathcal{X}}, \tau) \mapsto v_{t+\tau}|_{\mathcal{X}}$ (Yin et al. 2021b; Iakovlev et al. 2021) solve this limitation as they provide a prediction at arbitrary times $t + \tau$, thus remove dependency on the observed time discretization $\mathcal{T}$. Time discretization is performed by the solver.

Operator learning   Recently, operator-based models proposed space-continuous models, which aim at finding a parameterized mapping between functions. Neural operators (Kovachki et al. 2021) attempt to replace standard convolution with continuous alternatives. Fourier Neural Operator (FNO, Li et al. 2021c) applies convolution in the spectral domain via Fast Fourier Transform (FFT). Graph Neural Operator (GNO, Li et al. 2020b) performs convolution on a local interaction grid described by a graph. DeepONet (Lu et al. 2021) uses a coordinate-based NN to output a prediction at arbitrary time and space locations given a function observed on a fixed grid. Three types of temporal models were used for operators with some limitations.

- The standard approach, $v_0 \mapsto v_t$, models the output at a given time $t \in [0, T]$ within the train horizon (Li et al. 2020b);

- A sequential extension, $v_t \mapsto v_{t+\Delta t}$, was proposed in Li et al. 2021b.

- Finally, a time-continuous version $v_0 \mapsto (t \in [0, T] \mapsto v_t)$ in DeepONet propose a solution at arbitrary time and space locations.

The first and third approaches cannot generalize beyond the train horizon, *i.e.* when $t > T$ as they are not sequential. The second solves this limitation but can only predict solutions from t at fixed time increments of Δt and not in-between.

INRs   Another class of models is based on coordinate-based NNs, called INR (Sitzmann et al. 2020; Fathony et al. 2021; Tancik et al. 2020). These space-continuous models share a similar objective as operators, despite constituting a separate research field. INRs for spatiotemporal data take time as an input along spatial coordinates. PINNs (Raissi et al. 2019) in Section 2.2.2.2 use this formulation to solve PDEs, yet are limited to a single known differential equation and a set of initial and boundary conditions. Fresca et al. 2020 propose an agnostic INR approach to build reduced order models for electrophysiology. Extensions for multi-sequence learning, *e.g.* for video generation (Yu et al. 2022; Skorokhodov et al. 2022) or compression (Chen et al. 2021), learn a latent conditioning variable from an initial condition v_0, *i.e.* take the form $v_0 \mapsto (t \in [0, T] \mapsto v_t)$. Interestingly, these models can predict at an arbitrary time t in the train horizon without unrolling a sequential model up to t.

2.2.3   Generalization Strategies for Spatiotemporal Modeling

Unlike for classification, the topic of generalization in spatiotemporal modeling is new and not well-defined. Only a few contributions were proposed as of today and handle some specific generalization settings *e.g.* when:

1. the system parameters β vary,

2. the initial and boundary conditions vary,

3. the spatiotemporal grid changes *e.g.* its resolution or regularity,

4. the spatiotemporal evaluation locations differ from training ones.

5. the test time horizon goes beyond the training horizon.

Handling these generalization problems is an emerging research topic, which we started to adress in this thesis. We review some major existing directions.

Hybrid modeling   One way to handle to better generalize is to combine prior physical knowledge on the PDE and data a.k.a. hybrid modeling. Physical model generalize in any contexts where the underlying physical law applies unlike NNs. Yet, they may be incomplete. Combining them to data-driven approaches benefits both models (Yin et al. 2021b).

Conditioning on known information   Often, some external information is available for adapting to new settings. Brandstetter et al. 2022 proposed to condition its forecaster on a known boundary condition. Some methods condition on known system parameters (Gupta et al. 2023; Pan et al. 2022) and time-scales (Gupta et al. 2023). This allows the models to consider changing conditioning information at test time. Yet this available information may not always be available.

Conditioning on observed data for adaptation   A more general strategy is to condition on observed data to infer the target dynamics. This removes the need for incorporating external information. This strategy was applied to handle changing parameters (1). This setting can be assimilated to a concept shift problem as the target labeling function differs from the training one. Several methods inspired from classification methods (Section 2.1) were proposed to handle this setting. For example, LEADS (Yin et al. 2021a) follows a MTL approach. It leverages observed data across several domains each associated to an unknown system parameter. It performs adaptation in functional space by expressing a neural dynamics model as the sum of an invariant model and of a domain-specific function, learned on the conditioning data. It was demonstrated to learn efficiently from a set of dynamical systems in the MTL setting and to provide a good initialization for adaptation to

new environments. Another approach, HANIL (Lee et al. 2021) considers meta-learning for Hamiltonian systems, a specific instance of dynamical system that conserves the Hamiltonian. They demonstrated that for modeling such systems, it is possible to learn a NN that adapts to new system parameters at test time. These existing work consider restrictive settings, which we resolve in Chapter 6.

Spatiotemporal generalization   To handle settings (3,4,5), various methods were proposed to better account for the continuity of the underlying physical systems. This is the motivation of neural operators, detailed in Section 2.2.2. The practical implementation of neural operators does not however satisfy space and time continuity such that they are unable to handle many spatiotemporal extrapolation settings. We propose an alternative time and space continuous model in Chapter 7 which leverages the flexibility of INRs and neural ODEs.

Part II

DOMAIN ADAPTATION

In the following chapters, we consider the problem of Domain Adaptation (DA), when some data from the target domain are available at training time.

In Chapters 3 and 4, we consider the problem of learning representations for classification under Unsupervised Domain Adaptation (UDA) between a labelled source and an unlabelled target domain. We adress the limitations in existing approaches which learn domain-invariant representations under two problem settings: missing data (Chapter 3) and complex domain shifts (Chapter 4).

The main contributions are outlined in Section 1.3.1.

Chapter 3

Learning invariant representations under non-stochastic missing data

Chapter abstract

We consider UDA for classification problems in the presence of missing data in the unlabelled target domain. More precisely, motivated by practical applications, we analyze situations where distribution shift exists between domains and where some components are systematically absent on the target domain without available supervision for imputing the missing target components. We propose a generative approach for imputation. Imputation is performed in a domain-invariant latent space and leverages indirect supervision from a complete source domain. We introduce a single model performing joint adaptation, imputation and classification which, under our assumptions, minimizes an upper bound of its target generalization error and performs well under various representative divergence families ($\mathcal{H}$-divergence, Optimal Transport (OT)). Moreover, we compare the target error of our Adaptation-imputation framework and the "ideal" target error of a UDA classifier without missing target components. Our model is further improved with self-training, to bring the learned source and target class posterior distributions closer. We perform experiments on three families of datasets of different modalities: a classical digit classification benchmark, the Amazon product reviews dataset both commonly used in UDA and real-world digital advertising datasets. We show the benefits of jointly performing adaptation, classification and imputation on these datasets.

The work in this chapter has led to a journal paper presented at a conference:

- M. Kirchmeyer, P. Gallinari, A. Rakotomamonjy, and A. Mantrach (2021). "Unsupervised domain adaptation with non-stochastic missing data". In: *Joint European Conference on Machine Learning and Knowledge Discovery in*

Databases (ECML-PKDD) & Data Mining and Knowledge Discovery (DMKD) 35.6, pp. 2714–2755.

3.1   Introduction

Motivated by real applications, we consider a classification problem where: (1) a source and target domain are available with observed source labels and missing target labels, (2) a distribution shift exists between source and target on joint distributions in the input and label space, (3) source input data are fully available while target data have missing input components, which cannot be measured on this domain and (4) there is no possible supervision in the target domain for imputation, thus requiring indirect supervision from the source domain. Furthermore, unobserved features contain complementary information not present in the observed ones so that the former cannot be inferred directly from the latter. (1) and (2) correspond to the classical setting of UDA, (3) corresponds to a missing data imputation problem on the target with the difficulty (4). Rubin 1976; Little et al. 1986 distinguish three categories of missing data problems based on a missingness mechanism denoted ϕ. Let $\mathbf{m}$ define a pattern of missing data, ϕ defines the conditional distribution $p_\phi(\mathbf{m}|\mathbf{x})$ where $\mathbf{x}$ represents a sample. Missing Completely at Random (MCAR) problems verify $\forall \mathbf{x}, p_\phi(\mathbf{m}|\mathbf{x}) = p_\phi(\mathbf{m})$, Missing At Random, $\forall \mathbf{x}, p_\phi(\mathbf{m}|\mathbf{x}) = p_\phi(\mathbf{m}|\mathbf{x}^{\mathrm{obs}})$ with $\mathbf{x}^{\mathrm{obs}}$ the observed feature and Missing Not At Random covers all the other cases. The key idea behind Rubin's theory is that $\mathbf{m}$ is a random variable with a probability distribution and specific imputation approaches were developed for each missingness setting. We consider the setting where target data have systematically missing input components. This corresponds to MCAR with the additional difficulty that $\mathbf{m}$ is deterministic, not stochastic. This problem is more difficult than classical MCAR as neither classical maximum likelihood solutions nor stochasticity in missing features can be used to reconstruct the missing information. While general adaptation and imputation problems were considered independently, there are several instances where they occur simultaneously. This has seldom been analyzed and only for specific cases. We propose a principled solution to this problem under non-stochastic missingness and present practical situations where this occurs.

There are many problems where specific features in collected data may be systematically absent on a domain. In the literature, this setting is mostly considered when dealing with data with multiple modalities. For example, in disease diagnosis in medical imaging (Cai et al. 2018), for some collected dataset, several modalities are present while they are absent on other datasets for which the corresponding equipment was unavailable. In multi-lingual text classification (Doinychko et al. 2020) some collections may be available only for a limited set of

languages. Similar considerations hold for recommendation in advertising (Wang et al. 2018) and object recognition with multi-sensor data (Tran et al. 2017). The situation which initially motivated our investigation, is the *prospecting* setting in computational advertising. The classical framework for ads on the internet is *retargeting*: users have already interacted with a set of merchant sites and they are targeted when they come back on one of these sites. Retargeting makes use of global user statistics collected on the whole set of merchant sites and of statistics from the specific site the user is browsing. Prospecting aims at targeting a user that visits a site for the first time (Aggarwal et al. 2019); while for such a user, features from his general behavior are available, there is no user information for the targeted site and the corresponding features are absent. The second issue considered is the distribution shift between domains. For instance, data may be collected on different devices as in medical imaging (Chen et al. 2019a) or background noise may affect each domain differently. This issue has given rise to the literature of DA when aiming at transferring knowledge from one domain to the other (Pan et al. 2010). The ads case described above is subject to both missing data for prospecting users and distribution shift between retargeting and prospecting users as detailed in Section 3.6.3.

We propose a model addressing the Adaptation-imputation problem defined by (1) to (4), which learns to perform imputation for the target domain with a conditional generative model. Imputation makes use of indirect supervision from the complete source domain. This allows us to handle non-stochastic missing data, while satisfying the constraints related to adaptation in a latent space and to classification. The imputation process plays an important role, providing us with information about the missing target data while contributing to the alignment and the reconstruction losses. Extensive empirical evidence on handwritten digits, Amazon product reviews and Click-Through-Rate (CTR) prediction DA problems illustrate the benefit of our model. The original contributions are the following:

- We propose a new end-to-end model for handling non-stochastic missing data with DA. It generates relevant missing information in the latent space conditionally on available information while aligning latent source and target marginals and classifying labelled instances. The joint missing-data and adaptation problem has been seldom considered and never in our context.

- We derive an adaptation and an imputation upper bounds. The first one upper bounds our model's target generalization error and is minimized explicitly by our training objective. The second one upper bounds an ideal target error corresponding to an UDA problem without missing features in the target domain.

- We improve this model by bringing the source and target class posteriors closer to one another with self-training; this is a useful heuristic when class posteriors mismatch.

- We evaluate the model on academic benchmarks and on challenging real-world advertising data and illustrate on these datasets that conditional generative models improve regression-based approaches seen in the literature.

3.2 Related work

Our problem is related to generic Machine Learning (ML) topics usually addressed separately *e.g.* UDA and imputation, for which we provide a brief overview.

UDA   We present a detailed related work on UDA in Section 2.1.3.1. The major difference with the setting we consider here is that existing work consider non-missing data in the target domain while we consider non-stochastic missing data.

Imputation   Data imputation is addressed by several methods (Little et al. 1986; Van Buuren 2018). Most approaches consider a supervised setting where (1) paired or unpaired complete and incomplete data are available, (2) missingness corresponds to a stochastic process (*e.g.* a mask distribution for tabular data) and (3) imputation is performed in the original feature space. This is different from our setting when one considers (1) reconstruction in a latent space, (2) imputation for a classification task, (3) no direct supervision and (4) fixed missingness which prevents us from exploiting the statistics from different incomplete samples leading to a much more complex problem. Recently, generative models were adapted for data imputation, *e.g.* Yoon et al. 2018 and Mattei et al. 2019 for Generative Adversarial Network (GAN)s and VAEs respectively. The general approach with generative models is to learn a distribution over imputed data which is similar to the one of plain data. This comes in many different instances and usually, generative training alone is not sufficient; additional loss terms are often used. In paired problems where each missing datum is associated to a plain version, a reconstruction term imposed by a Mean-Squared Error (MSE) contraint is added (Isola et al. 2017); in unpaired problems a cycle-consistency loss is imposed (Zhu et al. 2017). Li et al. 2019; Pajot et al. 2019 are among the very few approaches addressing unsupervised imputation in which full instances are never directly used. Both extend AmbientGAN (Bora et al. 2018) and consider stochastic missingness. Our imputation problem is closer to the one addressed in some forms of inpainting (Pathak et al. 2016), missing view imputation (Doinychko et al. 2020) or multi-modality missing data (Cai et al. 2018). These approaches are fully supervised.

The latter considers, as we do, imputation when one modality is systematically absent, but on one domain only, *i.e.* without adaptation. Ding et al. 2014; Wei et al. 2019; Wei et al. 2017 are the only papers we are aware of that consider imputation as we do. Ding et al. 2014 considers low-rank constraints and dictionary learning to guide transfer and was not used here as a baseline due to a high complexity that prevents large-scale experiments. Wei et al. 2019; Wei et al. 2017 are close to our work but assume that missing data can be reconstructed from the observed one through regression. In our setting, this is not possible: given the observed features, there are multiple possible imputations for the missing features; regression is thus meaningless and one has to learn their distribution or at least some modes. This motivates learning a generative model. Moreover, in Wei et al. 2019; Wei et al. 2017 classification occurs as a downstream task whereas our approach is end-to-end for classification, adaptation and imputation. Finally, our method is theoretically justified and addresses a challenging large size application motivated by a concrete real-world problem never handled before.

Cold-start   Cold-start occurs when making predictions or recommendations when data from the item or user of interest is not available or was not observed in the training set. The standard hypothesis is Independent and Identically Distributed (IID) data coming from the same domain. In recommender systems, several papers address cold-start and leverage auxiliary information about users or items *e.g.* user attributes, profile, social context or cross-domain information (Barjasteh et al. 2015; Sahebi et al. 2013). Cold-start is related to zero-shot learning with unobserved data where usual solutions learn a representation space using auxiliary knowledge *e.g.* grounded word embeddings with visual context (Zablocki et al. 2019). As for our problem, cold-start deals with non-stochastic missing data, but usually considers only one domain while we deal with distribution shift as well through adaptation.

3.3   Problem definition

Notations   We presented the classification and UDA setting in Sections 2.1.1 and 2.1.3.1. In short, we are given a labeled source domain S and an unlabeled target domain T and $D \in \{S, T\}$ will refer to S or T with distribution p_D. The corresponding datasets are denoted $\mathcal{D}_S, \mathcal{D}_T$ ($\mathcal{D}_T$ is unlabelled). In this paper, we consider that $\mathbf{x}_D \sim p_D(X)$ has two components, $\mathbf{x}_D = (\mathbf{x}_{\mathbf{D_1}}, \mathbf{x}_{\mathbf{D_2}})$; X_1, X_2 refer to each component with values in $\mathcal{X}_1, \mathcal{X}_2$. Given input $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{m} \in \{0, 1\}^n$ is a binary mask indicating which entries of $\mathbf{x}$ are missing (1 for missing and 0 for observed).

We define $\mathcal{Z} = \mathcal{Z}_1 \times \mathcal{Z}_2$ as the representation space built with a feature extractor g. Assuming both components $(\mathbf{x_{D_1}}, \mathbf{x_{D_2}})$ are observed, we define g as

$$g : \mathcal{X}_1 \times \mathcal{X}_2 \rightarrow \mathcal{Z}_1 \times \mathcal{Z}_2$$
$$(\mathbf{x_{D_1}}, \mathbf{x_{D_2}}) \mapsto (g_1(\mathbf{x_{D_1}}), g_2(\mathbf{x_{D_2}})) \tag{3.1}$$

where $\mathbf{z_{D_1}} = g_1(\mathbf{x_{D_1}})$, $\mathbf{z_{D_2}} = g_2(\mathbf{x_{D_2}})$. Z_1, Z_2 are the corresponding random variables. This is illustrated in Figure 3.1 (b) using digits samples. X_2 is available on the source domain but absent on the target domain. As detailed in Section 7.3.3, we learn to perform imputation in the latent $\mathcal{Z}$ space via a generative Neural Network (NN) r operating on $\mathcal{Z}$. For this we introduce a mapping $\widehat{g}$ as follows:

$$\widehat{g} : \mathcal{X}_1 \rightarrow \mathcal{Z}_1 \times \mathcal{Z}_2$$
$$\mathbf{x_{D_1}} \mapsto (g_1(\mathbf{x_{D_1}}), r \circ g_1(\mathbf{x_{D_1}})) \tag{3.2}$$

where $g_1 : \mathcal{X}_1 \rightarrow \mathcal{Z}_1$, $r : \mathcal{Z}_1 \rightarrow \mathcal{Z}_2$ and $\widehat{\mathbf{z}}_{\mathbf{D_2}} = r \circ g_1(\mathbf{x_{D_1}})$. $\widehat{Z}_2$ is the corresponding random variable built from X_1 with $r \circ g_1$ with values in $\mathcal{Z}_2$ and $\widehat{Z} = (Z_1, \widehat{Z}_2)$. This is illustrated in Figure 3.1 (a). For reasons detailed later, this mapping $r \circ g_1(\cdot)$ will be used on both S and T.
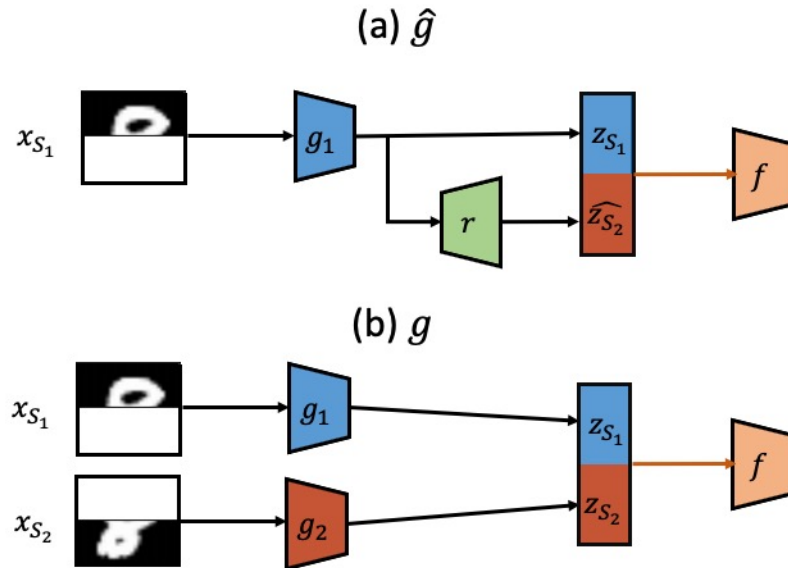
Figure 3.1. – Encoding of an input source digit $\mathbf{x_S} = (\mathbf{x_{S_1}}, \mathbf{x_{S_2}})$ with $\widehat{g}$ (a) and g (b). g_1 encodes the first part of the input $\mathbf{x_{S_1}}$ into $\mathbf{z_{S_1}}$. The second latent component is either built by encoding $\mathbf{x_{S_2}}$ with g_2 as $\mathbf{z_{S_2}}$ (b) or with reconstruction via $r \circ g_1$ as $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ (a). These latent components $\widehat{\mathbf{z}}_{\mathbf{S}}$ (a), $\mathbf{z_S}$ (b) are fed directly into a classifier f.

Assumptions   Let us now introduce formally the different assumptions underlying our context and model. We address UDA with non-stochastic missing target features and aim at finding a single hypothesis $h_{\widehat{g}} : \mathcal{X} \to \{0, ..., K\}$ of the form $f \circ \widehat{g}$, with $\widehat{g}$ the feature extractor defined in Eq. (3.2) and f the classifier with low target risk. Since the problem is under-specified, one has to make assumptions to define it properly:

Assumption 1. Labelled source data $\mathbf{x_S}$ are fully observed while unlabelled target data are partially observed with $\mathbf{x_{T_2}}$ missing. The missingness mechanism corresponds to MCAR (Little et al. 1986) on the target with fixed missingness pattern. Thus the distribution statistics of the missing data cannot be leveraged for imputation and we can only resort to indirect supervision with adaptation as in Section 7.3.3: we consider only statistics from the source to infer the imputation mechanism as later explained.

Assumption 2. The distribution of $X_2|X_1$ projected in the latent space with g from Eq. (3.1), $p_D(Z_2|Z_1)$, is multi-modal and Z_1 and Z_2 are not statistically independent. This allows to impute $\mathbf{z_{D_2}}$ given $\mathbf{z_{D_1}}$. However, regression on $\mathbf{z_{D_1}}$ cannot recover all modes as MSE produces blurry reconstructions by averaging modes. For example, assuming the feature variables Z_1, Z_2 encode the contour of the top, respectively bottom of a digit, given the bottom contours of a digit, we can reconstruct several candidates of the top contours (the bottom half of 7 can either be reconstructed into 1 or 7; regression will lead to a blurry digit averaging these two modes). As mentioned, this uncertainty is present for all our datasets.

Assumption 3. The distribution of $X_2|X_1$ projected in the latent space with g in Eq. (3.1) is the same across domains *i.e.* $p_S(Z_2|Z_1) = p_T(Z_2|Z_1)$. This allows us to make use of the source domain information (with available supervision) to infer the target conditional distribution and recover the missing latent component useful for classification. For example, assuming the feature variables Z_1, Z_2 encode the contour of the top, respectively bottom of a digit, $p(Z_2|Z_1)$ is the distribution of the contours of the bottom of the digit given those of the top; it is reasonable to assume that this distribution is the same across domains.

Assumption 4. Covariate Shift (CovS) is valid in the latent space obtained with $\widehat{g}$ in Eq. (3.2) *i.e.* $p_S(Y|\widehat{Z}) = p_T(Y|\widehat{Z})$ while $p_S(\widehat{Z}) \neq p_T(\widehat{Z})$. Thus, we can find a classifier $f \circ \widehat{g}$ with low source and target error; this is a common assumption in standard UDA.

3.4 Adaptation-Imputation model

As several generative approaches to UDA, we project source and target data onto a common latent space in which data distributions from the two domains should

match and learn a classifier using source labels. Our novelty is to offer a solution to deal with datasets with systematically missing data in the target domain. Our model, denoted *Adaptation-Imputation*, is trained to perform three operations jointly: imputation of missing information, alignment of the distributions of both domains and classification of source instances. The three operations are performed in a joint embedding space and all components are trained together with shared parameters. The term imputation is used here in a specific sense: our goal is to recover information from x_{T_2} that will be useful for adaptation and for the target data classification objective and not to reconstruct the whole missing x_{T_2}. This is achieved via a generative model, which for a given datum in T and conditionally on the available information x_{T_1}, attempts to generate the missing information. Because x_{T_2} is systematically missing for T (Assumption 1), there is no possible supervision with target samples; instead we use indirect supervision from source samples while transferring to the target. We consider two variants of the same model based on different divergence measures between distributions: the $\mathcal{H}$-divergence approximated through Adversarial (ADV) training and the Wasserstein distance (OT) computed through the primal by finding a joint coupling matrix γ with linear programming (Peyré et al. 2019). Our two models can be seen respectively as extensions of DANN (Ganin et al. 2015) and DeepJ-DOT (Damodaran et al. 2018) to the missing data problem. We only describe the ADV version in the main text, the extension to OT is detailed in Appendix A.1. Results for both models are in Section 3.6.

3.4.1 Inference

The latent space representations are denoted $\widehat{z}_D = (z_{D_1}, \widehat{z}_{D_2})$. $z_{D_1} = g_1(x_{D_1})$ is the mapping of the observed component x_{D_1} onto the latent space and $\widehat{z}_{D_2} = r \circ g_1(x_{D_1})$ is the second component's latent representation generated conditionally on x_{D_1} through generator r, as later described. At inference, given x_{T_1}, we generate $\widehat{z}_T = (z_{T_1}, \widehat{z}_{T_2})$ where $\widehat{z}_{T_2}$ encodes part of the missing information x_{T_2} in x_T (Figure 3.2 (b)). Finally $\widehat{z}_T$ is fed to the classifier f.

3.4.2 Training

For simplicity, we describe each component in turn but please note that they all interact and that their parameters are all optimized according to the three objectives mentioned above. The interaction is discussed after the description of each individual module. The model's components are illustrated in Figure 3.2 (a).

Adaptation   Adaptation aligns the distributions of $\widehat{\mathbf{z}}_{\mathbf{S}}$ and $\widehat{\mathbf{z}}_{\mathbf{T}}$ in the latent space. For ADV, alignment is performed via an adversarial loss operating on the latent representations

$$\mathcal{L}_1 = \mathbb{E}_{\mathbf{x_S} \in \mathcal{D}_S} \log D_1(\widehat{\mathbf{z}}_{\mathbf{S}}) + \mathbb{E}_{\mathbf{x_T} \in \mathcal{D}_T} \log(1 - D_1(\widehat{\mathbf{z}}_{\mathbf{T}})) \tag{3.3}$$

where $D_1(\widehat{\mathbf{z}})$ represents the probability that $\widehat{\mathbf{z}}$ comes from S rather than T.

Imputation   Imputation generates an encoding $\widehat{\mathbf{z}}_{\mathbf{T_2}}$ for the missing information, conditioned on the available $\mathbf{x_{T_1}}$ thanks to a generative model r. Since we never have access to $\mathbf{x_{T_2}}$, we develop a distant learning strategy: we learn imputation on S through $\widehat{\mathbf{z}}_{\mathbf{S_2}} = r \circ g_1(\mathbf{x_{S_1}})$ (Figure 3.2) and then transfer to the target domain ($\widehat{\mathbf{z}}_{\mathbf{T_2}}$ on the figure) via adaptation. For that we perform two operations in parallel. First, we align the distributions of $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ and $\mathbf{z}_{\mathbf{S_2}} = g_2(\mathbf{x_{S_2}})$ which is the encoding of $\mathbf{x_{S_2}}$, using an adversarial loss and discriminator D_2 ($\mathcal{L}_{ADV}$ on Figure 3.2). As alignment acts globally on distributions we have no guarantee that $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ will be associated to the corresponding $\mathbf{z}_{\mathbf{S_1}}$. We then enforce a one-to-one relationship by associating a $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ to its specific $\mathbf{z}_{\mathbf{S_1}}$. For that, we use a reconstruction term, the MSE distance between $\mathbf{z}_{\mathbf{S_2}}$ and $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ ($\mathcal{L}_{MSE}$ on Figure 3.2). This guarantees that the imputed $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ truly represents information in $\mathbf{z}_{\mathbf{S_2}}$. The learned mappings are used to perform imputation on the target data $\widehat{\mathbf{z}}_{\mathbf{T_2}} = r \circ g_1(\mathbf{x_{T_1}})$. The imputation loss $\mathcal{L}_2$ has thus two terms: an adversarial term $\mathcal{L}_{ADV}$ for aligning $\mathbf{z}_{\mathbf{S_2}}$ and $\widehat{\mathbf{z}}_{\mathbf{S_2}}$; and a reconstruction term $\mathcal{L}_{MSE}$:

$$\mathcal{L}_2 = \mathcal{L}_{ADV} + \lambda_{MSE} \times \mathcal{L}_{MSE} \tag{3.4}$$

$$\mathcal{L}_{ADV} = \mathbb{E}_{\mathbf{x_{S_2}} \in \mathcal{D}_S} \log D_2(\widehat{\mathbf{z}}_{\mathbf{S_2}}) + \mathbb{E}_{\mathbf{x_{S_1}} \in \mathcal{D}_S} \log(1 - D_2(\mathbf{z}_{\mathbf{S_2}})) \tag{3.5}$$

$$\mathcal{L}_{MSE} = \mathbb{E}_{\mathbf{x_S} \in \mathcal{D}_S} \|\mathbf{z}_{\mathbf{S_2}} - \widehat{\mathbf{z}}_{\mathbf{S_2}}\|_2^2 \tag{3.6}$$

where λ_{MSE} weights the regression term over the generative term. Imputation and adaptation influence each other and both are also influenced by classification described below. The latter forces the generated $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ to contain information about $\mathbf{x_{S_2}}$ relevant for the classification task. This information is transferred via adaptation to the target when generating $\widehat{\mathbf{z}}_{\mathbf{T_2}}$.

Classification   The last component is a classifier f, trained on source mappings $\widehat{\mathbf{z}}_{\mathbf{S}}$ as in UDA. The corresponding loss, with ℓ_{ce} a Cross Entropy (CE) loss, is

$$\mathcal{L}_3 = \mathbb{E}_{(\mathbf{x_S}, y_S) \in \mathcal{D}_S} \ell_{ce}(f(\widehat{\mathbf{z}}_{\mathbf{S}}), y_S) \tag{3.7}$$

Overall loss $\mathcal{L}$ is the weighted sum of the adaptation, imputation and classification losses

$$\mathcal{L} = \lambda_1 \times \mathcal{L}_1 + \lambda_2 \times \mathcal{L}_2 + \lambda_3 \times \mathcal{L}_3 \tag{3.8}$$

with $\lambda_1, \lambda_2, \lambda_3$ some hyperparameters and we solve

$$\min_{g_1, g_2, r, f} \max_{D_1, D_2} \mathcal{L} \tag{3.9}$$
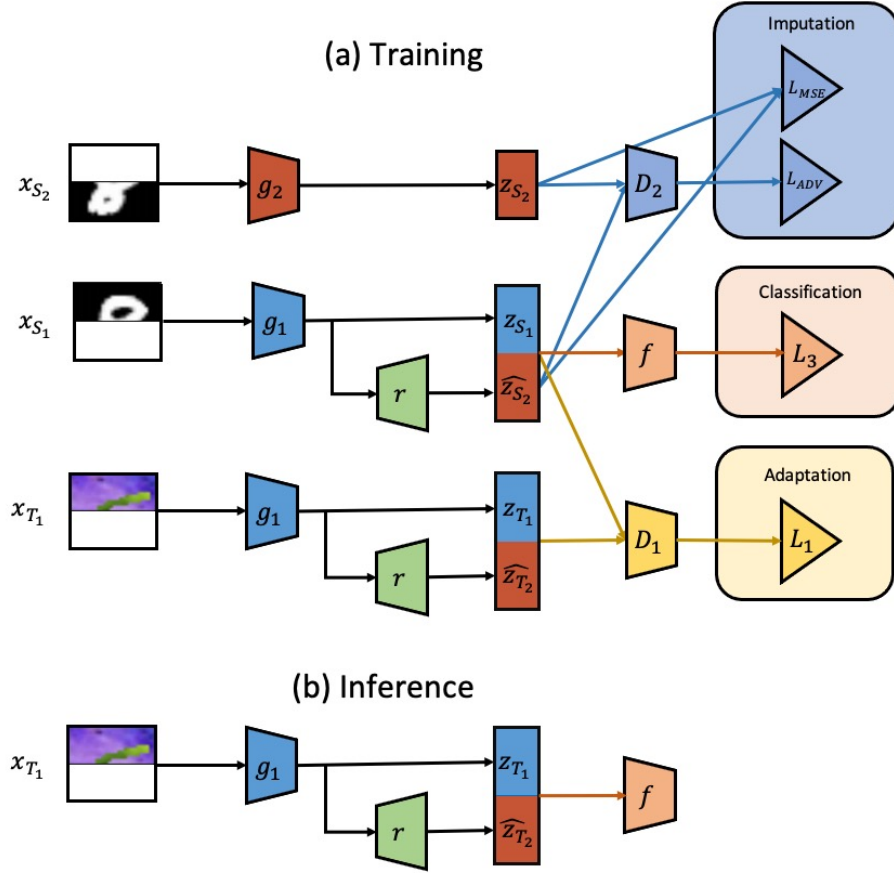
Figure 3.2. – *Adaptation-Imputation* model. The first column represents examples of raw data with missing and non-missing parts. Trapezoidal boxes represent mapping functions. Triangles in the last column represent loss functions used only for training. At training, the top-row depicts how $\mathbf{x_{S_2}}$ is mapped into the latent space with g_2. The second and third rows show how $\widehat{\mathbf{z}}_{\mathbf{S}}$ and $\widehat{\mathbf{z}}_{\mathbf{T}}$ are obtained. All these imputed and mapped source and target samples are then used in training losses. At inference, we only need the learned g_1 and r for mapping the target example with missing data into the latent space and f for predicting its class.

Interaction between the model's components   Mappings g_1, g_2, r appear in the three terms of L, meaning that they should learn to perform the three tasks simultaneously. g_1 maps $\mathbf{x_{S_1}}$ and $\mathbf{x_{T_1}}$ onto the latent space, the embeddings being denoted respectively $\mathbf{z_{S_1}}$ and $\mathbf{z_{T_1}}$. r learns to generate missing information $\widehat{\mathbf{z}}_{\mathbf{D_2}}$ from $\mathbf{z_{D_1}}$. $\widehat{\mathbf{z}}_{\mathbf{D}}$ is generated to fulfill the classification objective. g_2 should fulfill the imputation objective while preserving part of the information present in $\mathbf{x_{S_2}}$. Our model uses a unique mapping g_1 for both S and T; compared to using separate mappings, this reduces the number of parameters and performs as well.

Implementation   For adversarial training, discriminators D_1 (adaptation) and D_2 (imputation) are implemented by binary classifiers. D_1 is trained to distinguish $\widehat{\mathbf{z}}_{\mathbf{S}}$ from $\widehat{\mathbf{z}}_{\mathbf{T}}$ mappings while D_2 is trained to separate imputed $\widehat{\mathbf{z}}_{\mathbf{S_2}}$, generated from $\mathbf{x_{S_1}}$, and $\mathbf{z_{S_2}}$, a direct embedding of $\mathbf{x_{S_2}}$. We use gradient reversal layers (Ganin et al. 2015) for implementing the min-max condition on D_1 and D_2. To stabilize adversarial training, we update progressively λ_1, λ_2, respectively the hyperparameter for the adaptation loss $\mathcal{L}_1$ and the imputation loss $\mathcal{L}_2$, from 0 to 1 when updating the feature extractors g_1, g_2. Both λ_1 and λ_2 are set to 1 when updating the discriminators D_1, D_2 per Ganin et al. 2015. Moreover, we decay all learning rates. We fix $\lambda_3 = 1$ to avoid additional tuning and only tune λ_{MSE} as shown in the ablation study in Table 3.4 and Figure 3.5. All components are trained jointly after first initializing the classifier f and feature extractors g_1, g_2 to minimize $\mathcal{L}_3$ replacing $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ with $\mathbf{z_{S_2}}$ such that discriminitative components are learned before joint adaptation and imputation. Appendix A.4 provides details of all architectures and parameters.

Algorithm 3.1 Adversarial Adaptation-Imputation training procedure

N: number of epochs, k: batch size
1: Initialize f, g_1, g_2 by minimizing $\mathcal{L}_3$ replacing $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ with $\mathbf{z_{S_2}}$
2: **for** $n_{epoch} < N$ **do**
3:     Sample $\mathcal{D}_S = \{\mathbf{x_S^{(i)}}, y_S^{(i)}\}_{1 \le i \le k}$ from $p_S(X, Y)$
4:     Sample $\mathcal{D}_T = \{\mathbf{x_T^{(j)}}\}_{1 \le j \le k}$ from $p_T(X)$
5:     Decay learning rate and update gradient scale at each batch
6:     Compute $\mathcal{L} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_2 + \lambda_3 \mathcal{L}_3$ for adaptation-imputation-classification
7:     Update D_1, D_2 by ascending $\mathcal{L}$ through Gradient Reversal Layer
8:     Update f, g_1, g_2, h by descending $\mathcal{L}$
9: **end for**

3.5   Theoretical insights

3.5.1   Target generalization error

Given the model in Section 7.3.3, we now show that, despite having only unlabelled target samples, we minimize the model's target classification error using source labels with an adaptation upper bound (Theorem 3.1), under our assumptions. We then show an imputation upper bound of the "ideal" target error obtained with all components observed (classical UDA) by our model's target error times a factor (Proposition 3.2). The analytical expression of this factor highlights the role of two components: imputation on the source and transfer of this imputation from the source to the target. In the optimal case, when the model perfectly recovers these two operations, we show that our model retrieves the ideal target error. These two bounds thus provide an approach with adaptation and imputation to minimize our model's target error and reach the ideal target error, using only missing target data and source supervision for both labels and imputation.

Definitions   First, we recall some definitions. $\widehat{g}$ in Eq. (3.2) maps the first component of a sample to its imputed latent representation. $\widehat{g}$ can be applied to both source and target samples. On the other hand, g in Eq. (3.1) maps both input components on the latent space and is thus only applicable to source samples. In practice, $\widehat{g}$ and g share the same encoder for the first component g_1; the second encoder is respectively $r \circ g_1$ for $\widehat{g}$ and g_2 for g. The random variables associated to these projections are denoted respectively Z_2, for the latent missing component built from X_2 with g_2 and $\widehat{Z}_2$, for the reconstruction of Z_2 from X_1 with $r \circ g_1$. X_2 is missing on T but observed on S. Based on these mappings, we consider the risk of a hypothesis h on domain $D \in \{S, T\}$, either $h_g \in \mathcal{H}_g = \{f \circ g : f \in \mathcal{F}\}$ or $h_{\widehat{g}} \in \mathcal{H}_{\widehat{g}} = \{f \circ \widehat{g} : f \in \mathcal{F}\}$, as its error under the true labeling function f_D. In the following, we describe the adaptation and imputation bounds.

Adaptation bound   As target samples are unlabelled, we cannot directly minimize our model's target error, $\mathcal{E}_T(f \circ \widehat{g})$. In practice, we upper bound $\mathcal{E}_T(f \circ \widehat{g})$ in Theorem 3.1 with adaptation. Adaptation is performed on both components despite target missingness thanks to imputation which reconstructs the missing latent component conditionally on the observed one.

Theorem 3.1 (Proof in Appendix A.2). *Given $f \in \mathcal{F}, \widehat{g}$ in Eq. (3.2) and $p_S(\widehat{Z}), p_T(\widehat{Z})$ the latent marginal distributions obtained with $\widehat{g}$.*

$$\mathcal{E}_T(f \circ \widehat{g}) \leq \underbrace{\left[\mathcal{E}_S(f \circ \widehat{g}) + d_{\mathcal{F}\Delta\mathcal{F}}(p_S(\widehat{Z}), p_T(\widehat{Z})) + \lambda_{\mathcal{H}_{\widehat{g}}} \right]}_{\text{Domain Adaptation } (DA)} \tag{3.10}$$

with $\mathcal{E}_S(\cdot), \mathcal{E}_T(\cdot)$ the expected error under the labelling function f_S, f_T on S, T respectively; $\mathcal{F}\Delta\mathcal{F}$ the symmetric difference hypothesis space[1]; $d_{\mathcal{H}}$ the $\mathcal{H}$-divergence for $\mathcal{H} = \mathcal{F}\Delta\mathcal{F}$ and $\lambda_{\mathcal{H}_{\widehat{g}}} = \min_{f' \in \mathcal{F}}[\mathcal{E}_S(f' \circ \widehat{g}) + \mathcal{E}_T(f' \circ \widehat{g})]$, the joint risk of the optimal hypothesis.

The upper bound in Eq. (3.10) consists of $\mathcal{E}_S(f \circ \widehat{g})$ assessing the discriminative information of source latent components and $d_{\mathcal{F}\Delta\mathcal{F}}(p_S(\widehat{Z}), p_T(\widehat{Z})) + \lambda_{\mathcal{H}_{\widehat{g}}}$, assessing the transfer to the target. Our model minimizes this upper bound (DA); $\mathcal{L}_3$ in Eq. (3.7) corresponds to the first term while $\mathcal{L}_1$ in Eq. (3.3) to the second. Assumption 4 allows us to consider the third term as small. Adaptation affects both components $(Z_1, \widehat{Z}_2)$ as the missing component is imputed with $r \circ g_1$, yet, imputation here is not supervised with fully observed components.

Imputation bound   Given $f \in \mathcal{F}$, we compare under our assumptions $\mathcal{E}_T(f \circ \widehat{g})$ and the ideal target error with full data, $\mathcal{E}_T(f \circ g)$, with $g = (g_1, g_2)$ and $\widehat{g} = (g_1, r \circ g_1)$. This allows us to measure the loss in performance due to missingness when using $f \circ \widehat{g}$ instead of $f \circ g$. g_1 is shared in g and $\widehat{g}$ while $r \circ g_1$ reconstructs the missing component on both domains. We first derive Lemma 3.1 used in our upper bound in Proposition 3.2.

Lemma 3.1 (Proof in Appendix A.2). *For any continuous density distributions p, q defined on an input space $\mathcal{X}$, such that $\forall \mathbf{x} \in \mathcal{X}, q(\mathbf{x}) > 0$, the inequality $\sup_{\mathbf{x} \in \mathcal{X}}[p(\mathbf{x})/q(\mathbf{x})] \geq 1$ holds. Moreover, the minimum is reached when $p = q$.*

We derive Proposition 3.2. Under Assumption 3, given a classifier $f \in \mathcal{F}$ and encoders $g, \widehat{g}$, this proposition upper bounds $\mathcal{E}_T(f \circ g)$ with $\mathcal{E}_T(f \circ \widehat{g})$ multiplied by a factor (I_T) in Eq. (3.11). Our model minimizes both the Adaptation upper bound and the term (I_T).

Proposition 3.2 (Proof in Appendix A.2). *Under Assumption 3, let $f \in \mathcal{F}, \widehat{g}$ (Eq. (3.2)) and g (Eq. (3.1)),*

$$\mathcal{E}_T(f \circ g) \leq \underbrace{\underbrace{\sup_{\mathbf{z} \sim p(\mathcal{Z})} \left[\frac{p_S(Z_2 = \mathbf{z}_2 | \mathbf{z}_1)}{p_S(\widehat{Z}_2 = \mathbf{z}_2 | \mathbf{z}_1)} \right]}_{\text{Imputation error on S } (I_S)} \times \underbrace{\sup_{\mathbf{z} \sim p(\mathcal{Z})} \left[\frac{p_S(\widehat{Z}_2 = \mathbf{z}_2 | \mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2 | \mathbf{z}_1)} \right]}_{\text{Transfer error of Imputation } (T_I)}}_{\text{Imputation error on T } (I_T)} \times \mathcal{E}_T(f \circ \widehat{g}) \tag{3.11}$$

---

1. $h \in \mathcal{F}\Delta\mathcal{F} \iff h(\mathbf{x}) = f_1(\mathbf{x}) \oplus f_2(\mathbf{x})$ for some $f_1, f_2 \in \mathcal{F}$ where $\oplus$ is the XOR function.

Under Lemma 3.1, $(I_T) = 1$ is the minimal value reached when $p_S(Z_2|Z_1) = p_S(\widehat{Z}_2|Z_1)$ and $p_S(\widehat{Z}_2|Z_1) = p_T(\widehat{Z}_2|Z_1)$. In this case, $\mathcal{E}_T(f \circ g) = \mathcal{E}_T(f \circ \widehat{g})$.

The upper bound in Eq. (3.11) shows that for any $f, \widehat{g}, g$, $\mathcal{E}_T(f \circ g)$ is upper bounded by $\mathcal{E}_T(f \circ \widehat{g})$ times the multiplicative factor (I_T). The optimal situation, equality, is obtained when $(I_T) = 1$. (I_T) measures how imputation recovers the missing target component and is decomposed into two terms. (I_S) quantifies how imputation learns $p_S(Z_2|Z_1)$ with $p_S(\widehat{Z}_2|Z_1)$ *i.e.* reconstructs the component $Z_2 = g_2(X_2)$ with $\widehat{Z}_2 = r(Z_1)$ and $Z_1 = g_1(X_1)$ on the source. (T_I) measures the divergence of $\widehat{Z}_2|Z_1$ across domains; the lower, the better indirect imputation supervision from S transfers to T. The equality case occurs when (I_T) is minimal, i.e when $p_S(Z_2|Z_1) = p_S(\widehat{Z}_2|Z_1)$ and $p_S(\widehat{Z}_2|Z_1) = p_T(\widehat{Z}_2|Z_1)$. Our model minimizes (I_T) after first initializing f, g with $\operatorname{argmin}_{f,g} \mathcal{E}_S(f \circ g)$ replacing $\widehat{g}$ with g in $\mathcal{L}_3$, Eq. (3.7) to extract discriminative components $(\mathbf{z_{S_1}}, \mathbf{z_{S_2}})$. It minimizes (I_S) with $\mathcal{L}_2$ in Eq. (3.4) while (T_I) is minimized with the adaptation loss $\mathcal{L}_1$ in Eq. (3.3). Note that (I_T) is minimal when $\mathcal{L}_1 = \mathcal{L}_2 = 0$ yielding to the equality $\mathcal{E}_T(f \circ g) = \mathcal{E}_T(f \circ \widehat{g})$.

3.5.2 Self-training refinement

We now introduce a heuristic based on pseudo-labels useful for settings where Assumption 4 is not verified because $p_S(Y|\widehat{Z}) \neq p_T(Y|\widehat{Z})$. Assumption 4 allows to consider $\lambda_{\mathcal{H}_{\widehat{g}}}$ in Eq. (3.10) as small. Indeed, several authors *e.g.* Zhao et al. 2019; Johansson et al. 2019 recently demonstrated that minimizing the first two terms in (DA) Eq. (3.10) is not sufficient for successful UDA. They show that (1) even when CovS is true in the data space, it usually does not hold in the latent space; (2) even when the first two terms in (DA) in Eq. (3.10) are minimized, the third, $\lambda_{\mathcal{H}_{\widehat{g}}}$, might increase so that the bound is not minimized. Zhao et al. 2019 shows that in addition to the above conditions, one should enforce the posterior class distributions $p_D(Y|X)$ to be close on the two domains. Since T is unlabeled there is no direct way to do that. We instead propose a simple heuristic using pseudo-labels and show how they can be incorporated with a simple adaptation of Eq. (3.10). Pseudo-labels are tentative labels assigned to target unlabelled samples by a classifier, denoted $h_{\widehat{g}}$ below. As $\lambda_{\mathcal{H}_{\widehat{g}}}$ cannot be measured without target labels, we will approximately evaluate and minimize it with pseudo-labels.

Proposition 3.3 (Proof in Appendix A.2). *Assume a joint distribution $p_{\widetilde{T}}(X, Y)$ where $p_{\widetilde{T}}(X) = p_T(X)$ and $Y = h_{\widehat{g}}(X)$ where $h_{\widehat{g}} = f \circ \widehat{g} \in \mathcal{H}_{\widehat{g}}$ is a candidate hypothesis. Then,*

$$\lambda_{\mathcal{H}_{\widehat{g}}} \leq \min_{h_{\widehat{g}} \in \mathcal{H}_{\widehat{g}}} \left[\mathcal{E}_S(h_{\widehat{g}}) + \mathcal{E}_{\widetilde{T}}(h_{\widehat{g}}) + \mathcal{E}_T(f_{\widetilde{T}}) \right] \tag{3.12}$$

$\mathcal{E}_T(f_{\widetilde{T}}) = \Pr_{\mathbf{x} \sim p_T(X)}(f_{\widetilde{T}}(\mathbf{x}) \neq f_T(\mathbf{x}))$ *is the T error of the pseudo-labelling function $f_{\widetilde{T}}$.*

The first two terms on the right hand side of Eq. (3.12) may be controlled as we know source labels and target pseudo-labels; the third term is the error of the pseudo-labeling function, minimal if pseudo-labels are equal to true target labels. We cannot measure the last term but propose self-training as a way to heuristically improve the pseudo-labeling function.

We detail one way to do so in Algorithm 3.2. We start from an initial set of pseudo-labels, *e.g.* the pseudo-labels provided by the model in Section 7.3.3 and then refine them. Many self-training methods have been proposed. We use a combination of two such methods, initially proposed for Semi-Supervised Learning (SSL): an adaptation of the semi-supervised discriminant Classification Expectation Maximization (CEM) in Amini et al. 2005 and SSL by entropy minimization (Grandvalet et al. 2005). We found that combining these two approaches performed better than each method used alone.

In the following we assume to have a set $\mathcal{D}_S, \mathcal{D}_T$ respectively of labelled S and unlabelled T samples. Amini et al. 2005 introduce an iterative method which starts from pseudo-labels provided by an initial classifier and retrains the classifier with these labels. We start with $f(\widehat{\mathbf{z}})$ trained as in Section 7.3.3 and keep, at each iteration, all samples in $\mathcal{D}_T$ whose classification score is above a threshold, this set of pseudo-labelled instances is denoted $\mathcal{D}_T^{pl}$. We then minimize a CE loss on $\mathcal{D}_S \cup \mathcal{D}_T^{pl}$, between the labels for $\mathcal{D}_S$ or pseudo-labels for $\mathcal{D}_T^{pl}$ and the predicted scores. Grandvalet et al. 2005 optimizes an entropy loss on the distribution of the predicted class posteriors output from f for all unlabelled samples; we apply this loss to $\mathcal{D}_T \setminus \mathcal{D}_T^{pl}$. This entropy loss can be considered as a soft version of the discriminant CEM loss.

In conclusion, we first train the model without pseudo-labels minimizing L (Section 7.3.3). We then use the learned classifier to provide initial pseudo-labels and minimize jointly discriminant CEM and entropy loss to refine them. Given $h_{\widehat{g}} = f \circ \widehat{g} \in \mathcal{H}_{\widehat{g}}$ a hypothesis with $\forall k \in [1, K], h_{\widehat{g}_k}(\mathbf{x})$ the probability of predicting instance $\mathbf{x}$ to class k, ℓ_{ce} a CE loss and λ a weight for entropy, the objective function of our refinement method is:

$$\mathcal{L}_{\mathcal{R}} = \underbrace{\sum_{(\mathbf{x},y) \in \mathcal{D}_S \cup \mathcal{D}_T^{pl}} \ell_{ce}(h_{\widehat{g}}(\mathbf{x}), y)}_{\text{Discriminant CEM } (CEM)} + \lambda \underbrace{\sum_{\mathbf{x} \in \mathcal{D}_T \setminus \mathcal{D}_T^{pl}} \sum_{k=1}^{K} h_{\widehat{g}_k}(\mathbf{x}) \log h_{\widehat{g}_k}(\mathbf{x})}_{\text{Entropy } (E)} \qquad (3.13)$$

The first term in Eq. (3.13), (CEM), controls $\mathcal{E}_S(h_{\widehat{g}}) + \mathcal{E}_{\widetilde{T}}(h_{\widehat{g}})$ while the second term, (E), heuristically controls $\mathcal{E}_T(f_{\widehat{T}})$ by encouraging separation between classes. We found that this heuristically brings pseudo-labels closer to the target labels on our datasets. In practice, we minimize $\mathcal{L}_{\mathcal{R}}$ w.r.t. $f, \widehat{g}$.

Algorithm 3.2 Self-training procedure for Adaptation-Imputation

---

Input $\mathcal{D}_S = \{(\mathbf{x_S^{(i)}}, y_S^{(i)})\}_{i=1}^{N_S}$, $\mathcal{D}_T = \{\mathbf{x_T^{(i)}}\}_{i=1}^{N_T}$, Adaptation-Imputation method $\mathcal{A}$ in Section 7.3.3
Output Classifier f; Feature extractor $\widehat{g}$ defined in Eq. (3.2)
   1: $f, \widehat{g} = \mathcal{A}(\mathcal{D}_S, \mathcal{D}_T)$ ▷ Initialize $f, \widehat{g}$ with *Adaptation-Imputation* Eq. (3.8)
   2: $f, \widehat{g} = \arg\min_{f,\widehat{g}} \mathcal{L}_{\mathcal{R}}$ ▷ Semi-supervised refinement of $f, \widehat{g}$ Eq. (3.13)

3.6 Experiments

3.6.1 Datasets and experimental setting

Datasets Experiments are performed on three types of datasets. The first one, `digits`, is a classical multi-class classification benchmark used in many UDA studies and adapted to fit our missing data setting. The second one, which initially motivated our framework, consists of advertising datasets where we aim at transferring knowledge from retargeting users with full browsing information to prospecting users with missing information. The task is binary classification as measured by CTR or Conversion Rate (CR)[2] given user browsing traces. We use two such datasets: `ads-kaggle` is a public kaggle dataset[3], while `ads-real` was gathered internally. Both correspond to real advertising traffic. Finally, we performed tests on a text dataset, Amazon reviews, denoted `amazon`. The initial problem is transformed into binary classification and to a non-stochastic missing data problem. For both `digits` and `amazon`, a subset of the components are set to 0 to mimic missing data while on `ads`, data is missing structurally (more details in Appendix A.3).

Baselines We report results for the following models:

1. *Source-Full* trained without adaptation on $\mathbf{x_S}$ and tested on full $\mathbf{x_T}$; adaptation is added in *Adaptation-Full*. Note that this model is only applicable for our academic benchmark where we have access to full data.

2. *Source-ZeroImputation* and *Adaptation-ZeroImputation* do the same but considering full $\mathbf{x_S}$ while $\mathbf{x_T}$ is incomplete. Missing data $\mathbf{x_{T_2}}$ is set to $\mathbf{0}$, $\mathbf{x_T} = (\mathbf{x_{T_1}}, \mathbf{0})$.

3. *Source-IgnoreComponent* and *Adaptation-IgnoreComponent* are a variant of the above where only $\mathbf{x_{D_1}}$ is considered while $\mathbf{x_{D_2}}$ is ignored for both S and T.

---

2. CTR is the number of clicks made on ads divided by the number of shown ads. CR replaces clicks with purchases.
3. http://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset/

4. *Adaptation-Imputation*, our model, considers full $\mathbf{x_S}$ and $\mathbf{x_T} = (\mathbf{x_{T_1}}, \mathbf{0})$ adding imputation with a conditional generative model.

5. We add self-training to *Adaptation-Imputation* and if possible to *Adaptation-Full*.

Note that *Adaptation-Full* is an upper bound of our imputation model since it uses full information while $\mathbf{x_{T_2}}$ is not available in practice. *Adaptation-ZeroImputation* and *Adaptation-IgnoreComponent* are lower bounds for our model since they only perform adaptation and do not impute non-zero values.

Hyperparameters   Parameters are chosen using the DEV estimator (You et al. 2019). For `digits`, NN architectures are adapted from Ganin et al. 2015; we use Adam optimizer with $lr = 10^{-2}$ decayed; batch size of 128 and 100 epochs. For `ads` and `amazon`, three-layered NN with 128 neurons per layer are used as feature extractors; the classifier and discriminators are single-layered with 128 neurons; $lr = 10^{-6}$ and is decayed; batch size is 500 with 50 epochs. Reported results are mean value and standard deviation over five runs and best results are indicated in **bold**. Further details are given in the Appendix A.4.2.

3.6.2   Digits

Description   We consider UDA problems between several datasets: MNIST (Le-Cun et al. 1998), USPS (Hull 1994), SVHN (Netzer et al. 2011) and MNIST-M (Ganin et al. 2015) as illustrated in Figure 3.3 (a). MNIST $\rightarrow$ SVHN is not considered as it is difficult for traditional UDA (Ganin et al. 2015). All tasks are 10-class classification problems. From complete digits datasets, we build datasets with missing input values by setting corresponding pixel values to zero for horizontal patches of different sizes as illustrated on Figure 3.3 (b) for MNIST-M digits. It is clear that there is domain shift on these datasets as the pixel values have different mean and variance across domains.

Results with half of the digit missing   We first removed half of each target digit, the horizontal bottom part. We report target accuracy in Table 3.1 for both ADV and OT models. Removing half of the digit leads to a strong performance decrease for *Source-IgnoreComponent* and *Source-ZeroImputation* compared to the upper-bounds of *Source-Full*; the performance is partially recovered with adaptation. *Adaptation-Imputation* clearly improves on *Adaptation-IgnoreComponent* and *Adaptation-ZeroImputation* in all cases which validates the importance of imputation. However, it does not reach the upper bound performance of *Adaptation-Full*. Both ADV and OT versions exhibit the same behavior. In the results in Table 3.1,
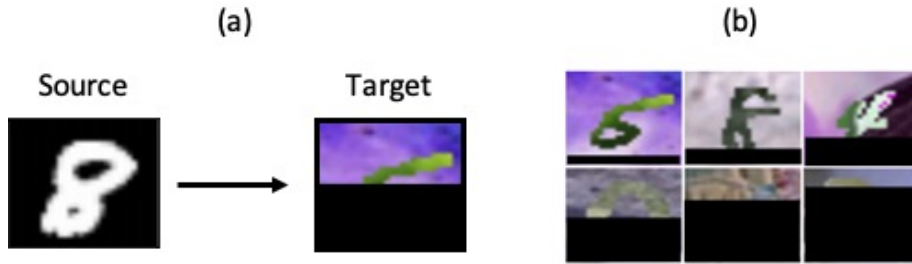
Figure 3.3. – (a) MNIST→MNIST-M adaptation; (b) Digits with missing horizontal patches of increasing size

ADV performance is higher than OT. This is because performance is highly dependent on the NN architectures and we tuned our NNs for ADV. OT models may reach performance similar to ADV but require an order of magnitude more parameters. To keep the comparison fair, we use the same NN models for both ADV and OT. Imputation models achieve their highest performance when adaptation between domains is complex (MNIST → MNIST-M, SVHN → MNIST) illustrating the importance of imputation when transfer is difficult. We show in Appendix A.5 the learned latent representations $\widehat{\mathbf{z}}_\mathbf{S}, \widehat{\mathbf{z}}_\mathbf{T}$ for various `digits` adaptation problems.

Varying missing patch size   We analyze the impact of the size of the missing patch by removing a percentage $p \in \{30\%, 40\%, 50\%, 60\%, 70\%\}$ of MNIST digits when adapting SVHN → MNIST, with the same hyperparameters. Mean values over five runs are reported in Figure 3.4 for ADV models. We notice that our model constantly beats the other baselines regardless of the missing patch size. The figure exhibits borderline cases when the size of the missing patch becomes very small ($< 30\%$) or very large ($> 65\%$). When the missing patch is small there is enough information for predicting the label thus simple models perform well; when it becomes big, there is not enough information for efficient reconstructions.

3.6.3 Ads

Description   The `ads` datasets are used for solving the binary classification problem of predicting if a **user** exposed to an ad from a **partner** (*e.g.* Booking.com) clicks given his browsing history. A row in this dataset is a vector $\mathbf{x} = (\mathbf{x}_{\mathbf{D_1}}, \mathbf{x}_{\mathbf{D_2}})$ specific to a (user-partner) pair where $\mathbf{x}_{\mathbf{D_1}}$ gathers mean statistics for this user on all visited partners summarizing the user's display and click statistics and $\mathbf{x}_{\mathbf{D_2}}$ corresponds to the user-partner specific traces. The label is the response to an ad for this (user, partner) pair, a click for `ads-kaggle` or a purchase for `ads-real`. We transfer knowledge from the labelled source domain composed of all user-partner
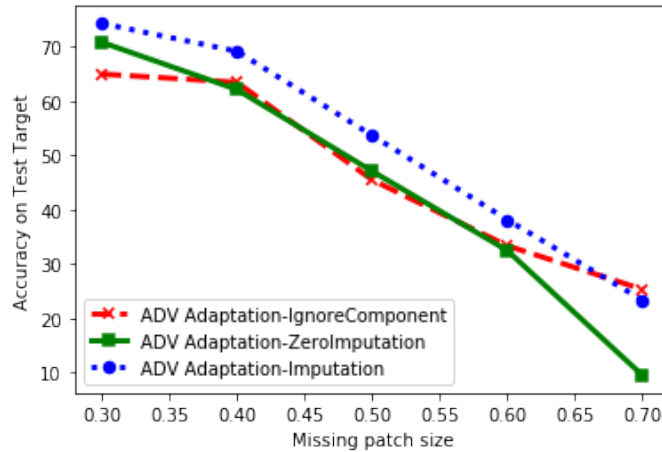
Figure 3.4. – ADV target accuracy (↑) on SVHN → MNIST with missing patch size

pairs for which the user has already interacted with the partner (**retargeting users**) to the unlabelled target domain composed of all the user-partner pairs for which the user has never interacted with this partner (**prospecting** users). x_{S_2} is known but x_{T_2} is **unknown**. There are several partners and users per domain. These datasets are large scale as seen in Table A.2 (1M and 24M source displays respectively for ads-kaggle, ads-real) with some specificities: there is class imbalance and five times less data on the target than on the source. For both datasets besides missingness, there is also an adaptation problem: prospecting users tend to be less active and their statistics are usually different from those of retargeting users, with a higher overall activity (*e.g.* in terms of frequency of a partner's website visits); this translates into distribution shifts on x_{D_1} across domains. We visualize in the Appendix in Table A.3 and Figure A.1 the domain shift in ads-kaggle which comprises 13 features. Table A.3 reports mean and standard deviation on each feature's value over a domain and Figure A.1 plots the histogram of the distribution of each feature where the y-axis is unnormalized and corresponds to real counts. Feature 5 is naturally missing on T and distributions are different in shape, mean and variance across domains. To show the benefit of modelling additional missing features, we artificially set features 1, 6, 7, 11 and 12 to zero on T such that in total 6 features are missing while 7 are present. On ads-real, 12 features are missing while 17 are present and we observe the same domain shift trend; however missing features are naturally missing and we do not have access to their value.

Results   We report results in Table 3.1 only for ADV models as we observed that the trend is similar for both ADV and OT. Missing features are structurally missing in the datasets, so we cannot report results for models using full inputs. The classes being imbalanced, accuracy is not relevant here so we report the log CE

Dataset	MNIST → USPS		USPS → MNIST		SVHN → MNIST		MNIST → MNIST-M		ads-kaggle	ads-real
Model w/o $\mathcal{R}$	ADV	OT	ADV	OT	ADV	OT	ADV	OT	ADV	ADV
Source-Full	71.5±2.7		74.2±2.7		58.1±1.1		28.3±1.4		NA	
Adaptation-Full	85.8±3.2	92.6±1.7	94.6±2.1	93.9±0.6	78.0±3.4	76.1±1.4	60.8±3.8	46.9±3.9	NA	
Source-ZeroImputation	25.7±3.7		39.2±2.6		31.5±2.		14.4±1.1		0.545±0.019	0.663±0.011
Adaptation-ZeroImputation	48.4±4.8	60.9±6.3	67.5±2.2	65.3±5.2	47.1±5.7	37.5±6.2	34.7±2.5	20.2±2.5	0.397±0.0057	0.660±0.025
Source-IgnoreComponent	52.9±9.7		54.3±1.6		44.6±1.9		19.1±2.6		0.406±0.00046	0.622±0.0048
Adaptation-IgnoreComponent	71.5±3.2	64.0±5.0	80.0±1.4	72.0±1.8	45.5±1.9	47.9±1.8	29.4±1.6	26.8±4.4	0.403±0.0030	0.634±0.0082
Adaptation-Imputation	**74.2±2.3**	**66.8±1.3**	**81.4±0.8**	72.5±2.7	**53.8±1.4**	**49.2±1.5**	57.9±2.3	29.2±1.4	**0.389±0.014**	**0.583±0.013**

Table 3.1. – Best target accuracy (↑) on digits and CE (↓) on ads without $\mathcal{R}$

between the predicted values and the true labels. CE is considered to be the most reliable metric to estimate revenue for the ads problem and for large user bases small CE improvements can lead to a large revenue increase. For ads-kaggle, an improvement of 0.001 in CE is considered as significant (Wang et al. 2017). A first observation is that the imputation model is substantially better than the baselines on both datasets. For ads-kaggle it improves by 2.3% the best adaptation model *i.e.* the adaptation model with zero imputation while for ads-real the improvement reaches 6.3% over the second-best *Source-IgnoreComponent*. A second observation is that for any model, adaptation consistently improves over the model without adaptation. The only exception is the setting ignoring the missing component in ads-real. A third observation is that there is a benefit of imputing the missing component for classification: source CE (not reported) shows that *Source-ZeroImputation* which exploits x_{D_2} is consistently higher than *Source-IgnoreComponent* which does not, leading to relative gains of 5.6% on ads-kaggle and 8.2% on ads-real. The imputation model is able to generate this information.

3.6.4 Amazon reviews

Description   Besides dealing with images and interaction features in the digits and ads datasets, we also performed experiments on an additional modality, text. amazon is the Amazon product review dataset (Blitzer et al. 2006) with four domains (Books, DVDs, Electronics, and Kitchen) transformed to binary classification with positives referring to reviews with rating above 3 stars and negatives to reviews with rating below 3 stars. Additional details on data processing can be found in Appendix A.3. We consider four adaptation problems and simulate missing features by setting the first half of the features to zero.

Results   Results are reported in Table 3.2 and confirm our prior findings *i.e.* that jointly performing adaptation and imputation improves our baselines. We also notice that our model achieves similar performance to models using full data showing that imputation successfully recovered the missing component.

Dataset	DVD → Electronics	Books → Kitchen	Kitchen → Electronics	DVD → Books
Source-Full	69.57	73.04	77.88	71.95
Adaptation-Full	73.62	74.09	79.63	72.65
Source-ZeroImputation	58.51	60.52	66.27	61.15
Adaptation-ZeroImputation	64.51	61.08	68.02	62.80
Source-IgnoreComponent	60.21	62.03	67.62	64.35
Adaptation-IgnoreComponent	61.02	64.08	68.47	66.00
Adaptation-Imputation	**72.57**	**72.69**	**78.18**	**72.61**

Table 3.2. – Best target accuracy (↑) on `amazon` without $\mathcal{R}$

ADV Model w/ $\mathcal{R}$	MNIST → USPS	USPS → MNIST	SVHN → MNIST	MNIST → MNIST-M	ads-kaggle
Adaptation-Full	95.9±0.6 (+12%)	96.8±0.6 (+2.3%)	83.3±3.9 (+6.8%)	60.9±3.7 (+0.2%)	NA
Adaptation-Imputation	78.5±1.6 (+5.8%)	82.5±0.5 (+1.4%)	58.6±1.8 (+8.9%)	58.2±2.3 (+0.5%)	0.317±0.0023 (+18.5%)

Table 3.3. – Refinement $\mathcal{R}$ with relative gain over Table 3.1; target accuracy (↑) on `digits` and CE (↓) on `ads`

3.6.5 Refinement

Results with pseudo-labels are reported in Table 3.3 on `digits` and `ads-kaggle` for *Adaptation-Full* and *Adaptation-Imputation*. We set the threshold score selection for the discriminative CEM component to 95% *i.e.* the pseudo labels of all target instances $\mathbf{x_T}$ s.t. $\max_k h_{\widehat{g}_k}(\mathbf{x_T}) \geq 0.95$ are considered to be true and set the entropy weight to $\lambda = 0.1$ on `digits` and $\lambda = 1$ on `ads-kaggle`. Learning rates used for solving Eq. (3.9) are divided by 10 and 10 epochs of successive refinement steps are applied. We observe a clear global improvement on both datasets showing that our refinement model is a good heuristic on real-world datasets for which we usually have $p_S(Y|\widehat{Z}) \neq p_T(Y|\widehat{Z})$. For standard UDA methods such as *Adaptation-Full*, performance is significantly improved everywhere with small change on MNIST → MNIST-M; *Adaptation-Full* is not measurable for `ads-kaggle`. Our imputation with refinement model follows the same trend with a considerable relative gain of $+18.5\%$ on `ads-kaggle`.

3.6.6 Ablation analysis

We analyze the importance of each model component on the public datasets (`digits`, `amazon`, `ads-kaggle`) and report results in Table 3.4 and Figure 3.5.

Adaptation   We measure the effect of adaptation term $\mathcal{L}_1$ Eq. (3.3) in L in Table 3.4 (first row). When removing adaptation, inference is performed as before by feeding $\widehat{\mathbf{z}_T}$ to f. This means that we only rely on the imputation and classification losses to learn the parameters of the model. For all datasets, adding $\mathcal{L}_1$ considerably increases performance.

Imputation   Imputation $\widehat{\mathbf{z}}_{\mathbf{S_2}} = h \circ g_1(\mathbf{x}_{\mathbf{S_1}})$, combines ADV training and conditioning on the input datum via MSE (MSE) in $\mathcal{L}_2$ Eq. (3.4). ADV aligns the distributions of $\mathbf{z}_{\mathbf{S_2}}$ and $\widehat{\mathbf{z}}_{\mathbf{S_2}}$ while MSE can be thought as performing regression. For a given $\mathbf{x}_{\mathbf{S_1}}$, there are possibly several potential $\mathbf{x}_{\mathbf{S_2}}$ and thus $\mathbf{z}_{\mathbf{S_2}}$. ADV allows us to focus on a specific mode of $\mathbf{z}_{\mathbf{S_2}}$, while MSE will favour a mean value of the distribution. Results in Table 3.4 (second row), show that for our datasets, combining MSE and ADV leads to improved results compared to using separately each loss. MSE alone already provides good performance, while using only ADV is clearly uncompetitive. Note that reconstruction is an ill-posed problem since the task is inherently ambiguous (different digits may be reconstructed from a half image). We performed tests with a stochastic input component to recover different modes, but the performance was broadly similar. We investigate in Figure 3.5 several weighted combinations of MSE and ADV: for `digits` and `amazon`, equal weights were found to be a good choice, while for `ads-kaggle` performance is improved with other weightings. On Figure 3.5, ADV induces a high variance in the results (left part of x-axis) while MSE stabilizes the performance (right part of x-axis). ADV allows for better performance at the expense of high variance; a small contribution from MSE, $\lambda_{MSE} = 0.005$, stabilizes the results.

Ablation study	ADV Model	MNIST → USPS	USPS → MNIST	SVHN → MNIST	MNIST → MNIST-M	ads-kaggle
Remove $\mathcal{L}_1$	$\mathcal{L} = \lambda_2\mathcal{L}_2 + \lambda_3\mathcal{L}_3$	64.2±1.8 (-13%)	51.3±2.5 (-37%)	44.5±1.4 (-17%)	24.1±2.6 (-58%)	0.410±0.0020 (-5.4%)
Weights in $\mathcal{L}_2$	$\mathcal{L}_{MSE}$	71.9±3.7 (-3.1%)	**81.4±1.2 (0%)**	52.5±3.7 (-2.4%)	56.5±2.8 (-2.4%)	0.400±0.0014 (-2.8%)
	$\mathcal{L}_{ADV}$	28.6±3.2 (-61%)	39.4±5.2 (-52%)	28.8±3.8 (-46%)	30.0±3.7 (-48%)	0.469±0.13 (-21%)
	$\mathcal{L}_{ADV} + 5\text{e-}3 \cdot \mathcal{L}_{MSE}$	47.8±3.7 (-36%)	49.6±5.8 (-39%)	46.0±2.6 (-15%)	50.6±2.2 (-13%)	**0.389±0.014 (0%)**
	$\mathcal{L}_{ADV} + \mathcal{L}_{MSE}$	**74.2±2.3 (0%)**	**81.4±0.8 (0%)**	**53.8±1.4 (0%)**	**57.9±2.3 (0%)**	0.401±0.0014 (-3.1%)

Ablation study	ADV Model	DVD → Electronics	Books → Kitchen	Kitchen → Electronics	DVD → Books
Weights in $\mathcal{L}_2$	$\mathcal{L}_{MSE}$	71.47 (-1.5%)	71.39 (-1.8%)	77.58 (-0.77%)	72.02 (-0.81%)
	$\mathcal{L}_{ADV} + \mathcal{L}_{MSE}$	**72.57 (0%)**	**72.69 (0%)**	**78.18 (0%)**	**72.61 (0%)**

Table 3.4. – Ablation with relative gain over Table 3.1; accuracy (↑) on `digits`, `amazon` and CE (↓) on `ads`

3.6.7   Discussion

Relationship between theoretical and experimental results   We comment on our experimental results in light of our adaptation Eq. (3.10) and imputation Eq. (3.11) upper-bounds. Let us first consider Eq. (3.10). The first term in Eq. (3.10), $\mathcal{E}_S(f \circ \widehat{g})$, is the classification loss $\mathcal{L}_3$ in Eq. (3.7). The second term in Eq. (3.10) $d_{\mathcal{F}\Delta\mathcal{F}}(p_S(\widehat{Z}), p_T(\widehat{Z}))$ is approximated by a proxy $\mathcal{L}_1$ Eq. (3.3) and accounts for alignment. $\mathcal{L}_1$ leads to substantial gains in Table 3.4 (first row) when added to the loss. The third term $\lambda_{\mathcal{H}_{\widehat{g}}}$ in Eq. (3.10) is the optimal joint error heuristically controlled with self-training as justified by upper-bound Eq. (3.12), with gains
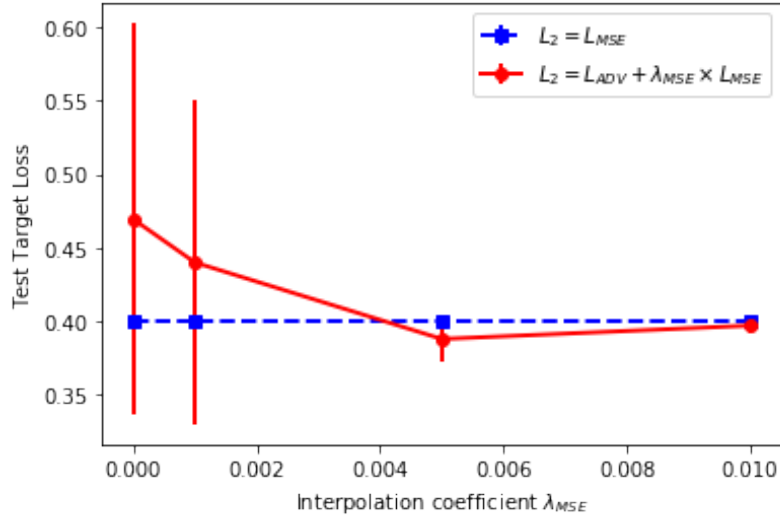
Figure 3.5. – *Adaptation-Imputation* target CE ($\downarrow$) with standard deviations on `ads-kaggle` w.r.t. λ_{MSE}

shown in Table 3.3. Second, we consider Eq. (3.11). It is the product of two terms, the target imputation error (I_T) and the error on the target $\mathcal{E}_T(f \circ \widehat{g})$ which is exactly the left hand side term in bound Eq. (3.10). (I_T) = (I_S) × (T_I), (I_S) is the source imputation error and is optimized when term $\mathcal{L}_2$ Eq. (3.4) is zero. (T_I) is the transfer error, optimized when $\mathcal{L}_1$ Eq. (3.3) is zero. Adding $\mathcal{L}_1$ to the loss improves the performance (Table 3.4). $\mathcal{L}_2$ Eq. (3.4) explains the gains of *Adaptation-Imputation* over *Adaptation-ZeroImputation* in Table 3.1 as *Adaptation-ZeroImputation* does not attempt to impute missing components. To summarize, minimizing our global error function L in Eq. (3.8) minimizes, according to the approximations just described, the two upper bounds in Eq. (3.10) and Eq. (3.11).

Limitations   Our results are obtained under some assumptions which we are the first to introduce to our knowledge for our problem. First, if the missing and the observed components are statistically independent, Assumption 2 is not valid, and then there is no way to impute this missing data. Second, if $p_S(Z_2|Z_1) \neq p_T(Z_2|Z_1)$ *i.e.* Assumption 3 is not valid, then we cannot transfer imputation from source to target. Yet, these assumptions are most often met in applications and allow to build a well-defined model with good empirical results.

3.7   Conclusion

We proposed a new model for UDA with non-stochastic target missingness with indirect supervision from a complete source. This method uses only labelled

source instances imputing the missing target values in a latent space. Under our assumptions, it minimizes an adaptation upper-bound of its target error and an imputation upper-bound of the ideal target error with full data and leads to important gains for two representative families of divergences (OT, ADV) on our benchmarks (digits, amazon) and on real-world advertising datasets, which are a complex task with missing features. We show that approaches using a pure regressive generator underperform compared to our approach on our real-world applications for which distributions are multi-modal. Finally, we introduced a heuristic refinement method based on self-training to deal with settings where posterior distributions mismatch. As follow-up, we plan to further investigate how to generate diverse outputs in our imputation network.

Chapter 4

Transferring representations under Generalized Target Shift

Chapter abstract

We consider the problem of UDA under conditional and label shift a.k.a. Generalized Target Shift (GeTarS). Unlike simpler UDA settings, few works have addressed this challenging problem. Recent approaches learn domain-invariant representations, yet they have practical limitations and rely on strong assumptions that may not hold in practice. In this paper, we explore a novel and general approach to align pretrained representations, which circumvents existing drawbacks. Instead of constraining representation invariance, it learns an OT map, implemented as a NN, which maps source representations onto target ones. Our approach is flexible and scalable, it preserves the problem's structure and it has strong theoretical guarantees under mild assumptions. In particular, our solution is unique, matches conditional distributions across domains, recovers target proportions and explicitly controls the target generalization risk. Through an exhaustive comparison on several datasets, we challenge the state-of-the-art in GeTarS.

The work in this chapter has led to the publication of a conference paper:

- M. Kirchmeyer, A. Rakotomamonjy, E. de Bézenac, and P. Gallinari (2022b). "Mapping conditional distributions for domain adaptation under generalized target shift". In: *International Conference on Learning Representations (ICLR).*

4.1   Introduction

UDA methods (Pan et al. 2010) train a classifier with labelled samples from a source domain S such that its risk on an unlabelled target domain T is low. This problem is ill-posed and simplifying assumptions were considered. Initial contri-

butions focused on three settings which decompose differently the joint distribution over input and label $X \times Y$: **CovS** (Shimodaira 2000) with $p_S(Y|X) = p_T(Y|X)$, $p_S(X) \neq p_T(X)$, **Target Shift (TarS)** (Zhang et al. 2013) with $p_S(Y) \neq p_T(Y)$, $p_S(X|Y) = p_T(X|Y)$ and **conditional shift** (Zhang et al. 2013) with $p_S(X|Y) \neq p_T(X|Y), p_S(Y) = p_T(Y)$. These assumptions are restrictive for real-world applications and were extended into **model shift** when $p_S(Y|X) \neq p_T(Y|X), p_S(X) \neq p_T(X)$ (Wang et al. 2014; Wang et al. 2015) and **GeTarS** (Zhang et al. 2013) when $p_S(X|Y) \neq p_T(X|Y), p_S(Y) \neq p_T(Y)$. We consider GeTarS where a key challenge is to map the source domain onto the target one to minimize both conditional and label shifts, without using target labels. The current SoTA in Gong et al. 2016; Combes et al. 2020; Rakotomamonjy et al. 2021; Shui et al. 2021 learns domain-invariant representations and uses estimated class-ratios between domains as importance weights in the training loss. However, this approach has several limitations. First, it updates representations through adversarial alignment which is prone to well-known instabilities, especially on applications where there is no established Deep Learning (DL) architectures *e.g.* CTR prediction, spam filtering etc. in contrast to vision. Second, to transfer representations, the domain-invariance constraint breaks the original problem structure and it was shown that this may degrade the discriminativity of target representations (Liu et al. 2019). Existing approaches that consider this issue (Xiao et al. 2019; Li et al. 2020a; Chen et al. 2019b) were not applied to GeTarS. Finally, generalization guarantees are derived under strong assumptions, detailed in Section 4.2.3, which may not hold in practice.

In this paper, we address these limitations with a new general approach, named Optimal Sample Transformation and Reweight (OSTAR), which maps pretrained representations using OT. OSTAR proposes an alternative to constraining representation invariance and performs jointly three operations: given a pretrained encoder, (i) it learns an OT map, implemented as a NN, between encoded source and target conditionals, (ii) it estimates target proportions for sample reweighting and (iii) it learns a classifier for the target domain using source labels. OSTAR has several benefits: (i) it is flexible, scalable and preserves target discriminativity and (ii) it provides strong theoretical guarantees under mild assumptions. In summary, our contributions are:

- We propose an approach, OSTAR, to align pretrained representations under GeTarS. Without constraining representation-invariance, OSTAR jointly learns a classifier for inference on the target domain and an OT map, which maps representations of source conditionals to those of target ones under class-reweighting. OSTAR preserves target discriminativity and experimentally challenges the state-of-the-art for GeTarS.

- OSTAR implements its OT map as a NN shared across classes. Our approach is thus flexible and has native regularization biases for stability. Moreover it is scalable and generalizes beyond training samples unlike standard linear programming based OT approaches.

- OSTAR has strong theoretical guarantees under mild assumptions: its solution is unique, recovers target proportions and correctly matches source and target conditionals at the optimum. It also explicitly controls the target risk with a new Wasserstein-based bound.

Our paper is organized as follows. In Section 4.2, we define our problem, approach and assumptions. In Section 4.3, we derive theoretical results. In Section 4.4, we describe our implementation. We report in Section 4.5 experimental results and ablation studies. In Section 4.6, we present related work.

4.2 Proposed approach

In this section, we successively define our problem, present our method, OSTAR and its main ideas and introduce our assumptions, used to provide theoretical guarantees for our method.

4.2.1 Problem definition

We refer to the UDA setting described in Sections 2.1.1 and 2.1.3.1. In short, we are given a labeled source domain S and an unlabeled target domain T with distributions p_S, respectively p_T. The corresponding datasets are denoted $\mathcal{D}_S, \mathcal{D}_T$ ($\mathcal{D}_T$ is unlabelled). For simplicity, given a domain $D \in \{S, T\}$, $\boldsymbol{p}_D^Y \in \mathbb{R}^K$ denotes the label marginal $p_D(Y)$. We consider a NN composed of an encoder g and classifier f. Given $D \in \{S, T\}$, $\mathcal{Z}_D$ denotes the encoded input domain via g.

In all generality, latent conditional distributions and label marginals differ across domains; this is the GeTarS assumption (Zhang et al. 2013) made in feature space $\mathcal{Z}$ rather than input space $\mathcal{X}$, as in Definition 4.1.

Definition 4.1 (GeTarS). GeTarS is defined by conditional mismatch across domains *i.e.* $\exists j \in \{1, \cdots, K\}, p_S(Z|Y = j) \neq p_T(Z|Y = j)$ and label shift *i.e.* $\boldsymbol{p}_S^Y \neq \boldsymbol{p}_T^Y$.

GeTarS is illustrated in Figure 4.1a and states that representations from a given class are different across domains with different label proportions. Operating in the latent space has several practical advantages *e.g.* improved discriminativity and dimension reduction. Our goal is to learn a classifier f in $\mathcal{Z}$ with low target risk, using source labels. This is challenging as (i) target labels are unknown

and (ii) there are two shifts to handle. We will show that this can be achieved with pretrained representations if we recover two key properties: (i) a map which matches source and target conditional distributions and (ii) target proportions to reweight samples by class-ratios and thus account for label shift. Our approach, OSTAR, achieves this objective.

4.2.2   Mapping conditional distributions under label shift

We now present OSTAR's components and the various training steps.

Components   The main components of OSTAR, illustrated in Figure 4.1b, are detailed below. These components are learned and estimated using the algorithm detailed in Section 4.4. They include:

- a fixed encoder $g : \mathcal{V} \to \mathcal{Z}$, defined in Section 4.2.1.

- a mapping $\phi : \mathcal{Z} \to \mathcal{Z}$, acting on source samples encoded by g.

- a label proportion vector $\boldsymbol{p}_N^Y$ on the simplex Δ_K.

- a classifier $f_N : \mathcal{Z} \to \{1, \ldots, K\}$ for the target domain in a hypothesis class $\mathcal{H}$ over $\mathcal{Z}$.

Objective g encodes source and target samples in a latent space such that it preserves rich information about the target task and such that the risk on the source domain is small. g is fixed throughout training to preserve target discriminativity. ϕ should map encoded source conditionals in $\mathcal{Z}_S$ onto corresponding encoded target ones in $\mathcal{Z}_T$ to account for conditional shift; $\mathcal{Z}_N$ denotes the mapped space. $\boldsymbol{p}_N^Y$ should estimate the target proportions $\boldsymbol{p}_T^Y$ to account for label shift. Components $(\phi, \boldsymbol{p}_N^Y)$ define a new labelled domain in latent space $N = (\mathcal{Z}_N, \mathcal{Y}_N, p_N(Z, Y))$ through a Sample Transformation And Reweight operation of the encoded S domain, as illustrated in Figure 4.1b. Indeed, the pushforward[1] by ϕ of encoded source conditionals defines conditionals in domain N, $p_N^\phi(Z|Y)$:

$$\forall k, \ p_N^\phi(Z|Y = k) \triangleq \phi_\# \Big(p_S(Z|Y = k) \Big)$$

Then, $\boldsymbol{p}_N^Y$ weights each conditional in N. This yields a marginal distribution in N,

$$p_N^\phi(Z) \triangleq \sum_{k=1}^{K} \boldsymbol{p}_N^{Y=k} p_N^\phi(Z|Y = k) \tag{4.1}$$

---

1. $f_\# \rho$ is the push-forward measure $f_\# \rho(B) = \rho(f^{-1}(B))$, for all measurable set B.

(a) UDA under GeTarS in latent space   (b) Mapping & reweighting with OSTAR
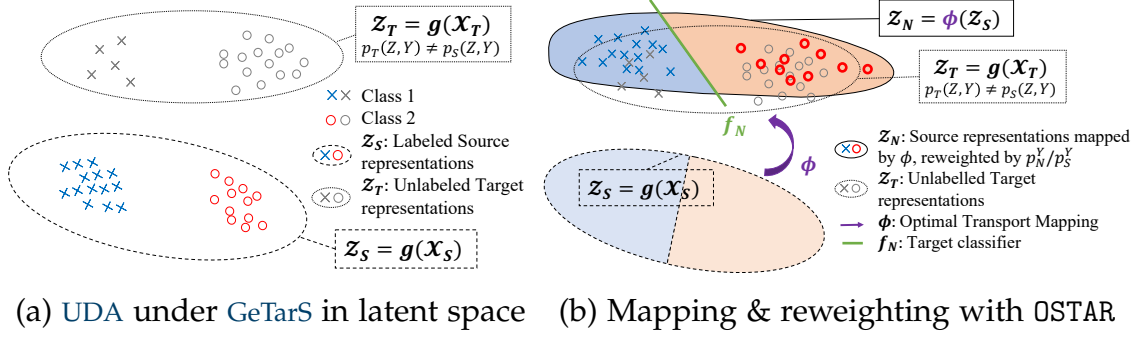
Figure 4.1. – **Illustration of OSTAR on a 2-class UDA problem**. (a) A pretrained
encoder g defines a latent space $\mathcal{Z}$ with labelled source samples ×∘
and unlabelled target samples ×∘ under conditional and label shift
(GeTarS). (b) We train a target classifier f_N on a new domain N, where
labelled samples ×∘ are obtained by (i) mapping source samples
with ϕ acting on conditional distributions and (ii) reweighting these
samples by estimated class-ratios $\boldsymbol{p}_N^Y/\boldsymbol{p}_S^Y$. ϕ should match source and
target conditionals and $\boldsymbol{p}_N^Y$ should estimate target proportions $\boldsymbol{p}_T^Y$.

Finally, classifier f_N is trained on labelled samples from domain N. This is possible
as each sample in N is a projection of a labelled sample from S. f_N can then be
used for inference on T. We will show that it has low target risk when components
ϕ and $\boldsymbol{p}_N^Y$ achieve their objectives detailed above.

Training   We train OSTAR's components in two stages. First, we train g along a
source classifier f_S from scratch by minimizing source classification loss; alterna-
tively, g can be tailored to specific problems with pretraining. Second, we jointly
learn $(f_N, \phi, \boldsymbol{p}_N^Y)$ to minimize a classification loss in domain N and to match tar-
get conditionals and proportions with those in domain N. As target conditionals
and proportions are unknown, we propose a proxy problem for $(\phi, \boldsymbol{p}_N^Y)$ to match
instead latent marginals $p_T(Z)$ and $p_N^\phi(Z)$ Eq. (4.1). We solve this proxy problem
under least action principle measured by a Monge transport cost (Santambrogio
2015), denoted $\mathcal{C}(\phi)$, as in Eq. (OT):

$$\min_{\phi, \boldsymbol{p}_N^Y \in \Delta_K} \mathcal{C}(\phi) \triangleq \sum_{k=1}^K \int_{\mathbf{z} \in \mathcal{Z}} \|\phi(\mathbf{z}) - \mathbf{z}\|_2^2 \, p_S(\mathbf{z}|Y=k)d\mathbf{z} \tag{OT}$$
$$\text{subject to } p_N^\phi(Z) = p_T(Z)$$

For any function ϕ *e.g.* a NN, $\mathcal{C}(\phi)$ is the transport cost of encoded source con-
ditionals by ϕ. It uses a cost function, $c(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^p$, where without loss of
generality $p = 2$. The optimal $\mathcal{C}(\phi)$ is the sum of Wasserstein-2 distances between
source conditionals and their mappings. Eq. (OT) seeks to minimize $\mathcal{C}(\phi)$ under
marginal matching. We provide some further background on optimal transport in

Appendix B.3 and discuss there the differences between our OT problem Eq. (OT) and the standard Monge OT problem between $p_S(Z)$ and $p_T(Z)$.

Our OT formulation is key to the approach. First, it is at the basis of our theoretical analysis. Under Assumption 6 later defined, the optimal transport cost is the sum of the Wasserstein-2 distance between source and matched target conditionals. We can then provide conditions on these distances in Assumption 7 to formally define when the solution to Eq. (OT) correctly matches conditionals. Second, it allows us to learn the OT map with a NN. This NN approach: (i) generalizes beyond training samples and scales up with the number of samples unlike linear programming based OT approaches (Courty et al. 2017b) and (ii) introduces useful stability biases which make learning less prone to numerical instabilities as highlighted in Bézenac et al. 2021; Karkar et al. 2020.

4.2.3   Assumptions

In Section 4.3 we will introduce the theoretical properties of our method which offers several guarantees. As always, this requires some assumptions which are introduced below. We discuss their relations with assumptions used in related work and detail why they are less restrictive. We provide some additional discussion on their motivation and validity in Appendix B.4.

Assumption 5 (Cluster Assumption on S). $\forall k, \boldsymbol{p}_S^{Y=k} > 0$ and there is a partition of the source domain $\mathcal{Z}_S$ such that $\mathcal{Z}_S = \cup_{k=1}^K \mathcal{Z}_S^{(k)}$ and $\forall k \; p_S(Z \in \mathcal{Z}_S^{(k)} | Y = k) = 1$.

Assumption 5, inspired from Chapelle et al. 2010, states that source representations with the same label are within the same cluster. It helps guarantee that only one map is required to match source and target conditionals. Assumption 5 is for instance satisfied when the classification loss on the source domain is zero which corresponds to the training criterion of our encoder g. Interestingly other approaches *e.g.* Combes et al. 2020; Rakotomamonjy et al. 2021 assume that it also holds for target representations; this is harder to induce as target labels are unknown.

Assumption 6 (Conditional matching). A mapping ϕ solution to our matching problem in Eq. (OT) maps a source conditional to a target one *i.e.* $\forall k \; \exists j \; \phi_\#(p_S(Z|Y = k)) = p_T(Z|Y = j)$.

Assumption 6 guarantees that mass of a source conditional will be entirely transferred to a target conditional; ϕ solution to Eq. (OT) will then perform optimal assignment between conditionals. It is less restrictive than alternatives: the *ACons* assumption in Zhang et al. 2013; Gong et al. 2016 states the existence of a map matching the right conditional pairs *i.e.* $j = k$ in Assumption 6, while the *GLS*

assumption of Combes et al. 2020 imposes that $p_S(Z|Y) = p_T(Z|Y)$. *GLS* is thus included in Assumption 6 when $j = k$ and $\phi = \text{Id}$ and is more restrictive.

Assumption 7 (Cyclical monotonicity between S and T). For all K elements permutation σ, conditional probabilities in the source and target domains satisfy $\sum_{k=1}^{K} \mathcal{W}_2(p_S(Z|Y = k), p_T(Z|Y = k)) \leq \sum_{k=1}^{K} \mathcal{W}_2(p_S(Z|Y = k), p_T(Z|Y = \sigma(k)))$ with $\mathcal{W}_2$, the Wasserstein-2 distance.

Assumption 7, introduced in Rakotomamonjy et al. 2021, formally defines settings where conditionals are guaranteed to be correctly matched with an optimal assignment in the latent space under Assumption 6. One sufficient condition for yielding this assumption is when $\forall k, j, \ \mathcal{W}_2(p_S(Z|Y = k), p_T(Z|Y = k)) \leq \mathcal{W}_2(p_S(Z|Y = k), p_T(Z|Y = j))$. This last condition is typically achieved when conditionals between source and target of the same class are "sufficiently near" to each other.

Assumption 8 (Conditional linear independence on T). $\{p_T(Z|Y = k)\}_{k=1}^{K}$ are linearly independent implying $\forall k, \nexists \alpha \in \left\{ \Delta_K | \ p_T(Z|Y = k) = \sum_{j=1, j \neq k}^{K} \alpha_j p_T(Z|Y = j) \right\}$.

Assumption 8 is standard and seen in TarS to guarantee correctly estimating target proportions (Redko et al. 2019; Garg et al. 2020). It discards pathological cases like when $\exists(i, j, k) \exists(a, b) \ s.t. \ a + b = 1, p_T(Z|Y = i) = a \times p_T(Z|Y = j) + b \times p_T(Z|Y = k)$. It is milder than its alternative *A2Cons* in Zhang et al. 2013; Gong et al. 2016 which states linear independence of linear combinations of source and target conditionals, in $\mathcal{X}$ respectively $\mathcal{Z}$.

4.3 Theoretical results

We present our theoretical results, with proofs in Appendix B.5, for OSTAR under our mild assumptions in Section 4.2.3. We first analyze in Section 4.3.1 the properties of the solution to our problem in Eq. (OT). Then, we show in Section 4.3.2 that given the learned components g, ϕ and $\boldsymbol{p}_N^Y$, the target generalization error of a classifier in the hypothesis space $\mathcal{H}$ can be upper-bounded by different terms including its risk on domain N and the Wasserstein-1 distance between marginals in N and T.

4.3.1 Properties of the solution to the OT alignment problem

Proposition 4.2 (Unicity and match). *For any encoder g which defines $\mathcal{Z}$ satisfying Assumption 5, 6, 7, 8, there is a unique solution $(\phi, \boldsymbol{p}_N^Y)$ to Eq. (OT) and $\phi_\#(p_S(Z|Y)) = p_T(Z|Y)$ and $\boldsymbol{p}_N^Y = \boldsymbol{p}_T^Y$.*

Given an encoder g satisfying our assumptions, Proposition 4.2 shows two strong results. First, the solution to Eq. (OT) exists and is unique. Second, we prove that this solution defines a domain N, via the sample transformation and reweight operation defined in Eq. (4.1), where encoded conditionals and label proportions are equal to target ones. For comparison, Combes et al. 2020; Shui et al. 2021 recover target proportions only under *GLS i.e.* when conditionals are already matched, while we match both conditional and label proportions under the more general GeTarS.

4.3.2   Controlled target generalization risk

We now characterize the generalization properties of a classifier f_N trained on this domain N with a new general upper-bound. First, we introduce some notations; given an encoder g onto a latent space $\mathcal{Z}$, we define the risk of a classifier f as $\mathcal{E}_D^g(f) \triangleq \mathbb{E}_{\mathbf{z} \sim p_D(Z,Y)}[f(\mathbf{z}) \neq y]$ with $D \in \{S, T, N\}$.

Theorem 4.3 (Target risk upper-bound). *Given a fixed encoder g defining a latent space $\mathcal{Z}$, two domains N and T satisfying cyclical monotonicity in $\mathcal{Z}$, assuming that we have $\forall k, \boldsymbol{p}_N^{Y=k} > 0$, then $\forall f_N \in \mathcal{H}$ where $\mathcal{H}$ is a set of M-Lipschitz continuous functions over $\mathcal{Z}$, we have*

$$\mathcal{E}_T^g(f_N) \leq \underbrace{\mathcal{E}_N^g(f_N)}_{\text{Classification } (C)} + \frac{2M}{\min_{k=1}^K \boldsymbol{p}_N^{Y=k}} \underbrace{\mathcal{W}_1\Big(p_N(Z), p_T(Z)\Big)}_{\text{Alignment } (A)} +$$

$$2M(1 + \frac{1}{\min_{k=1}^K \boldsymbol{p}_N^{Y=k}}) \underbrace{\mathcal{W}_1\Big(\sum_{k=1}^K \boldsymbol{p}_T^{Y=k} p_T(Z|Y=k), \sum_{k=1}^K \boldsymbol{p}_N^{Y=k} p_T(Z|Y=k)\Big)}_{\text{Label } (L)} \quad (4.2)$$

We first analyze our upper-bound in Eq. (4.2). The target risk of f_N is controlled by three main terms: the first (C) is the risk of f_N on domain N, the second (A) is the Wasserstein-1 distance between latent marginals of domain N and T, the third term (L) measures a divergence between label distributions using, as a proxy, two ad-hoc marginal distributions. There are two other terms in Eq. (4.2): first, a Lipschitz-constant M that can be made small by implementing f_N as a NN with piece-wise linear activations and regularized weights; second the minimum of $\boldsymbol{p}_N^Y$, which says that generalization is harder when a target class is less represented. We learn OSTAR's components to minimize the r.h.s of Eq. (4.2). Terms (C) and (A) can be explicitly minimized, respectively by training a classifier f_N on domain N and by learning $(\phi, \boldsymbol{p}_N^Y)$ to match marginals of domains N and T. Term (L) is not computable, yet its minimization is naturally handled by OSTAR. Indeed, term (L) is minimal when $\boldsymbol{p}_N^Y = \boldsymbol{p}_T^Y$ under Assumption 8 per Redko et al. 2019. With

OSTAR, this sufficient condition is guaranteed in Proposition 4.2 by the solution to Eq. (OT) under our mild assumptions in Section 4.2.3. This solution defines a domain N for which $\boldsymbol{p}_N^Y = \boldsymbol{p}_T^Y$ and term (A) equals zero.

We now detail the originality of this result over existing Wasserstein-based generalization bounds. First, our upper-bound is general and can be explicitly minimized even when target labels are unknown unlike Shui et al. 2021 or Combes et al. 2020 which require knowledge of $\boldsymbol{p}_T^Y$ or its perfect estimation. Combes et al. 2020 claims that correct estimation of $\boldsymbol{p}_T^Y$ *i.e.* (L) close to zero, is achieved under *GLS i.e.* $p_S(Z|Y) = p_T(Z|Y)$. However, *GLS* is hardly guaranteed when the latent space is learned: a sufficient condition in Combes et al. 2020 requires knowing $\boldsymbol{p}_T^Y$, which is unrealistic in UDA. Second, our bound is simpler than the one in Rakotomamonjy et al. 2021: in particular, it removes several redundant terms which are unmeasurable due to unknown target labels.

4.4 Implementation

Our solution, detailed below, minimizes the generalization bound in Eq. (4.2). It implements f_N, g with NNs and ϕ with a residual NN. Our solution jointly solves (i) a classification problem to account for term (C) in Eq. (4.2) and (ii) an alignment problem on pretrained representations Eq. (OT) to account for terms (A) and (L) in Eq. (4.2). Our pseudo-code and runtime / complexity analysis are presented in Appendix B.6.

Encoder initialization   Prior to learning $\phi, \boldsymbol{p}_N^Y, f_N$, we first learn the encoder g jointly with a source classifier f_S to yield a zero source classification loss via Eq. (4.3). With ℓ_{ce} the CE loss,

$$\min_{f_S, g} \mathcal{L}_c^g(f_S, \mathcal{D}_S) \triangleq \min_{f_S, g} \frac{1}{n} \sum_{i=1}^n \ell_{ce}(f_S \circ g(\mathbf{x_S^{(i)}}), y_S^{(i)}) \tag{4.3}$$

g is then fixed to preserve the original problem structure. This initialization step helps enforce Assumption 5 and could alternatively be replaced by directly using a pretrained encoder if available.

Joint alignment and classification   We then solve alternatively (i) a classification problem on domain N w.r.t. f_N to account for term (C) in Eq. (4.2) and (ii) the Eq. (OT) w.r.t. $(\phi, \boldsymbol{p}_N^Y)$ to account for term (A). Term (L) in Eq. (4.2) is handled by matching $\boldsymbol{p}_N^Y$ and $\boldsymbol{p}_T^Y$ through the minimization of Eq. (OT). $\boldsymbol{p}_N^Y$ is estimated with the confusion-based approach in Lipton et al. 2018, while ϕ minimizes the

Lagrangian relaxation, with hyperparameter λ_{OT}, of the constrained optimization problem Eq. (OT). The equality constraint in Eq. (OT) is measured through a Wasserstein distance as our bound in Eq. (4.2) is tailored to this distance, however, we are not restricted by this choice as other discrepancy measures are also possible. The objective based on Eq. (4.2) corresponds to the optimization problem:

$$\min_{\phi, f_N} \mathcal{L}_c^g(f_N, \mathcal{D}_N) + \lambda_{OT} \ \mathcal{L}_{OT}^g(\phi) + \mathcal{L}_{wd}^g(\phi, \boldsymbol{p}_N^Y)$$

$$\text{subject to } \boldsymbol{p}_N^Y = \operatorname*{arg\,min}_{\mathbf{p} \geq 0, \mathbf{p} \in \Delta_K} \frac{1}{2} \| \hat{\boldsymbol{p}}_T^Y - \hat{\mathbf{C}} \frac{\mathbf{p}}{\boldsymbol{p}_S^Y} \|_2^2 \qquad \text{[Label proportion estimation]} \tag{CAL}$$

$$\text{where } \mathcal{L}_c^g(f_N, \mathcal{D}_N) \triangleq \frac{1}{n} \sum_{i=1}^n \frac{\boldsymbol{p}_N^{y_S^{(i)}}}{\boldsymbol{p}_S^{y_S^{(i)}}} \ell_{ce}(f_N \circ \phi \circ g(\mathbf{x_S^{(i)}}), y_S^{(i)}) \qquad \text{[Classification loss in } N\text{]} \tag{4.4}$$

$$\text{and } \mathcal{L}_{OT}^g(\phi) \triangleq \sum_{k=1}^K \frac{1}{\#\{y_S^{(i)} = k\}_{i \in [\![1,n]\!]}} \sum_{y_S^{(i)} = k, i \in [\![1,n]\!]} \| \phi(\mathbf{z_S^{(i)}}) - \mathbf{z_S^{(i)}} \|_2^2 \quad \text{[Objective of Eq. (OT)]}$$

$$\tag{4.5}$$

$$\text{and } \mathcal{L}_{wd}^g(\phi, \boldsymbol{p}_N^Y) \triangleq \sup_{\|v\|_L \leq 1} \frac{1}{n} \sum_{i=1}^n \frac{\boldsymbol{p}_N^{y_S^{(i)}}}{\boldsymbol{p}_S^{y_S^{(i)}}} v \circ \phi(\mathbf{z_S^{(i)}}) - \frac{1}{m} \sum_{j=1}^m v(\mathbf{z_T^{(j)}}) \quad \text{[Relaxation of Eq. (OT)]} \tag{4.6}$$

$\hat{\mathbf{C}}$ is the confusion matrix of f_N on domain N, $\hat{\boldsymbol{p}}_T^Y \in \Delta_K$ is the target label proportions estimated with f_N. $\mathcal{L}_c^g(f_N, N)$ Eq. (4.4) is the classification loss of f_N on N, derived in Appendix B.5, which minimizes term (C) in Eq. (4.2). Note that samples in domain N are obtained by mapping source samples with ϕ; they are reweighted to account for label shift. $\mathcal{L}_{OT}^g$ Eq. (4.5) defines the transport cost of ϕ on S. Implementing ϕ with a ResNet performed better than standard MLPs, thus we minimize in practice the dynamical transport cost, better tailored to residual maps and used in Bézenac et al. 2021; Karkar et al. 2020. $\mathcal{L}_{wd}^g$ Eq. (4.6) is the empirical form of the dual Wasserstein-1 distance between $p_N^\phi(Z)$ and $p_T(Z)$ and seeks at enforcing the equality constraint in Eq. (OT) *i.e.* minimizing term (A). OSTAR's assumptions also guarantee that term (L) is small at the optimum.

Improve target discriminativity   OSTAR solves a transfer problem with pre-trained representations, but cannot change their discriminativity in the target. This may harm performance when the encoder g is not well suited for the target. Domain-invariant methods are less prone to this issue as they update target representations. In our upper-bound in Eq. (4.2), target discriminativity is assessed by the value of term (C) at optimum of the alignment problem Eq. (OT). This value depends on g and may be high since g has been trained with only source labels. Nevertheless, it can be reduced by updating g using target outputs such that class separability for target representations is better enforced. To achieve this

goal, we propose an extension of Eq. (CAL) using Information Maximization (IM), not considered in existing domain-invariant GeTarS methods. IM was originally introduced for source-free adaptation without alignment (Liang et al. 2020). In this context, target predictions are prone to errors and there is no principled way to mitigate this problem. In our case, we have access to source samples and OSTAR minimizes an upper-bound to its target error, which avoids performance degradation. IM refines the decision boundaries of f_N with two terms on target samples. $\mathcal{L}_{ent}^{g}(f_N, T)$ Eq. (4.7) is the conditional entropy of f_N which favors low-density separation between classes. Denoting $\delta_k(\cdot)$ the k-th component of the softmax,

$$\mathcal{L}_{ent}^{g}(f_N, \mathcal{D}_T) = \sum_{i=1}^{m} \sum_{k=1}^{K} \delta_k(f_N \circ g(\mathbf{x_T^{(i)}})) \log(\delta_k(f_N \circ g(\mathbf{x_T^{(i)}}))) \tag{4.7}$$

$\mathcal{L}_{div}^{g}(f_N, \mathcal{D}_T)$ Eq. (4.8) promotes diversity by regularizing the average output of $f_N \circ g$ to be uniform on T. It avoids predictions from collapsing to the same class thus softens the effect of conditional entropy.

$$\mathcal{L}_{div}^{g}(f_N, \mathcal{D}_T) = \sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k = D_{KL}(\hat{p}, \frac{1}{K}\mathbf{1}_K) - \log K; \hat{p} = \mathbb{E}_{\mathbf{x_T} \in \mathcal{X}_T}[\delta(f_N \circ g(\mathbf{x_T}))]$$
$$\tag{4.8}$$

Our variant introduces two additional steps in the learning process. First, the latent space is fixed and we optimize f_N with IM in Eq. (SS). Then, we optimize the representations in Eq. (SSg) while avoiding modifying source representations by including $\mathcal{L}_c^{g}(f_S, \mathcal{D}_S)$ Eq. (4.3) with a fixed source classifier f_S.

$$\min_{f_N} \mathcal{L}_c^{g}(f_N, \mathcal{D}_N) + \mathcal{L}_{ent}^{g}(f_N, \mathcal{D}_T) + \mathcal{L}_{div}^{g}(f_N, \mathcal{D}_T) \tag{SS}$$

$$\min_{f_N, g} \mathcal{L}_c^{g}(f_N, \mathcal{D}_N) + \mathcal{L}_{ent}^{g}(f_N, \mathcal{D}_T) + \mathcal{L}_{div}^{g}(f_N, \mathcal{D}_T) + \mathcal{L}_c^{g}(f_S, \mathcal{D}_S) \tag{SSg}$$

4.5 Experimental results

We now present our experimental results on several UDA problems under GeTarS and show that OSTAR outperforms recent SOTA baselines. The GeTarS assumption is particularly relevant on our datasets as encoded conditional distributions do not initially match as seen in Appendix Figure B.2.

Setting   We consider: (i) an academic benchmark Digits with two adaptation problems between USPS (Hull 1994) and MNIST (LeCun et al. 1998), (ii) a synthetic to real images adaptation benchmark VisDA12 (Peng et al. 2017) and (iii) two object categorizations problems Office31 (Saenko et al. 2010), OfficeHome

(Venkateswara et al. 2017a) with respectively six and twelve adaptation problems. The original datasets have fairly balanced classes, thus source and target label distributions are similar. This is why we subsample our datasets to make label proportions dissimilar across domains as detailed in Appendix Table B.6. For `Digits`, we subsample the target domain and investigate three settings - balanced, mild and high as Rakotomamonjy et al. 2021. For other datasets, we modify the source domain by considering 30% of the samples coming from the first half of their classes as Combes et al. 2020. We report a `Source` model, trained using only source samples without adaptation, and various UDA methods: two `CovS` methods and three recent SOTA GeTarS models. Chosen UDA methods learn invariant representations with reweighting in GeTarS models or without reweighting in other baselines. The two `CovS` baselines are `DANN` (Ganin et al. 2016a) which approaches $\mathcal{H}$-divergence and `WD`$_{\beta=0}$ (Shen et al. 2018) which computes Wasserstein distance. The GeTarS baselines are Wu et al. 2019; Rakotomamonjy et al. 2021; Combes et al. 2020. We use Wasserstein distance to learn invariant representations such that only the strategy to account for label shift differs. Wu et al. 2019, denoted `WD`$_\beta$, performs assymetric alignment with parameter β, for which we test different values ($\beta \in \{1, 2\}$). `MARSc`, `MARSg` (Rakotomamonjy et al. 2021) and `IW-WD` (Combes et al. 2020) estimate target proportions respectively with optimal assignment or with the estimator in Lipton et al. 2018 also used in `OSTAR`. We report `DI-Oracle`, an oracle which learns invariant representations with Wasserstein distance and makes use of true class-ratios. Finally, we report `OSTAR` with and without IM. All baselines are reimplemented for a fair comparison with the same NN architectures detailed in Appendix B.7.

Results   In Table 4.1, we report, over 10 runs, mean and standard deviations for balanced accuracy, *i.e.* average recall on each class. This metric is suited for imbalanced problems (Brodersen et al. 2010). For better visibility, results are aggregated over all imbalance settings of a dataset (line "subsampled"). In the Appendix, full results are reported in Table B.1 along the ℓ_1 target proportion estimation error for GeTarS baselines and `OSTAR+IM` in Figure B.3. First, we note that low estimation error of p_T^Y is correlated to high accuracy for all models and that `DI-Oracle` upper-bounds domain-invariant approaches. This shows the importance of correctly estimating p_T^Y. Second, we note that `OSTAR` improves `Source` (column 2) and is competitive w.r.t. `IW-WD` (column 9) on `VisDA`, `Office31` and `OfficeHome` and `MARSg` (column 7) on `Digits` although it keeps target representations fixed unlike these two methods. `OSTAR+IM` (column 11) is less prone to initial target discriminativity and clearly outperforms or equals the baselines on both balanced accuracy and proportion estimation. It even improves `DI-Oracle` (column 12) for balanced accuracy despite not using true class-ratios. This (i) shows the benefits of not constraining domain-invariance which may degrade target discriminativ-

Table 4.1. – Balanced accuracy (↑) over 10 runs. The best performing model is indicated in **bold**. Results are aggregated over all imbalance scenarios and adaptation problems within a same dataset.

Setting	Source	DANN	$WD_{\beta=0}$	$WD_{\beta=1}$	$WD_{\beta=2}$	MARSg	MARSc	IW-WD	OSTAR	OSTAR+IM	DI-Oracle
					Digits						
balanced	74.98 ± 3.8	90.81 ± 1.3	92.63 ± 1.0	82.80 ± 4.7	76.07 ± 7.1	92.18 ± 2.2	94.91 ± 1.4	95.89 ± 0.5	91.66 ± 0.9	$\mathbf{97.51 \pm 0.3}$	96.90 ± 0.2
subsampled	75.05 ± 3.1	89.91 ± 1.5	89.45 ± 1.0	81.56 ± 4.8	77.77 ± 6.5	91.87 ± 2.0	93.75 ± 1.4	93.22 ± 1.1	88.39 ± 1.5	$\mathbf{96.69 \pm 0.7}$	96.43 ± 0.3
					VisDA12						
original	48.63 ± 1.0	53.72 ± 0.9	57.40 ± 1.1	47.56 ± 0.8	36.21 ± 1.8	55.62 ± 1.6	55.33 ± 0.8	51.88 ± 1.6	50.37 ± 0.6	$\mathbf{59.24 \pm 0.5}$	57.61 ± 0.3
subsampled	42.46 ± 1.4	47.57 ± 0.9	47.32 ± 1.4	41.48 ± 1.6	31.83 ± 3.0	55.00 ± 1.9	51.86 ± 2.0	50.65 ± 1.5	49.05 ± 0.9	$\mathbf{58.84 \pm 1.0}$	55.77 ± 1.1
					Office31						
subsampled	74.50 ± 0.5	76.13 ± 0.3	76.24 ± 0.3	74.23 ± 0.5	72.40 ± 1.8	80.20 ± 0.4	80.00 ± 0.5	77.28 ± 0.4	76.19 ± 0.8	$\mathbf{82.61 \pm 0.4}$	81.07 ± 0.3
					OfficeHome						
subsampled	50.56 ± 2.8	50.87 ± 1.05	53.47 ± 0.7	52.24 ± 1.1	49.48 ± 1.3	56.60 ± 0.4	56.22 ± 0.6	54.87 ± 0.4	54.64 ± 0.7	$\mathbf{59.51 \pm 0.4}$	57.97 ± 0.3

ity especially under label estimation errors; (ii) validates our theoretical results which show that OSTAR controls the target risk and recovers target proportions. We visualize how OSTAR aligns S and T representations in Appendix Figure B.2.

Ablation studies   We perform two studies. We first measure the **role of IM in our model**. In Appendix Table B.2, we show the contribution of Eq. (SS) (column 3) and Eq. (SSg) (column 4) to Eq. (CAL) (column 2). We also evaluate the effect of IM on our baselines in Appendix Table B.3, even if this is not part of their original work. IM improves performance on VisDA and Office and degrades it on Digits. The performance remains below the ones of OSTAR+IM. Finally, we evaluate the impact of IM on our upper-bound in Eq. (4.2) in Appendix Table B.4. We assume that target conditionals are known to compute term (L). We observe that term (C), related to target discriminativity, and the alignment terms (A) and (L) are reduced. This explains the improvements due to IM and shows empirically that IM helps better optimize the three functions ϕ, g, f_N in our upper-bound. The second study in Table 4.2 measures the **effect of the OT transport cost** in our objective Eq. (CAL). Proposition 4.2 showed that OT guarantees recovering target proportions and matching conditionals. We consider MNIST→USPS under various shifts (Line 1) and initialization gains *i.e.* the standard deviation of the weights of the NN (Line 2). On line 1, we note that $\lambda_{OT} \neq 0$ in problem Eq. (CAL) improves balanced accuracy (left) and ℓ_1 estimation error (middle) over $\lambda_{OT} = 0$ over all shifts, especially high ones. This improvement is correlated with lower mean and standard deviation of the normalized transport cost per sample (right). We observe the same trends when changing initialization gains in Line 2. This confirms our theoretical results and shows the advantages of OT regularization biases for performance and stability. In Appendix Table B.5, we test additional

Table 4.2. – OT and balanced accuracy ↑ (left), ℓ_1 estimation error ↓ (middle) and normalized transport cost $\mathcal{L}_{OT}^g/n$ ↓ (right) for MNIST→USPS. The best model for accuracy is in **bold** with a "⋆". We vary shifts (Line 1) and initialization gains (standard deviation of NN's weights) (Line 2)

MNIST→USPS - initialization gain 0.02		
Shift	$\lambda_{OT} = 0$	$\lambda_{OT} = 10^{-2}$
balanced	**94.92 ± 0.6**	**95.12 ± 0.6**
mild	88.28 ± 1.5	**91.77 ± 1.2**
high	85.24 ± 1.6	**88.55 ± 1.1**

MNIST→USPS - high imbalance		
Init Gain	$\lambda_{OT} = 0$	$\lambda_{OT} = 10^{-2}$
0.02	85.24 ± 1.6	**88.55 ± 1.1**
0.1	84.62 ± 2.3	**88.41 ± 1.3**
0.3	83.11 ± 2.4	**89.41 ± 1.6**
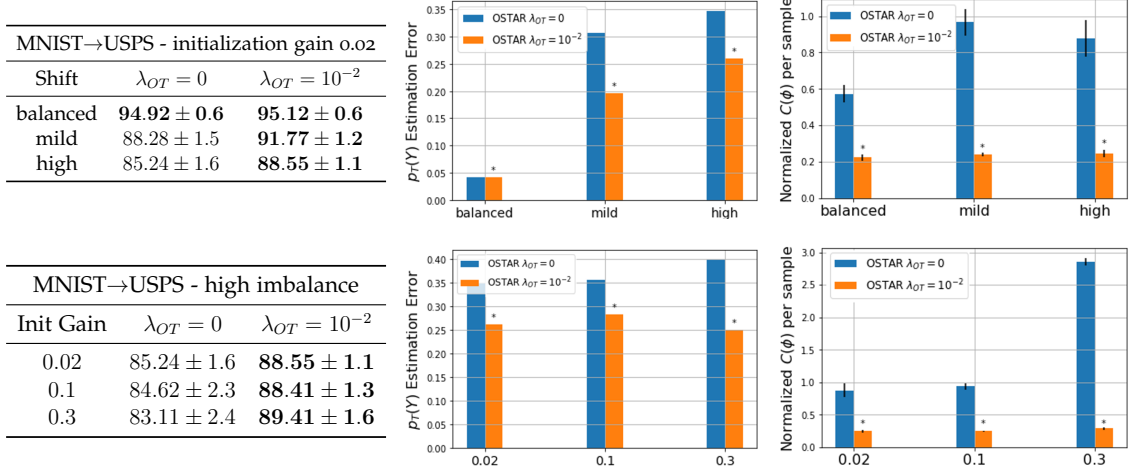
values of λ_{OT} and see that `OSTAR` recovers the `Source` model under high λ_{OT}. Indeed, the latter constrains $\phi \approx \mathrm{Id}$.

4.6   Related Work

Handling label shift in UDA   All existing UDA approaches reviewed in Section 2.1.3.1 are well-defined under CovS, yet fail under label shift (TarS and GeTarS). Handling label shift requires estimating $\boldsymbol{p}_T^Y$ to reweight source samples by estimated class-ratios (Zhao et al. 2019). An alternative is to use a fixed weight (Wu et al. 2019). When conditionals are unchanged *i.e.* TarS, $\boldsymbol{p}_T^Y$ can be recovered without needs for alignment (Lipton et al. 2018; Redko et al. 2019). In Redko et al. 2019, target proportions are estimated through minimization of a reweighted Wasserstein distance between marginals. Under GeTarS, there is the additional difficulty of matching conditionals. The SOTA methods for GeTarS are domain-invariant with sample reweighting (Rakotomamonjy et al. 2021; Combes et al. 2020; Gong et al. 2016; Shui et al. 2021). Rakotomamonjy et al. 2021 formulates two separate OT problems for class-ratio estimation and conditional alignment and Shui et al. 2021 is the OT extension of Combes et al. 2020 to multi-source UDA. An earlier mapping-based approach was proposed in Zhang et al. 2013 to align conditionals, also under reweighting. Estimators used in GeTarS are confusion-based in Combes et al. 2020; Shui et al. 2021; derived from optimal assignment in Rakotomamonjy et al. 2021 or from the minimization of a reweighted Maximum Mean Discrepancy (MMD) between marginals in Zhang et al. 2013; Gong et al. 2016.

`OSTAR` **and GeTarS methods**   `OSTAR` is a new mapping-based approach for GeTarS, that does not learn invariant representations. As stated earlier, the invariance constraint may degrade target discriminativity (Liu et al. 2019; Chen et al. 2019b), which is not the case for `OSTAR`. It has several practical (flexibility, scalability, stability) and theoretical improvements over prior domain-invariant GeTarS approaches (Rakotomamonjy et al. 2021; Combes et al. 2020; Gong et al. 2016; Shui et al. 2021) as detailed in this paper. The existing mapping-based approach for GeTarS, Zhang et al. 2013, is not flexible, operates on high dimensional input spaces and does not scale up to large label spaces as it considers a linear map for each pair of conditionals. `OSTAR` solves these limitations.

`OSTAR` **and OT UDA methods**   `OSTAR` is the first OT mapping-based approach for GeTarS. Compared to prior OT methods (*cf.* Section 2.1.3.1) it has several benefits: (i) representation are kept separate, thus target discriminativity is not deteriorated, (ii) a single OT problem is solved unlike Rakotomamonjy et al. 2021, (iii) the OT map is implemented with a NN which generalizes beyond training samples and scales up with the number of samples unlike barycentric OT maps; (iv) it improves efficiency of matching by encoding samples, thereby improving discriminativity and reducing dimensionality, unlike Courty et al. 2017b.

4.7 Conclusion

We introduced `OSTAR`, a new general approach to align pretrained representations under GeTarS, which does not constrain representation invariance. `OSTAR` learns a flexible and scalable map between conditional distributions in the latent space. This map, implemented as a ResNet, solves an OT matching problem with native regularization biases. Our approach provides strong generalization guarantees under mild assumptions as it explicitly minimizes a new upper-bound to the target risk. Experimentally, it challenges recent invariant GeTarS methods on several UDA benchmarks.

Part III

DOMAIN GENERALIZATION IN
CLASSIFICATION

In Chapter 5, we consider the problem of Domain Generalization (DG) for classification problems, where we do not observe data from the target domain at train or test time. This is a more complex setting than in Part ii. We propose to use the links between ensembling and Weight Averaging (WA) to learn a Neural Network (NN) that generalizes Out-of-distribution (OOD).

The main contributions are outlined in Section 1.3.2.

Chapter 5

Diverse Weight Averaging for Domain Generalization

Chapter abstract

Standard NNs struggle to generalize under distribution shifts in computer vision. Fortunately, combining multiple networks can consistently improve out-of-distribution generalization. In particular, WA strategies were shown to perform best on the competitive DomainBed benchmark; they directly average the weights of multiple networks despite their nonlinearities. In this paper, we propose Diverse Weight Averaging (DiWA), a new WA strategy whose main motivation is to increase the functional diversity across averaged models. To this end, DiWA averages weights obtained from several independent training runs: indeed, models obtained from different runs are more diverse than those collected along a single run thanks to differences in hyperparameters and training procedures. We motivate the need for diversity by a new bias-variance-covariance-locality decomposition of the expected error, exploiting similarities between WA and standard functional ensembling. Moreover, this decomposition highlights that WA succeeds when the variance term dominates, which we show occurs when the marginal distribution changes at test time. Experimentally, DiWA consistently improves the state of the art on DomainBed without inference overhead.

The work in this chapter has led to the publication of a conference paper:

- A. Rame*, M. Kirchmeyer*, T. Rahier, A. Rakotomamonjy, P. Gallinari, and M. Cord (2022). "Diverse Weight Averaging for Out-of-Distribution Generalization". In: *Neural Information Processing Systems (NeurIPS).*

5.1   Introduction

Learning robust models that generalize well is critical for many real-world applications (Zech et al. 2018; DeGrave et al. 2021). Yet, the classical Expected Risk Minimization (ERM) lacks robustness to distribution shifts (Hendrycks et al. 2019; Shah et al. 2020; D'Amour et al. 2020). To improve out-of-distribution (OOD) generalization in classification, several recent works proposed to train models simultaneously on multiple related but different domains (Muandet et al. 2013). Though theoretically appealing, domain-invariant approaches (Peters et al. 2016) either underperform (Arjovsky et al. 2019a; Krueger et al. 2021) or only slightly improve (Sun et al. 2016; Rame et al. 2022) ERM on the reference DomainBed benchmark (Gulrajani et al. 2021b). The state-of-the-art strategy on DomainBed is currently to average the weights obtained along a training trajectory (Izmailov et al. 2018). Cha et al. 2021 argues that this WA succeeds in OOD because it finds solutions with flatter loss landscapes.

In this paper, we show the limitations of this flatness-based analysis and provide a new explanation for the success of WA in OOD. It is based on WA's similarity with ensembling (Lakshminarayanan et al. 2017), a well-known strategy to improve robustness (Ovadia et al. 2019; Ashukha et al. 2020), that averages the predictions from various models. Based on Ueda et al. 1996, we present a bias-variance-covariance-locality decomposition of WA's expected error. It contains four terms: *first* the bias that we show increases under shift in label posterior distributions (*i.e.* correlation shift (Ye et al. 2022)); *second*, the variance that we show increases under shift in input marginal distributions (*i.e.* diversity shift (Ye et al. 2022)); *third*, the covariance that decreases when models are diverse; *finally*, a locality condition on the weights of averaged models.

Based on this analysis, we aim at obtaining diverse models whose weights are averageable with our Diverse Weight Averaging (DiWA) approach. In practice, DiWA averages in weights the models obtained from independent training runs that share the same initialization. The motivation is that those models are more diverse than those obtained along a single run (Fort et al. 2019; Gontijo-Lopes et al. 2022). Yet, averaging the weights of independently trained networks with batch normalization (Ioffe et al. 2015) and ReLU layers (Agarap 2018) may be counterintuitive. Such averaging is efficient especially when models can be connected linearly in the weight space via a low loss path. Interestingly, this linear mode connectivity property (Frankle et al. 2020) was empirically validated when the runs start from a shared pretrained initialization (Neyshabur et al. 2020). This insight is at the heart of DiWA but also of other recent works (Wortsman et al. 2022b; Wortsman et al. 2022a; Matena et al. 2021).

In summary, our main contributions are the following:

- We propose a new theoretical analysis of WA for OOD based on a bias-variance-covariance-locality decomposition of its expected error (Section 5.2). By relating correlation shift to its bias and diversity shift to its variance, we show that WA succeeds under diversity shift.

- We empirically tackle covariance by increasing the diversity across averaged models. In our DiWA approach, we decorrelate their training procedures: in practice, these models are obtained from independent runs (Section 5.3). We then empirically validate that diversity improves OOD performance (Section 5.4) and show that DiWA is state of the art on all real-world datasets from the DomainBed benchmark (Gulrajani et al. 2021b) (Section 5.5).

5.2 Theoretical insights

We introduce WA in Section 5.2.1 and decompose its expected OOD error in Section 5.2.2. Then, we separately consider the four terms of this bias-variance-covariance-locality decomposition in Section 5.2.3. This theoretical analysis will allow us to better understand when WA succeeds, and most importantly, how to improve it empirically in Section 5.3.

5.2.1 WA for OOD and limitations of current analysis

Setting. Given the notations and general setting in Section 2.1.1, we recall the main elements of context.

We consider a NN $f(\cdot, \theta) : \mathcal{X} \to \mathcal{Y}$ made of a fixed architecture f with weights θ. S is the training (source) domain with distribution p_S, and T is the test (target) domain with distribution p_T. f_S, f_T are the labeling functions in S and T. The weights are typically learned on a training dataset $\mathcal{D}_S$ from S (composed of n_S Independent and Identically Distributed (IID) samples from $p_S(X, Y)$) with a configuration c, which contains all other sources of randomness in learning (*e.g.* initialization, hyperparameters, training stochasticity, epochs, *etc.*). We call $l_S = \{\mathcal{D}_S, c\}$ a learning procedure on domain S, and explicitly write $\theta(l_S)$ to refer to the weights obtained after stochastic minimization of $1/n_S \sum_{(x,y) \in \mathcal{D}_S} \ell(f(x, \theta), y)$ w.r.t. θ under l_S.

We seek θ minimizing the target generalization error $\mathcal{E}_T(\theta) \triangleq \mathcal{E}_T(f(\cdot, \theta))$ in Eq. (2.2). However, this is complex in the OOD setup because we only have data from domain S in training, related yet different from T. The differences between S and T are due to distribution shifts. These shifts are decomposed per Ye et al. 2022 into **diversity shift** (a.k.a. covariate shift), when marginal distributions differ

(*i.e.* $p_S(X) \neq p_T(X)$), and **correlation shift** (a.k.a. concept shift), when posterior distributions differ (*i.e.* $p_S(Y|X) \neq p_T(Y|X)$ and $f_S \neq f_T$).

WA. We study the benefits of combining M individual member weights $\{\theta_m\}_{m=1}^{M} \triangleq \{\theta(l_S^{(m)})\}_{m=1}^{M}$ obtained from M (potentially correlated) Identically Distributed (ID) learning procedures $L_S^M \triangleq \{l_S^{(m)}\}_{m=1}^{M}$. Under conditions discussed in Section 5.3.2, these M weights can be averaged despite nonlinearities in the architecture f. WA (Izmailov et al. 2018), defined as:

$$f_{\text{WA}} \triangleq f(\cdot, \theta_{\text{WA}}), \text{ where } \theta_{\text{WA}} \triangleq \theta_{\text{WA}}(L_S^M) \triangleq 1/M \sum_{m=1}^{M} \theta_m, \qquad (5.1)$$

is the state of the art (Cha et al. 2021; Arpit et al. 2021) on DomainBed (Gulrajani et al. 2021b) when the weights $\{\theta_m\}_{m=1}^{M}$ are sampled along a single training trajectory (a description we refine in Appendix C.2.2 from Appendix C.2.2).

Limitations of the flatness-based analysis. To explain this success, Cha et al. 2021 argue that flat minima generalize better; indeed, WA flattens the loss landscape. Yet, as shown in Appendix C.1, this analysis does not fully explain WA's spectacular results on DomainBed. First, flatness does not act on distribution shifts thus the OOD error is uncontrolled with their upper bound (see Appendix C.1.1). Second, this analysis does not clarify why WA outperforms Sharpness-Aware Minimizer (SAM) (Foret et al. 2021) for OOD generalization, even though SAM directly optimizes flatness (see Appendix C.1.2). Finally, it does not justify why combining WA and SAM succeeds in IID (Kaddour et al. 2022) yet fails in OOD (see Appendix C.1.3). These observations motivate a new analysis of WA; we propose one below that better explains these results.

5.2.2   Bias-variance-covariance-locality decomposition

We now introduce our bias-variance-covariance-locality decomposition which extends the bias-variance decomposition (Kohavi et al. 1996) to WA. In the rest of this theoretical section, ℓ is the Mean-Squared Error (MSE) for simplicity: yet, our results may be extended to other losses as in Domingos 2000. In this case, the expected error of a model with weights $\theta(l_S)$ w.r.t. the learning procedure l_S was decomposed in Kohavi et al. 1996 into:

$$\mathbb{E}_{l_S} \mathcal{E}_T(\theta(l_S)) = \mathbb{E}_{(x,y)\sim p_T}[\text{bias}^2(x,y) + \text{var}(x)], \qquad \text{(BV)}$$

where $\text{bias}(x, y), \text{var}(x)$ are the bias and variance of the considered model w.r.t. a sample (x, y), defined later in Eq. (BVCL). To decompose WA's error, we leverage

the similarity as Izmailov et al. 2018 between WA and Ensembles (ENS) (Lakshminarayanan et al. 2017; Dieterich 2000), a more traditional way to combine a collection of weights. More precisely, ENS averages the predictions, $f_{ENS} \triangleq f_{ENS}(\cdot, \{\theta_m\}_{m=1}^M) \triangleq 1/M \sum_{m=1}^M f(\cdot, \theta_m)$. Lemma 5.1 establishes that f_{WA} is a first-order approximation of f_{ENS} when $\{\theta_m\}_{m=1}^M$ are close in the weight space.

Lemma 5.1 (WA and ENS. Proof in Appendix C.2.1. Adapted from Izmailov et al. 2018; Wortsman et al. 2022a.). *Given* $\{\theta_m\}_{m=1}^M$ *with learning procedures* $L_S^M \triangleq \{l_S^{(m)}\}_{m=1}^M$. *Denoting* $\Delta_{L_S^M} = \max_{m=1}^M \|\theta_m - \theta_{WA}\|_2$, $\forall (x,y) \in \mathcal{X} \times \mathcal{Y}$:

$$f_{WA}(x) = f_{ENS}(x) + O(\Delta_{L_S^M}^2) \text{ and } \ell(f_{WA}(x), y) = \ell(f_{ENS}(x), y) + O(\Delta_{L_S^M}^2).$$

This similarity is useful since Eq. (BV) was extended into a bias-variance-covariance decomposition for ENS in Ueda et al. 1996; Brown et al. 2005. We can then derive the following decomposition of WA's expected test error. To take into account the M averaged weights, the expectation is over the joint distribution describing the M ID learning procedures $L_S^M \triangleq \{l_S^{(m)}\}_{m=1}^M$.

Proposition 5.1 (Bias-variance-covariance-locality decomposition of WA's expected OOD generalization error. Proof in Appendix C.2.2.). *With* $\bar{f}_S(x) = \mathbb{E}_{l_S}[f(x, \theta(l_S))]$, *under* ID *learning procedures* $L_S^M \triangleq \{l_S^{(m)}\}_{m=1}^M$, *the expected generalization error on domain* T *of* $\theta_{WA}(L_S^M) \triangleq \frac{1}{M} \sum_{m=1}^M \theta_m$ *over the joint distribution of* L_S^M *is:*

$$\mathbb{E}_{L_S^M} \mathcal{E}_T(\theta_{WA}(L_S^M)) = \mathbb{E}_{(x,y) \sim p_T} \left[\text{bias}^2(x, y) + \frac{1}{M} \text{var}(x) + \frac{M-1}{M} \text{cov}(x) \right] + O(\bar{\Delta}^2),$$

$$\text{where bias}(x, y) = y - \bar{f}_S(x),$$

$$\text{and var}(x) = \mathbb{E}_{l_S} \left[\left(f(x, \theta(l_S)) - \bar{f}_S(x) \right)^2 \right],$$

$$\text{and cov}(x) = \mathbb{E}_{l_S, l_S'} \left[\left(f(x, \theta(l_S)) - \bar{f}_S(x) \right) \left(f(x, \theta(l_S')) - \bar{f}_S(x) \right) \right],$$

$$\text{and } \bar{\Delta}^2 = \mathbb{E}_{L_S^M} \Delta_{L_S^M}^2 \text{ with } \Delta_{L_S^M} = \max_{m=1}^M \|\theta_m - \theta_{WA}\|_2.$$

$$\tag{BVCL}$$

cov *is the prediction covariance between two member models whose weights are averaged. The locality term* $\bar{\Delta}^2$ *is the expected* MSE *between weights and their average.*

Eq. (BVCL) decomposes WA's OOD error into four terms. Bias is the same as that of each of its ID members. Variance is split into the variance of each of its ID members divided by M and a covariance term. The last locality term constrains the weights to ensure the validity of our approximation. In conclusion, combining M models divides the variance by M but introduces the covariance and locality terms which should be controlled along bias to guarantee low OOD error.

5.2.3   Analysis of the BVCL decomposition

We now analyze the four terms in Eq. (BVCL). We show that bias dominates under correlation shift (Section 5.2.3) and variance dominates under diversity shift (Section 5.2.3). Then, we discuss a trade-off between covariance, reduced with diverse models (Section 5.2.3), and the locality term, reduced when weights are similar (Section 5.2.3). This analysis shows that *WA is effective against diversity shift when M is large and when its members are diverse but close in the weight space.*

Bias and correlation shift (and support mismatch)   We relate OOD bias to correlation shift (Ye et al. 2022) under Assumption 9, where $\bar{f}_S(x) \triangleq \mathbb{E}_{l_S}[f(x, \theta(l_S))]$. As discussed in Appendix C.2.3.2, Assumption 9 is reasonable for a large NN trained on a large dataset representative of the source domain S. It is relaxed in Proposition C.2 from Appendix C.2.3.

Assumption 9 (Small IID bias). $\exists \epsilon > 0$ small s.t. $\forall x \in \mathcal{X}_S, |f_S(x) - \bar{f}_S(x)| \leq \epsilon$.

Proposition 5.2 (OOD bias and correlation shift. Proof in Appendix C.2.3). *With a bounded difference between the labeling functions $f_T - f_S$ on $\mathcal{X}_T \cap \mathcal{X}_S$, under Assumption 9, the bias on domain T is:*

$$\mathbb{E}_{(x,y)\sim p_T}[\text{bias}^2(x, y)] = \textit{Correlation shift} + \textit{Support mismatch} + O(\epsilon),$$

$$\textit{where Correlation shift} = \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T(x) - f_S(x))^2 p_T(x)dx,$$

$$\textit{and Support mismatch} = \int_{\mathcal{X}_T \setminus \mathcal{X}_S} (f_T(x) - \bar{f}_S(x))^2 p_T(x)dx.$$

(5.2)

We analyze the first term by noting that $f_T(x) \triangleq \mathbb{E}_{p_T}[Y|X = x]$ and $f_S(x) \triangleq \mathbb{E}_{p_S}[Y|X = x]$, $\forall x \in \mathcal{X}_T \cap \mathcal{X}_S$. This expression confirms that our correlation shift term measures shifts in posterior distributions between source and target, as in Ye et al. 2022. It increases in presence of spurious correlations: *e.g.* on ColoredMNIST (Arjovsky et al. 2019a) where the color/label correlation is reversed at test time. The second term is caused by support mismatch between source and target. It was analyzed in Ruan et al. 2022 and shown irreducible in their "No free lunch for learning representations for DG". Yet, this term can be tackled if we transpose the analysis in the feature space rather than the input space. This motivates encoding the source and target domains into a shared latent space, *e.g.* by pretraining the encoder on a task with minimal domain-specific information as Ruan et al. 2022.

This analysis explains why WA fails under correlation shift, as shown on ColoredMNIST in Appendix C.7. Indeed, combining different models does *not* reduce the bias. Section 5.2.3 explains that WA is however efficient against diversity shift.

Variance and diversity shift   Variance is known to be large in OOD (D'Amour et al. 2020) and to cause a phenomenon named underspecification, when models behave differently in OOD despite similar test IID accuracy. We now relate OOD variance to diversity shift (Ye et al. 2022) in a simplified setting. We fix the source dataset $\mathcal{D}_S$ (with input support $X_{\mathcal{D}_S}$), the target dataset $\mathcal{D}_T$ (with input support $X_{\mathcal{D}_T}$) and the network's initialization. We get a closed-form expression for the variance of f over all other sources of randomness under Assumptions 10 and 11.

Assumption 10 (Kernel regime). f is in the kernel regime (Jacot et al. 2018).

This states that f behaves as a Gaussian Process (GP); it is reasonable if f is a wide network (Jacot et al. 2018; Lee et al. 2017). The corresponding kernel K is the Neural Tangent Kernel (NTK) (Jacot et al. 2018) depending only on the initialization. GPs are useful because their variances have a closed-form expression (Appendix C.2.4.1). To simplify the expression of variance, we now make Assumption 11.

Assumption 11 (Constant norm, low intra-sample similarity on $\mathcal{D}_S$). $\exists$ $0 \leq \epsilon \ll \lambda_S$ s.t. $\forall x_S \in X_{\mathcal{D}_S}, K(x_S, x_S) = \lambda_S$ and $\forall x'_S \neq x_S \in X_{\mathcal{D}_S}, |K(x_S, x'_S)| \leq \epsilon$.

This states that training samples have the same norm (following standard practice (Lee et al. 2017; Ah-Pine 2010; Ghojogh et al. 2021; Rennie 2005)) and weakly interact (He et al. 2020; Seleznova et al. 2022). This assumption is further discussed and relaxed in Appendix C.2.4.2. We are now in a position to relate variance and diversity shift when $\epsilon \to 0$.

Proposition 5.3 (OOD variance and diversity shift. Proof in Appendix C.2.4). *Given f trained on source dataset $\mathcal{D}_S$ (of size n_S) with NTK K, under Assumptions 10 and 11, the variance on dataset d_T is:*

$$\mathbb{E}_{x_T \in X_{\mathcal{D}_T}}[\mathrm{var}(x_T)] = \frac{n_S}{2\lambda_S} MMD^2(X_{\mathcal{D}_S}, X_{\mathcal{D}_T}) + \lambda_T - \frac{n_S}{2\lambda_S}\beta_T + O(\epsilon), \qquad (5.3)$$

where MMD is the empirical Maximum Mean Discrepancy (MMD) in the RKHS of $K^2(x, y) = (K(x, y))^2$; $\lambda_T \triangleq \mathbb{E}_{x_T \in X_{\mathcal{D}_T}} K(x_T, x_T)$ and $\beta_T \triangleq \mathbb{E}_{(x_T, x'_T) \in X^2_{\mathcal{D}_T}, x_T \neq x'_T} K^2(x_T, x'_T)$ are the empirical mean similarities respectively measured between identical (w.r.t. K) and different (w.r.t. K^2) samples averaged over $X_{\mathcal{D}_T}$.

The MMD empirically estimates shifts in input marginals, *i.e.* between $p_S(X)$ and $p_T(X)$. Our expression of variance is thus similar to the diversity shift formula in Ye et al. 2022: MMD replaces the L_1 divergence used in Ye et al. 2022. The other terms, λ_T and β_T, both involve internal dependencies on the target dataset $\mathcal{D}_T$: they are constants w.r.t. $X_{\mathcal{D}_T}$ and do not depend on distribution shifts. At fixed $\mathcal{D}_T$ and under our assumptions, Eq. (5.3) shows that variance on $\mathcal{D}_T$ decreases when $X_{\mathcal{D}_S}$ and $X_{\mathcal{D}_T}$ are closer (for the MMD distance defined by the kernel K^2) and increases when they deviate. Intuitively, the further $X_{\mathcal{D}_T}$ is from $X_{\mathcal{D}_S}$, the less the model's predictions on $X_{\mathcal{D}_T}$ are constrained after fitting $\mathcal{D}_S$.

This analysis shows that WA reduces the impact of diversity shift as combining M models divides the variance per M. This is a strong property achieved *without requiring data from the target domain*.

Covariance and diversity   The covariance term increases when the predictions of $\{f(\cdot, \theta_m)\}_{m=1}^{M}$ are correlated. In the worst case where all predictions are identical, covariance equals variance and WA is no longer beneficial. On the other hand, the lower the covariance, the greater the gain of WA over its members; this is derived by comparing Equations (BV) and (BVCL), as detailed in Appendix C.2.5. It motivates tackling covariance by encouraging members to make different predictions, thus to be functionally diverse. Diversity is a widely analyzed concept in the ensemble literature (Lakshminarayanan et al. 2017), for which numerous measures have been introduced (Kuncheva et al. 2003; Aksela 2003; Kornblith et al. 2019). In Section 5.3, we aim at decorrelating the learning procedures to increase members' diversity and reduce the covariance term.

Locality and linear mode connectivity   To ensure that WA approximates ENS, the last locality term $O(\bar{\Delta}^2)$ constrains the weights to be close. Yet, the covariance term analyzed in Section 5.2.3 is antagonistic, as it motivates functionally diverse models. Overall, to reduce WA's error in OOD, we thus seek a good trade-off between diversity and locality. In practice, we consider that the main goal of this locality term is to ensure that the weights are averageable despite the nonlinearities in the NN such that WA's error does not explode. This is why in Section 5.3, we empirically relax this locality constraint and simply require that the weights are linearly connectable in the loss landscape, as in the linear mode connectivity (Frankle et al. 2020). We empirically verify later in Figure 5.1 that the approximation $f_{\text{WA}} \approx f_{\text{ENS}}$ remains valid even in this case.

5.3   DiWA: Diverse Weight Averaging

5.3.1   Motivation: different runs for more diversity

Limitations of previous WA approaches.   Our analysis in Section 5.2.3 showed that the bias and the variance terms are mostly fixed by the distribution shifts at hand. In contrast, the covariance term can be reduced by enforcing diversity across models (Section 5.2.3) obtained from learning procedures $\{l_S^{(m)}\}_{m=1}^{M}$. Yet, previous methods (Cha et al. 2021; Arpit et al. 2021) only average weights obtained along a single run. This corresponds to highly correlated procedures sharing the same initialization, hyperparameters, batch orders, data augmentations and noise, that

only differ by the number of training steps. The models are thus mostly similar: this does not leverage the full potential of WA.

DiWA. Our Diverse Weight Averaging approach seeks to reduce the OOD expected error in Eq. (BVCL) by decreasing covariance across predictions: DiWA decorrelates the learning procedures $\{l_S^{(m)}\}_{m=1}^M$. Our weights are obtained from $M \gg 1$ different runs, with diverse learning procedures: these have different hyperparameters (learning rate, weight decay and dropout probability), batch orders, data augmentations (*e.g.* random crops, horizontal flipping, color jitter, grayscaling), stochastic noise and number of training steps. Thus, the corresponding models are more diverse on domain T per Gontijo-Lopes et al. 2022 and reduce the impact of variance when M is large. However, this may break the locality requirement analyzed in Section 5.2.3 if the weights are too distant. Empirically, we show that DiWA works under two conditions: shared initialization and mild hyperparameter ranges.

5.3.2 Approach

Shared initialization. The shared initialization condition follows Neyshabur et al. 2020: when models are fine-tuned from a shared pretrained model, their weights can be connected along a linear path where error remains low (Frankle et al. 2020). Following standard practice on DomainBed (Gulrajani et al. 2021b), our encoder is pretrained on ImageNet (Krizhevsky et al. 2012); this pretraining is key as it controls the bias (by defining the feature support mismatch, see Section 5.2.3) and variance (by defining the kernel K, see Appendix C.2.4.4). Regarding the classifier initialization, we test two methods. The first is the random initialization, which may distort the features (Kumar et al. 2022). The second is Linear Probing (LP) (Kumar et al. 2022): it first learns the classifier (while freezing the encoder) to serve as a shared initialization. Then, LP fine-tunes the encoder and the classifier together in the M subsequent runs; the locality term is smaller as weights remain closer (see Kumar et al. 2022).

Mild hyperparameter search. As shown in Figure 5.5, extreme hyperparameter ranges lead to weights whose average may perform poorly. Indeed, weights obtained from extremely different hyperparameters may not be linearly connectable; they may belong to different regions of the loss landscape. In our experiments, we thus use the mild search space defined in Table C.5, first introduced in SWAD (Cha et al. 2021). These hyperparameter ranges induce diverse models that are averageable in weights.

Weight selection. The last step of our approach (summarized in Algorithm B.1) is to choose which weights to average among those available. We explore two simple weight selection protocols, as in Wortsman et al. 2022a. The first *uniform* equally averages all weights; it is practical but may underperform when some runs are detrimental. The second *restricted* (*greedy* in Wortsman et al. 2022a) solves this drawback by restricting the number of selected weights: weights are ranked in decreasing order of validation accuracy and sequentially added only if they improve DiWA's validation accuracy.

In the following sections, we experimentally validate our theory. First, Section 5.4 confirms our findings on the OfficeHome dataset (Venkateswara et al. 2017b) where diversity shift dominates (Ye et al. 2022) (see Appendix C.4.5 for a similar analysis on PACS (Li et al. 2017a)). Then, Section 5.5 shows that DiWA is state of the art on DomainBed (Gulrajani et al. 2021b).

Algorithm 5.1 DiWA Pseudo-code

Require: θ_0 pretrained encoder and initialized classifier; $\{h_m\}_{m=1}^{H}$ hyperparameter configurations.

1: *Training:* $\forall m = 1$ to H, $\theta_m \triangleq \mathrm{FineTune}(\theta_0, h_m)$
2: *Weight selection:*
3: *Uniform:* $\mathcal{M} = \{1, \cdots, H\}$.
4: *Restricted:* Rank $\{\theta_m\}_{m=1}^{H}$ by decreasing $\mathrm{ValAcc}(\theta_m)$. $\mathcal{M} \leftarrow \emptyset$.
5: **for** $m = 1$ to H **do**
6:      **If** $\mathrm{ValAcc}(\theta_{\mathcal{M} \cup \{m\}}) \geq \mathrm{ValAcc}(\theta_{\mathcal{M}})$
7:      $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
8: **end for**
9: *Inference:* with $f(\cdot, \theta_{\mathcal{M}})$, where $\theta_{\mathcal{M}} = \sum_{m \in \mathcal{M}} \theta_m / |\mathcal{M}|$.

5.4 Empirical validation of our theoretical insights

We consider several collections of weights $\{\theta_m\}_{m=1}^{M}$ ($2 \leq M < 10$) trained on the "Clipart", "Product" and "Photo" domains from OfficeHome (Venkateswara et al. 2017b) with a shared random initialization and mild hyperparameter ranges. These weights are first indifferently sampled from a single run (every 50 batches) or from different runs. They are evaluated on "Art", the last domain.

WA vs. ENS. Figure 5.1 validates Lemma 5.1 and that $f_{\textbf{WA}} \approx f_{\textbf{ENS}}$. More precisely, $f_{\textbf{WA}}$ slightly but consistently improves $f_{\textbf{ENS}}$: we discuss this in Appendix C.3. Moreover, a larger M improves the results; in accordance with Eq. (BVCL), this
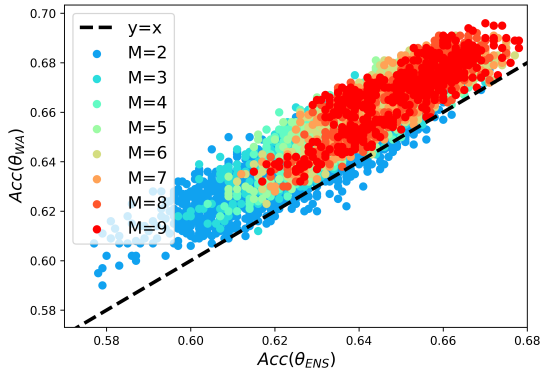
Figure 5.1. – Each dot displays the accuracy (↑) of WA vs. accuracy (↑) of ENS for M models.
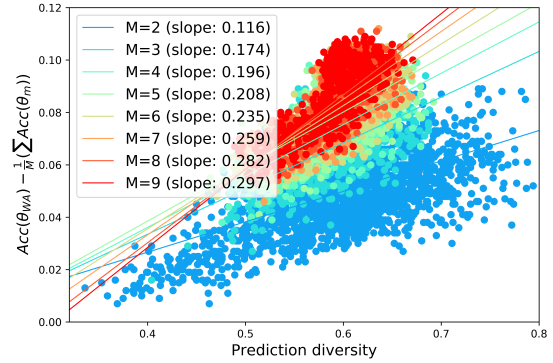
Figure 5.2. – Each dot displays the accuracy (↑) gain of WA over its members vs. the prediction diversity (Aksela 2003) (↑) for M models.

motivates averaging as many weights as possible. In contrast, large M is computationally impractical for ENS at test time, requiring M forwards.

Diversity and accuracy. We validate in Figure 5.2 that f_{WA} benefits from diversity. Here, we measure diversity with the ratio-error (Aksela 2003), *i.e.* the ratio $N_{\mathrm{diff}}/N_{\mathrm{simul}}$ between the number of different errors N_{diff} and of simultaneous errors N_{simul} in test for a pair in $\{f(\cdot, \theta_m)\}_{m=1}^{M}$. A higher average over the $\binom{M}{2}$ pairs means that members are less likely to err on the same inputs. Specifically, the gain of $\mathrm{Acc}(\theta_{\mathrm{WA}})$ over the mean individual accuracy $\frac{1}{M}\sum_{m=1}^{M}\mathrm{Acc}(\theta_m)$ increases with diversity. Moreover, this phenomenon intensifies for larger M: the linear regression's slope (*i.e.* the accuracy gain per unit of diversity) increases with M. This is consistent with the $(M-1)/M$ factor of $\mathrm{cov}(x)$ in Eq. (BVCL), as further highlighted in Appendix C.4.2. Finally, in Appendix C.4.1, we show that the conclusion also holds with CKAC (Kornblith et al. 2019), another established diversity measure.

Increasing diversity thus accuracy via different runs. Now we investigate the difference between sampling the weights from a single run or from different runs. Figure 5.3 *first* shows that diversity increases when weights come from different runs. *Second*, in Figure 5.4, this is reflected on the accuracies in OOD. Here, we rank by validation accuracy the 60 weights obtained (1) from 60 different runs and (2) along 1 well-performing run. We then consider the WA of the top M weights as M increases from 1 to 60. Both have initially the same performance and improve with M; yet, WA of weights from different runs gradually outperforms the single-run WA. *Finally*, Figure 5.5 shows that this holds only for mild hyperparameter
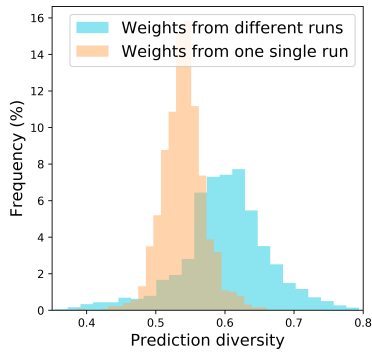
Figure 5.3. – Frequencies of prediction diversities (↑) across 2 weights.
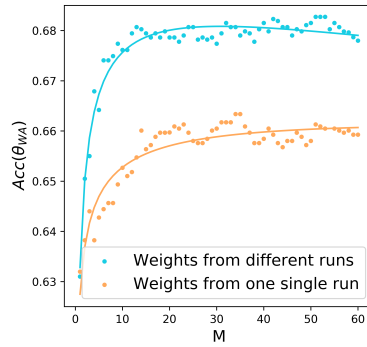
Figure 5.4. – WA's accuracy (↑) as the number of weights M increases.
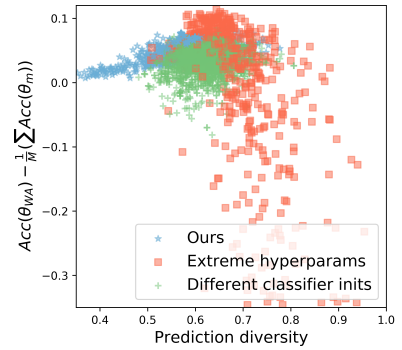
Figure 5.5. – WA's accuracy gain vs. prediction diversity (↑) - $2 \leq M < 10$.

ranges and with a shared initialization. Otherwise, when hyperparameter distributions are extreme (as defined in Table C.5) or when classifiers are not similarly initialized, DiWA may perform worse than its members due to a violation of the locality condition. These experiments confirm that *diversity is key as long as the weights remain averageable*.

5.5 Experimental results on DomainBed

Datasets. We now present our evaluation on DomainBed (Gulrajani et al. 2021b). By imposing the code, the training procedures and the ResNet50 (He et al. 2016b) architecture, DomainBed is arguably the fairest benchmark for OOD generalization. It includes 5 multi-domain real-world datasets: PACS (Li et al. 2017a), VLCS (Fang et al. 2013), OfficeHome (Venkateswara et al. 2017b), TerraIncognita (Beery et al. 2018b) and DomainNet (Peng et al. 2019). Ye et al. 2022 showed that *diversity shift dominates in these datasets*. Each domain is successively considered as the target T while other domains are merged into the source S. The validation dataset is sampled from S, *i.e.* we follow DomainBed's training-domain model selection. The experimental setup is further described in Appendix C.6.1.

Baselines. ERM is the standard Empirical Risk Minimization. Coral (Sun et al. 2016) is the best approach based on domain invariance. SWAD (Stochastic Weight Averaging Densely) (Cha et al. 2021) and MA (Moving Average) (Arpit et al. 2021) average weights along one training trajectory but differ in their weight selection strategy. SWAD (Cha et al. 2021) is the current state of the art (SoTA) thanks to it "overfit-aware" strategy, yet at the cost of three additional hyperparameters (a patient parameter, an overfitting patient parameter and a tolerance rate) tuned

per dataset. In contrast, MA (Arpit et al. 2021) is easy to implement as it simply combines all checkpoints uniformly starting from batch 100 until the end of training. Finally, we report the scores obtained in Arpit et al. 2021 for the costly Deep Ensembles (DENS) (Lakshminarayanan et al. 2017) (with different initializations): we discuss other ensembling strategies in Appendix C.3.

Our runs. ERM and DiWA share the same training protocol in DomainBed: yet, instead of keeping only one run from the grid-search, DiWA leverages M runs. In practice, we sample 20 configurations from the hyperparameter distributions detailed in Table C.5 and report the mean and standard deviation across 3 data splits. For each run, we select the weights of the epoch with the highest validation accuracy. ERM and MA select the model with highest validation accuracy across the 20 runs, following standard practice on DomainBed. ENS averages the predictions of all $M = 20$ models (with shared initialization). DiWA-restricted selects $1 \leq M \leq 20$ weights with Algorithm B.1 while DiWA-uniform averages all $M = 20$ weights. DiWA† averages uniformly the $M = 3 \times 20 = 60$ weights from all 3 data splits. DiWA† benefits from larger M (without additional inference cost) and from data diversity (see Appendix C.4.3). However, we cannot report standard deviations for DiWA† for computational reasons. Moreover, DiWA† cannot leverage the restricted weight selection, as the validation is not shared across all 60 weights that have different data splits.

5.5.1 Results on DomainBed

We report our **main results** in Table 5.1, detailed per domain in Appendix E.1. With a randomly initialized classifier, DiWA†-uniform is the best on PACS, VLCS and OfficeHome: DiWA-uniform is the second best on PACS and OfficeHome. On TerraIncognita and DomainNet, DiWA is penalized by some bad runs, filtered in DiWA-restricted which improves results on these datasets. Classifier initialization with LP (Kumar et al. 2022) improves all methods on OfficeHome, TerraIncognita and DomainNet. On these datasets, DiWA† increases MA by 1.3, 0.5 and 1.1 points respectively. After averaging, DiWA† with LP *establishes a new SoTA of* 68.0%, improving SWAD by 1.1 points.

DiWA with different objectives. So far we used ERM that does not leverage the domain information. Table 5.2 shows that DiWA-uniform benefits from averaging weights trained with Interdomain Mixup (Yan et al. 2020) and Coral (Sun et al. 2016): accuracy gradually improves as we add more objectives. Indeed, as highlighted in Appendix C.4.3, DiWA benefits from the increased diversity brought by the various objectives. This also suggests a new kind of linear connectivity across models despite being trained with different objectives; the complete analysis of this is left for future research.

Table 5.1. – **Accuracy (%, ↑) on DomainBed with ResNet50** (best in **bold** and second best <u>underlined</u>).

	Method	Weight selection	Init	PACS	VLCS	OfficeHome	TerraInc	DomainNet	Avg
	ERM	N/A		85.5 ± 0.2	77.5 ± 0.4	66.5 ± 0.3	46.1 ± 1.8	40.9 ± 0.1	63.3
	Coral	N/A		86.2 ± 0.3	78.8 ± 0.6	68.7 ± 0.3	47.6 ± 1.0	41.5 ± 0.1	64.6
	SWAD	Overfit-aware	Random	88.1 ± 0.1	79.1 ± 0.1	70.6 ± 0.2	50.0 ± 0.3	46.5 ± 0.1	66.9
	MA	Uniform		87.5 ± 0.2	78.2 ± 0.2	70.6 ± 0.1	50.3 ± 0.5	46.0 ± 0.1	66.5
	DENS	Uniform: $M = 6$		87.6	78.5	70.8	49.2	**47.7**	66.8
	ERM	N/A		85.5 ± 0.5	77.6 ± 0.2	67.4 ± 0.6	48.3 ± 0.8	44.1 ± 0.1	64.6
	MA	Uniform		87.9 ± 0.1	78.4 ± 0.1	70.3 ± 0.1	49.9 ± 0.2	46.4 ± 0.1	66.6
	ENS	Uniform: $M = 20$	Random	88.0 ± 0.1	78.7 ± 0.1	70.5 ± 0.1	51.0 ± 0.5	47.4 ± 0.2	67.1
	DiWA	Restricted: $M \leq 20$		87.9 ± 0.2	<u>79.2 ± 0.1</u>	70.5 ± 0.1	50.5 ± 0.5	46.7 ± 0.1	67.0
	DiWA	Uniform: $M = 20$		88.8 ± 0.4	79.1 ± 0.2	71.0 ± 0.1	48.9 ± 0.5	46.1 ± 0.1	66.8
	DiWA†	Uniform: $M = 60$		**89.0**	**79.4**	71.6	49.0	46.3	67.1
	ERM	N/A		85.9 ± 0.6	78.1 ± 0.5	69.4 ± 0.2	50.4 ± 1.8	44.3 ± 0.2	65.6
	MA	Uniform		87.8 ± 0.3	78.5 ± 0.4	71.5 ± 0.3	51.4 ± 0.6	46.6 ± 0.0	67.1
	ENS	Uniform: $M = 20$	LP	88.1 ± 0.3	78.5 ± 0.1	71.7 ± 0.1	50.8 ± 0.5	47.0 ± 0.2	67.2
	DiWA	Restricted: $M \leq 20$		88.0 ± 0.3	78.5 ± 0.1	71.5 ± 0.2	<u>51.6 ± 0.9</u>	**47.7 ± 0.1**	67.5
	DiWA	Uniform: $M = 20$		88.7 ± 0.2	78.4 ± 0.2	<u>72.1 ± 0.2</u>	51.4 ± 0.6	47.4 ± 0.2	<u>67.6</u>
	DiWA†	Uniform: $M = 60$		**89.0**	78.6	**72.8**	**51.9**	**47.7**	**68.0**

(left margin label spanning the second and third groups: Our runs)

Table 5.2. – **Accuracy (%, ↑) on OfficeHome** domain "Art" with various objectives.

Algorithm	No WA	MA	DiWA	DiWA†
ERM	62.9 ± 1.3	<u>65.0 ± 0.2</u>	67.3 ± 0.2	67.7
Mixup	63.1 ± 0.7	**66.2 ± 0.3**	67.8 ± 0.6	68.4
Coral	**64.4 ± 0.4**	64.4 ± 0.4	67.7 ± 0.2	68.2
ERM/Mixup	N/A	N/A	67.9 ± 0.7	<u>68.9</u>
ERM/Coral	N/A	N/A	<u>68.1 ± 0.3</u>	68.7
ERM/Mixup/Coral	N/A	N/A	**68.4 ± 0.4**	**69.1**

5.6 Discussion

Limitations of DiWA   Despite this success, DiWA has some limitations. *First*, DiWA cannot benefit from additional diversity that would break the linear connectivity between weights — as discussed in Appendix C.3. *Second*, DiWA (like all WA approaches) can tackle diversity shift but not correlation shift: this property is explained for the first time in Section 5.2.3 and illustrated in Appendix C.7 on ColoredMNIST.

Related work   We provided the related work for ensembling and DG in Section 2.1.3.3. We detail here more specifically the differences with the recent "Model soups" (Wortsman et al. 2022a), which developed a WA algorithm similar to Algorithm B.1. Yet, the task, the theoretical analysis and most importantly the goals of these two works are different. Regarding the task, Wortsman et al. 2022a and our work complement each other. Wortsman et al. 2022a demonstrates the robustness of "Model soups" out-of-distribution by evaluating it on several ImageNet variants with distribution shift. We illustrate that DiWA achieves state-of-the-art OOD performance against other established OOD methods via a thorough and fair comparison on the multi-domain DomainBed benchmark. Theoretically, we explain why WA succeeds under diversity shift. The bias/correlation shift, variance/diversity shift and diversity-based findings are novel; they are confirmed empirically. DiWA aims at combining weights from diverse models: our work may be analyzed as a general framework to average in weights models which can be obtained in various ways, as traditionally done in ensembling. In contrast, Wortsman et al. 2022a challenges the standard model selection after a grid search. Thus, DiWA and Wortsman et al. 2022a are theoretically complementary and applied successfully in different contexts.

5.7 Conclusion

In this paper, we propose a new explanation for the success of WA in OOD by leveraging its ensembling nature. Our analysis is based on a new bias-variance-covariance-locality decomposition for WA, where we theoretically relate bias to correlation shift and variance to diversity shift. It also shows that diversity is key to improve generalization. This motivates our DiWA approach that averages in weights models trained independently. DiWA improves the state of the art on DomainBed, the reference benchmark for OOD generalization. Critically, DiWA has no additional inference cost — removing a key limitation of standard ensem-

bling. Our work may encourage the community to further create diverse learning procedures and objectives — whose models may be averaged in weights.

Part IV

GENERALIZATION IN SPATIOTEMPORAL FORECASTING

In the following chapters, we consider the problem of generalization for spatiotemporal forecasting and adress two different settings.

In Chapter 6, we consider the problem of generalizing at test-time to new dynamics, *e.g.* when the parameters change. CoDA is a new approach to perform domain-conditioning for test-time parameter adaptation. CoDA can be seen as the adaptive extension of DiWA in Chapter 5: both approaches use the weights of multiple independent NNs, DiWA uses these weights explicitly while CoDA uses them implicitly via a linear hypernetwork.

In Chapter 7, we consider an alternative approach of generalizing, by designing better NN architectures for the problem at hand. We consider the problem of spatiotemporal generalization in Partial Differential Equation (PDE) modeling and propose a new solution that better accounts for the continuous nature of these dynamical systems.

The main contributions are outlined in Section 1.3.3.

Chapter 6

Context-Informed Dynamics Adaptation

Chapter abstract

Data-driven approaches to modeling physical systems fail to generalize to unseen systems that share the same general dynamics with the learning domain, but correspond to different physical contexts. We propose a new framework for this key problem, context-informed dynamics adaptation (CoDA), which takes into account the distributional shift across systems for fast and efficient adaptation to new dynamics. CoDA leverages multiple environments, each associated to a different dynamic, and learns to condition the dynamics model on contextual parameters, specific to each environment. The conditioning is performed via a hypernetwork, learned jointly with a context vector from observed data. The proposed formulation constrains the search hypothesis space for fast adaptation and better generalization across environments with few samples. We theoretically motivate our approach and show state-of-the-art generalization results on a set of nonlinear dynamics, representative of a variety of application domains. We also show, on these systems, that new system parameters can be inferred from context vectors with minimal supervision.

6.1   Introduction

NN approaches to modeling dynamical systems have recently raised the interest of several communities leading to an increasing number of contributions. This topic was explored in several domains, ranging from simple dynamics *e.g.* Hamiltonian systems (Greydanus et al. 2019; Chen et al. 2020b) to more complex settings *e.g.* fluid dynamics (Kochkov et al. 2021; Li et al. 2021c; Wandel et al. 2021), earth system science and climate (Reichstein et al. 2019), or health (Fresca et al. 2020). NN emulators are attractive as they may for example provide fast and low cost approximations to complex numerical simulations (Duraisamy et al. 2019; Kochkov et al. 2021), complement existing simulation models when the physical law is partially known (Yin et al. 2021b) or even offer solutions when classical solvers fail *e.g.* with very high number of variables (Sirignano et al. 2018).

A model of a real-world dynamical system should account for a wide range of contexts resulting from different external forces, spatio-temporal conditions, boundary conditions, sensors characteristics or system parameters. These contexts characterize the dynamics phenomenon. For instance, in cardiac electrophysiology (Neic et al. 2017; Fresca et al. 2020), each patient has its own specificities and represents a particular context. In the study of epidemics' diffusion (Shaier et al. 2021), computational models should handle a variety of spatial, temporal or even sociological contexts. The same holds for most physical problems, *e.g.* forecasting of spatial-location-dependent dynamics in climate (de Bézenac et al. 2018b), fluid dynamics prediction under distinct external forces (Li et al. 2021c), *etc*.

The physics approach for modeling dynamical systems relies on a strong prior knowledge about the underlying phenomenon. This provides a causal mechanism which is embedded in a physical dynamics model, usually a system of differential equations, and allows the physical model to handle a whole set of contexts. Moreover, it is often possible to adapt the model to new or evolving situations, *e.g.* via data assimilation (Kalman 1960; Courtier et al. 1994).

In contrast, ERM based Machine Learning (ML) fails to generalize to unseen dynamics. Indeed, it assumes IID data for training and inference while dynamical observations are non-IID due to changing initial conditions or physical contexts.

Thus any ML framework that handles this question should consider other assumptions. A common one used *e.g.* in DG (Wang et al. 2021b), states that data come from several environments a.k.a. domains, each with a different distribution. Training is performed on a sample of the environments and test corresponds to new ones. DG methods attempt to capture problem invariants via a unique model, assuming that there exists a representation space suitable for all the environments. This might be appropriate for classification, but not for dynamical

systems where the underlying dynamics differs for each environment. For this problem, we need to learn a function that adapts to each environment, based on a few observations, instead of learning a single domain-invariant function. This is the objective of meta-learning (Thrun et al. 1998), a general framework for fast adaptation to unknown contexts. The standard gradient-based methods (*e.g.* Finn et al. 2017) are unsuitable for complex dynamics due to their bi-level optimization and are known to overfit when little data is available for adaptation, as in the few-shot learning setting explored in this paper (Mishra et al. 2018). Like invariant methods, meta-learning usually handles basic tasks *e.g.* classification; regression on static data or simple sequences and not challenging dynamical systems.

Generalization for modeling real-world dynamical systems is a recent topic. Simple simulated dynamics were considered in Reinforcement Learning (Lee et al. 2020; Clavera et al. 2019) while physical dynamics were modeled in recent works (Yin et al. 2021a). These approaches consider either simplified settings or additional hypotheses *e.g.* prior knowledge and do not offer general solutions to our adaptation problem (details in Section 6.6).

We propose a new ML framework for generalization in dynamical systems, called COntext-informed Dynamics Adaptation (CoDA). Like in DG, we assume availability of several environments, each with its own specificity, yet sharing some physical properties. Training is performed on a sample of the environments. At test time, we assume access to example data from a new environment, here a trajectory. Our goal is to adapt to the new environment distribution with this trajectory. More precisely, CoDA assumes that the underlying system is described by a parametrized differential equation, either an Ordinary Differential Equation (ODE) or a PDE. The environments share the parametrized form of the equation but differ by the values of the parameters or initial conditions. CoDA conditions the dynamics model on learned environment characteristics a.k.a. contexts and generalizes to new environments and trajectories with few data. Our main contributions are:

- We introduce a multi-environment formulation of the generalization problem for dynamical systems.

- We propose a novel context-informed framework, CoDA, to this problem. It conditions the dynamics model on context vectors via a hypernetwork. CoDA introduces a locality and a low-rank constraint, which enable fast and efficient adaptation with few data.

- We analyze theoretically the validity of our low-rank adaptation setting for modeling dynamical systems.

- We evaluate two variations of CoDA on several ODEs/PDEs representative of a variety of application domains, *e.g.* chemistry, biology, physics. CoDA achieves

SOTA generalization results on in-domain and one-shot adaptation scenarios. We also illustrate how, with minimal supervision, CoDA infers accurately new system parameters from learned contexts.

The paper is organized as follows. In Section 6.2, we present our multi-environment problem. In Section 6.3, we introduce the CoDA framework. In Section 6.4, we detail implementation. In Section 6.5, we present our experimental results.

6.2   Generalization for Dynamical Systems

We present our generalization problem for dynamical systems, then introduce our multi-environment formalization.

6.2.1   Problem Setting

We recall the setting presented in Sections 2.2.1 and 2.2.2. We consider dynamical systems that are driven by unknown temporal differential equations of the form:

$$\frac{\partial v_t}{\partial t} = f(v_t), \tag{6.1}$$

where $t \in \mathbb{R}$ is a time index, v_t is a time-dependent state in a space $\mathcal{V}$ and $f : \mathcal{V} \to T\mathcal{V}$ a function that maps $v_t \in \mathcal{V}$ to its temporal derivatives in the tangent space $T\mathcal{V}$. f belongs to a class of vector fields $\mathcal{F}$. $\mathcal{V} \subseteq \mathbb{R}^d$ ($d \in \mathbb{N}^\star$) for ODEs or $\mathcal{V}$ is a space of functions defined over a spatial domain $\mathcal{X}$ (*e.g.* 2D or 3D Euclidean space) for PDEs.

Functions $f \in \mathcal{F}$ define a space $\mathcal{D}^f$ of state trajectories $v : \mathcal{T} \to \mathcal{V}$, mapping t in an interval $\mathcal{T}$ including 0, to the state $v_t \in \mathcal{V}$. Trajectories are defined by the initial condition $v_0 \sim p(V_0)$ and take the form:

$$\forall t \in \mathcal{T}, v_t = v_0 + \int_0^t f(v_\tau)\mathrm{d}\tau \in \mathcal{V} \tag{6.2}$$

In the following, we assume that $f \in \mathcal{F}$ is parameterized by unknown attributes *e.g.* physical parameters, external forcing terms which affect the trajectories.

6.2.2 Multi-Environment Learning Problem

We propose to learn the class of functions $\mathcal{F}$ with a data-driven *dynamics model* g_θ parametrized by $\theta \in \mathbb{R}^{d_\theta}$. Given $f \in \mathcal{F}$, we observe N trajectories in $\mathcal{D}^f$.

The standard ERM objective considers that all trajectories are IID. Here, we propose a multi-environment learning formulation where observed trajectories of f form an environment $e \in \mathcal{E}$. We denote f^e and $\mathcal{D}^e$ the corresponding function and set of N trajectories. We assume that we observe training environments $\mathcal{E}_{\text{tr}}$, consisting of several trajectories from a set of known functions $\{f^e\}_{e \in \mathcal{E}_{\text{tr}}}$.

The goal is to learn g_θ that adapts easily and efficiently to new environments $\mathcal{E}_{\text{ad}}$, corresponding to unseen functions $\{f^e\}_{e \in \mathcal{E}_{\text{ad}}}$ ("ad" stands for adaptation). We define $\forall e \in \mathcal{E}$ the MSE loss, over $\mathcal{D}^e$ as

$$\mathcal{L}(\theta, \mathcal{D}^e) \triangleq \sum_{v^e \in \mathcal{D}^e} \int_{t \in \mathcal{T}} \|f^e(v^e_t) - g_\theta(v^e_t)\|_2^2 \mathrm{d}t \qquad (6.3)$$

In practice, f^e is unavailable and we can only approximate it from discretized trajectories. We detail later in Eq. (6.10) our approximation method based on an integral formulation. It fits observed trajectories directly in state space.

6.3 The CoDA Learning Framework

We introduce CoDA, a new context-informed framework for learning dynamics on multiple environments. It relies on a general adaptation rule (Section 6.3.1) and introduces two key properties: locality, enforced in the objective (Section 6.3.2) and low-rank adaptation, enforced in the proposed model via hypernetwork-decoding (Section 6.3.3). The validity of this framework for dynamical systems is analyzed in Section 6.3.4 and its benefits are discussed in Section 6.3.5.

6.3.1 Adaptation Rule

The dynamics model g_θ should adapt to new environments. Hence, we propose to condition g_θ on observed trajectories $\mathcal{D}^e, \forall e \in \mathcal{E}$. Conditioning is performed via an *adaptation network* A_π, parametrized by π, which adapts the weights of g_θ to an environment $e \in \mathcal{E}$ according to

$$\theta^e \triangleq A_\pi(\mathcal{D}^e) \triangleq \theta^c + \delta\theta^e, \quad \pi \triangleq \{\theta^c, \{\delta\theta^e\}_{e \in \mathcal{E}}\} \qquad (6.4)$$

$\theta^c \in \mathbb{R}^{d_\theta}$ are shared parameters, used as an initial value for fast adaptation to new environments. $\delta\theta^e \in \mathbb{R}^{d_\theta}$ are environment-specific parameters conditioned on $\mathcal{D}^e$.

6.3.2   Constrained Optimization Problem

Given the adaptation rule in Eq. (6.4), we introduce a constrained optimization problem which learns parameters π such that $\forall e \in \mathcal{E}, \delta\theta^e$ is small and g fits observed trajectories. It introduces a locality constraint with a norm $\| \cdot \|$:

$$\min_\pi \sum_{e \in \mathcal{E}} \|\delta\theta^e\|^2 \text{ s.t. } \forall v^e \in \mathcal{D}^e, \forall t \in \mathbb{R}, \frac{\partial v_t^e}{\partial t} = g_{\theta^c + \delta\theta^e}(v_t^e)$$

We consider an approximation of this problem which relaxes the equality constraint with the MSE loss $\mathcal{L}$ in Eq. (6.3).

$$\min_\pi \sum_{e \in \mathcal{E}} \left(\mathcal{L}(\theta^c + \delta\theta^e, \mathcal{D}^e) + \lambda\|\delta\theta^e\|^2 \right) \tag{6.5}$$

λ is a hyperparameter. For training, we minimize Eq. (6.5) w.r.t. π over training environments $\mathcal{E}_{\text{tr}}$. After training, θ^c is frozen. For adaptation, we minimize Eq. (6.5) over new environments $\mathcal{E}_{\text{ad}}$ w.r.t. $\{\delta\theta^e\}_{e \in \mathcal{E}_{\text{ad}}}$.

The locality constraint in the training objective Eq. (6.5) enforces $\delta\theta^e$ to remain close to the shared θ^c solutions. It plays several roles. First, it fosters fast adaptation by acting as a constraint over $\theta^c \in \mathbb{R}^{d_\theta}$ during training s.t. minimas $\{\theta^{e\star}\}_{e \in \mathcal{E}}$ are in a neighborhood of θ^c i.e. can be reached from θ^c with few update steps. Second, it constrains the hypothesis space at fixed θ^c. Under some assumptions, it can simplify the resolution of the optimization problem w.r.t. $\delta\theta^e$ by turning optimization to a quadratic convex problem with an unique solution. We show this property for our solution in Proposition 6.1. The positive effects of this constraint will be illustrated on an ODE system in Section 6.3.3.

6.3.3   Context-Informed Hypernetwork

Eq. (6.5) involves learning $\delta\theta^e$ for each environment. For adaptation, $\delta\theta^e$ should be inferred from few observations of the new environment. Learning such high-dimensional parameters is prone to over-fitting, especially in low data regimes. We propose a hypernetwork-based solution (Figure 6.1) to solve efficiently this problem. It operates on a low-dimensional space, yields fixed-cost adaptation and shares efficiently information across environments.
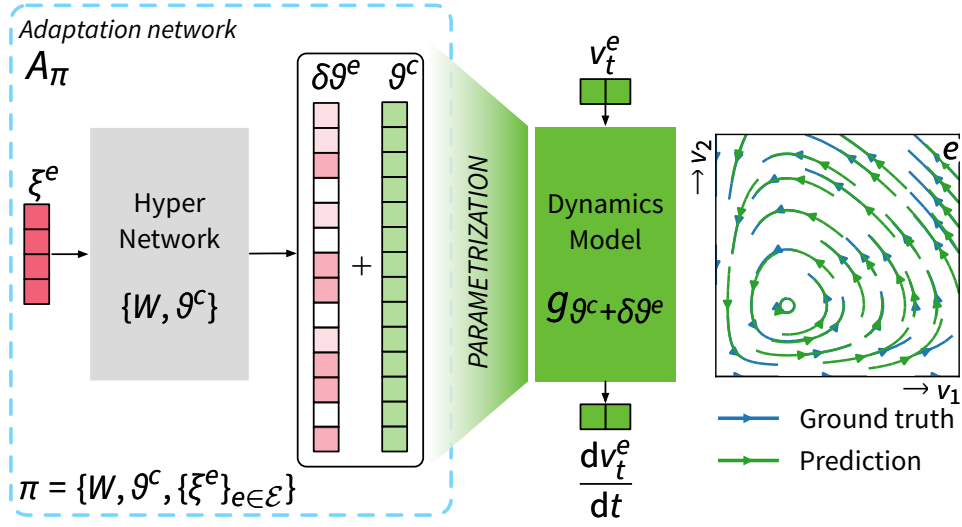
Figure 6.1. – Context-Informed Dynamics Adaptation (CoDA).

Formulation   We estimate $\delta\theta^e$ through a linear mapping of conditioning information, called context, learned from $\mathcal{D}^e$ and denoted $\xi^e \in \mathbb{R}^{d_\xi}$. $W = (W_1, \cdots, W_{d_\xi}) \in \mathbb{R}^{d_\theta \times d_\xi}$ is the weight matrix of the linear decoder s.t.

$$A_\pi(\mathcal{D}^e) \triangleq \theta^c + W\xi^e, \quad \pi \triangleq \{W, \theta^c, \{\xi^e\}_{e \in \mathcal{E}}\} \tag{6.6}$$

W is shared across environments and defines a low-dimensional subspace $\mathcal{W} \triangleq \mathrm{Span}(W_1, ..., W_{d_\xi})$, of dimension at most d_ξ, to which the search space of $\delta\theta^e$ is restricted. ξ^e is specific to each environment and can be interpreted as learning rates along the rows of W. In our experiments, $d_\xi \ll d_\theta$ is small, at most 2. Thus, *adaptation to new environments only requires to learn very few parameters, which define a completely new dynamics model g.*

A_π corresponds to an affine mapping of ξ^e parametrized by $\{W, \theta^c\}$, a.k.a. a linear hypernetwork. Note that hypernetworks (Ha et al. 2017) have been designed to handle single-environment problems and learn a separate context per layer. Our formalism involves multiple environments and defines a context per environment for all layers of g.

Linearity of the hypernetwork is not restrictive as contexts are directly learned through an inverse problem detailed in Equations (6.7) and (6.8), s.t. expressivity is similar to a nonlinear hypernetwork with a final linear activation.

Objectives   We derive the training and adaptation objectives by inserting Eq. (6.6) into Eq. (6.5). For training, both contexts and hypernetwork are learned with:

$$\min_{\theta^c, W, \{\xi^e\}_{e \in \mathcal{E}_{\mathrm{tr}}}} \sum_{e \in \mathcal{E}_{\mathrm{tr}}} \left(\mathcal{L}(\theta^c + W\xi^e, \mathcal{D}^e) + \lambda \|W\xi^e\|^2 \right) \tag{6.7}$$

After training, θ^c is kept fixed and for adaptation to a new environment, only the context vector ξ^e is learned with:

$$\min_{\{\xi^e\}_{e\in\mathcal{E}_{\text{ad}}}} \sum_{e\in\mathcal{E}_{\text{ad}}} \left(\mathcal{L}(\theta^c + W\xi^e, \mathcal{D}^e) + \lambda\|W\xi^e\|^2 \right) \tag{6.8}$$

Implementation of Equations (6.7) and (6.8) is detailed in Section 6.4. We apply gradient descent. In Proposition 6.1, we show for $\|\cdot\| = \ell_2$, that Eq. (6.8) admits an unique solution, recovered from initialization at $\mathbf{0}$ with a single preconditioned gradient step, projected onto subspace $\mathcal{W}$ defined by W.

Proposition 6.1 (Proof in Appendix D.2). *Given $\{\theta^c, W\}$ fixed, if $\|\cdot\| = \ell_2$, then Eq. (6.8) is quadratic. If $\lambda'W^\top W$ or $\bar{H}^e(\theta^c) = W^\top\nabla_\theta^2\mathcal{L}(\theta^c, \mathcal{D}^e)W$ are invertible then $\bar{H}^e(\theta^c) + \lambda'W^\top W$ is invertible except for a finite number of λ' values. The problem in Eq. (6.8) is then also convex and admits an unique solution, $\{\xi^{e\star}\}_{e\in\mathcal{E}_{ad}}$. With $\lambda' \triangleq 2\lambda$,*

$$\xi^{e*} = -\left(\bar{H}^e(\theta^c) + \lambda'W^\top W \right)^{-1} W^\top\nabla_\theta\mathcal{L}(\theta^c, \mathcal{D}^e) \tag{6.9}$$

Interpretation   We now interpret CoDA by visualizing its loss landscape in Figure 6.2a and comparing it to ERM's loss landscape in Figure 6.2b. We use the package in Li et al. (2018b) to plot loss landscapes around θ^c and consider the *Lotka-Volterra* system, described in Section 6.5.1.

In Figure 6.2a, loss values of CoDA are projected onto subspace $\mathcal{W}$, where $d_\xi = 2$. We make three observations. First, across environments, the loss is smooth and has a single minimum around θ^c. Second, the local optimum of the loss is close to θ^c across environments. Finally, the minimal loss value on $\mathcal{W}$ around θ^c is low across environments. The two first properties were discussed in Section 6.3.2 and are a direct consequence of the locality constraint on $\mathcal{W}$. When $\|\cdot\| = \ell_2$, it makes the optimization problem in Eq. (6.7) quadratic w.r.t. ξ^e and convex under invertibility of $\bar{H}^e(\theta^c) + \lambda'W^\top W$ as detailed in Proposition 6.1. We provided in Eq. (6.9) the closed form expression of the solution. It also imposes small $\|\xi^e\|$ s.t. when minimizing the loss in Eq. (6.7), θ^c remains close to local optimas of all training environments. The final observation illustrates that CoDA finds a subspace $\mathcal{W}$ with environment-specific parameters of low loss values *i.e.* low-rank adaptation performs well.

In Figure 6.2b, loss values of ERM are projected onto the span of the two principal gradient directions. We observe that, unlike CoDA, ERM does not find low loss values. Indeed, it aims at finding θ^c with good performance across environments, thus cannot model several dynamics.
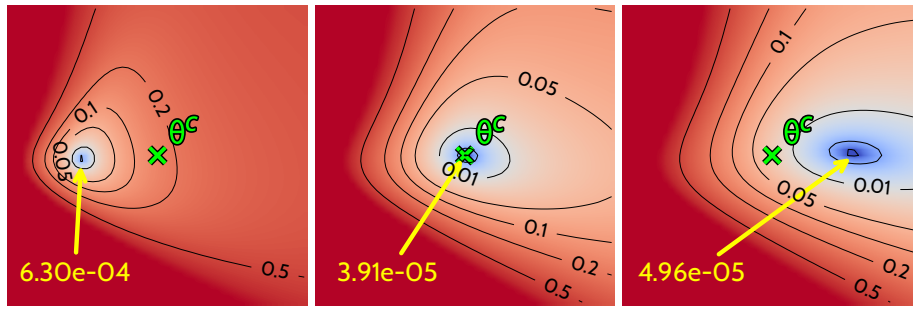
(a) CoDA's loss projected onto $\mathcal{W} = \mathrm{Span}(W_1, W_2)$.
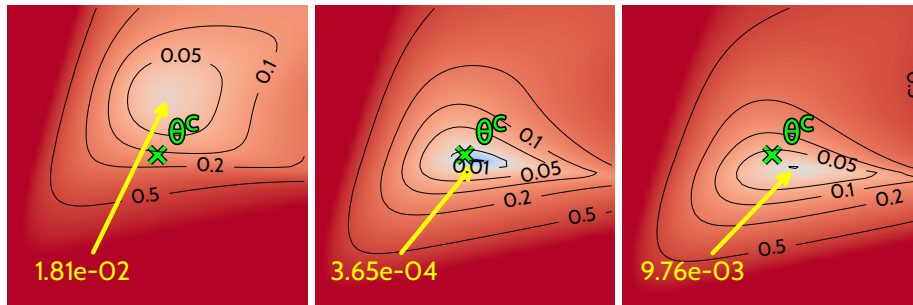
(b) ERM's loss projected onto the span of the two principal gradient directions computed with SVD.

Figure 6.2. – Loss landscapes centered in θ^c, marked with ×, for 3 environments on the *Lotka-Volterra* ODE. $\forall e, \rightarrow$ points to the local optimum $\theta^{e\star}$ with loss value reported in yellow.

6.3.4 Validity for Dynamical Systems

We further motivate low-rank decoding in our context-informed hypernetwork approach by providing some evidence that gradients at θ^c across environments define a low-dimensional subspace. We consider the loss $\mathcal{L}$ in Eq. (6.3) and define the gradient subspace in Definition 6.2.

Definition 6.2 (Gradient directions). With $\mathcal{L}$ in Eq. (6.3), $\forall \theta^c \in \mathbb{R}^{d_\theta}$ parametrizing a dynamics model g_{θ^c}, the subspace generated by gradient directions at θ^c across environments $\mathcal{E}$ is denoted $\mathcal{G}_{\theta^c} \triangleq \mathrm{Span}(\{\nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e)\}_{e \in \mathcal{E}})$.

We show, in Proposition 6.3, low-dimensionality of $\mathcal{G}_{\theta^c}$ for linearly parametrized systems.

Proposition 6.3 (Low-rank under linearity. Proof in Appendix D.2). *Given a class of linearly parametrized dynamics $\mathcal{F}$ with d_p varying parameters, $\forall \theta^c \in \mathbb{R}^{d_\theta}$, subspace $\mathcal{G}_{\theta^c}$ in Definition 6.2 is low-dimensional and $\dim(\mathcal{G}_{\theta^c}) \le d_p \ll d_\theta$.*

The linearity assumption is not restrictive as it is present in a wide variety of real-world systems *e.g.* Burger or Korteweg–De Vries PDE (Raissi et al. 2019),

convection-diffusion (Long et al. 2018b), wave and reaction diffusion equations (Yin et al. 2021b) *etc*.

Under nonlinearity, we do not have the same theoretical guarantee, yet, we show empirically in Appendix D.4 that low-dimensionality of parameters of the dynamics model still holds for several systems. This property is comforted by recent work that highlighted that gradients are low-rank throughout optimization in single-domain settings, *i.e.* that the solution space is low-dimensional (Gur-Ari et al. 2019; Li et al. 2018a; Li et al. 2018b). In the same spirit as CoDA, this property was leveraged to design efficient solutions to the learning problems (Frankle et al. 2019; Vogels et al. 2019).

6.3.5   Benefits of CoDA

We highlight the benefits of CoDA. CoDA is a general time-continuous framework that can be used with any approximator g_θ of the derivative Eq. (6.3). It can be trained with a given temporal resolution and tested on another; it handles irregularly-sampled sequences. The choice of the approximator g_θ defines the ability to handle different spatial resolutions for PDEs, as further detailed in Section 6.5.3.

Compared to related adaptation methods, CoDA presents several advantages. First, as detailed in Appendix D.1.1, the adaptation rule in Eq. (6.4) is similar to the one used in Gradient-Based Meta Learning (GBML); yet, our first order joint optimization problem in Eq. (6.5) simplifies the complex bi-level optimization problem (Antoniou et al. 2019). Second, CoDA introduces the two key properties of locality constraint and low-rank adaptation which guarantee efficient adaptation to new environments as discussed in Section 6.3.3. Third, it generalizes contextual meta-learning methods (Garnelo et al. 2018; Zintgraf et al. 2019), which also perform low-rank adaptation, via the hypernetwork decoder (details in Appendix D.1.2). Our decoder learns complex environment-conditional dynamics models while controlling their complexity. Finally, CoDA learns context vectors through an inverse problem as Zintgraf et al. (2019). This decoder-only strategy is particularly efficient and flexible in our setting. An alternative is to infer them via a learned encoder of $\mathcal{D}^e$ as Garnelo et al. (2018). Yet, the latter was observed to underfit (Kim et al. 2019a), requiring extensive tuning of the encoder and decoder architecture. Overall, CoDA is easy to implement and maintains expressivity with a linear decoder.

6.4   Framework Implementation

We detail how to perform trajectory-based learning with CoDA and describe two instantiations of the locality constraint. We detail the pseudo-code.

Trajectory-Based Formulation   As derivatives in Eq. (6.3) are not directly observed, we use in practice for training a trajectory-based formulation of Eq. (6.3). We consider a set of N trajectories, $\mathcal{D}^e$. Each trajectory is discretized over a uniform temporal grid $\mathcal{T}$ and spatial grid $\mathcal{X}$ for PDEs. Our loss writes as:

$$\mathcal{L}(\theta, \mathcal{D}^e) = \sum_{v^e \in \mathcal{D}^e} \sum_{x \in \mathcal{X}} \sum_{t \in \mathcal{T}} \|v_t^e(x) - \tilde{v}_t^e(x)\|_2^2 \text{ where } \tilde{v}_t^e = v_0^e + \int_{t_0}^{t} g_\theta(\tilde{v}_\tau^e)\mathrm{d}\tau \qquad (6.10)$$

$v_t^e(x)$ is the state value in the considered trajectory from environment e at spatial coordinate x and time t. $v_t^e \triangleq [v_t^e(x)]_{x \in \mathcal{X}}^\top$ is the state vector in the considered trajectory from environment e over the spatial domain $\mathcal{X}$ at time t and v_0^e is the corresponding initial condition. To compute $\tilde{v}_t^e$, we apply for integration a numerical solver (Hairer et al. 2000) as detailed later.

Locality   Instead of penalizing $\lambda\|W\xi^e\|^2$ in Eq. (6.7), we found it more efficient to penalize separately W and ξ^e. We thus use the following regularization:

$$R(W, \xi^e) \triangleq \lambda_\xi \|\xi^e\|_2^2 + \lambda_\Omega \Omega(W) \qquad (6.11)$$

It involves hyperparameters $\lambda_\xi, \lambda_\Omega$ and a norm $\Omega(W)$ which depends on the choice of $\|\cdot\|$ in Eq. (6.5). Minimizing $R(W, \xi^e)$ minimizes an upper-bound to $\|\cdot\|$, derived in Appendix D.5 for the two considered variations of $\|\cdot\|$:

- CoDA-ℓ_2 sets $\|\cdot\| \triangleq \ell_2(\cdot)$ and $\Omega \triangleq \ell_2^2$, constraining $W\xi^e$ to a sphere.

- CoDA-ℓ_1 sets $\|\cdot\| \triangleq \ell_1(\cdot)$ and $\Omega = \ell_{1,2}$ over rows *i.e.* $\Omega(W) \triangleq \sum_{i=1}^{d_\theta} \|W_{i,:}\|_2$ to induce sparsity and find most important parameters for adaptation. $\ell_{1,2}$ constrains $\mathcal{W}$ to be axis-aligned; then the number of solutions is finite as $\dim(\mathcal{W})$ is finite.

Pseudo-Code   We solve Eq. (6.7) for training and Eq. (6.8) for adaptation using Equations (6.10) and (6.11) and Algorithm 6.1. We back-propagate through the solver with torchdiffeq (Chen 2021) and apply exponential Scheduled Sampling (Bengio et al. 2015) to stabilize training.

Algorithm 6.1 CoDA Pseudo-code

1: *Training:*
Require: $\mathcal{E}_{\text{tr}} \subset \mathcal{E}$, $\{\mathcal{D}^{e_{\text{tr}}}\}_{e_{\text{tr}} \in \mathcal{E}_{\text{tr}}}$ with $\forall e_{\text{tr}} \in \mathcal{E}_{\text{tr}}, \#\mathcal{D}^{e_{\text{tr}}} = N_{\text{tr}}$;
2: $\pi = \{W, \theta^c, \{\xi^{e_{\text{tr}}}\}_{e_{\text{tr}} \in \mathcal{E}_{\text{tr}}}\}$ where $W \in \mathbb{R}^{d_\theta \times d_\xi}, \theta^c \in \mathbb{R}^{d_\theta}$ randomly initialized and $\forall e_{\text{tr}} \in \mathcal{E}_{\text{tr}}, \xi^{e_{\text{tr}}} = \mathbf{0} \in \mathbb{R}^{d_\xi}$.
3: **loop**
4:     $\pi \leftarrow \pi - \eta \nabla_\pi \Big(\sum\limits_{e_{\text{tr}} \in \mathcal{E}_{\text{tr}}} \mathcal{L}(\theta^c + W\xi^{e_{\text{tr}}}, \mathcal{D}^{e_{\text{tr}}}) + R(W, \xi^{e_{\text{tr}}}) \Big)$
5: **end loop**
6: *Adaptation:*
Require: $e_{\text{ad}} \in \mathcal{E}_{\text{ad}}$; $\mathcal{D}^{e_{\text{ad}}}$ with $\#\mathcal{D}^{e_{\text{ad}}} = N_{\text{ad}}$;
7: Trained $W \in \mathbb{R}^{d_\theta \times d_\xi}, \theta^c \in \mathbb{R}^{d_\theta}$ and $\xi^{e_{\text{ad}}} = \mathbf{0} \in \mathbb{R}^{d_\xi}$.
8: **loop**
9:     $\xi^{e_{\text{ad}}} \leftarrow \xi^{e_{\text{ad}}} - \eta \nabla_{\xi^{e_{\text{ad}}}} \Big(\mathcal{L}(\theta^c + W\xi^{e_{\text{ad}}}, \mathcal{D}^{e_{\text{ad}}}) + R(W, \xi^{e_{\text{ad}}}) \Big)$
10: **end loop**

Table 6.1. – Test MSE ($\downarrow$) in training environments $\mathcal{E}_{\text{tr}}$ (*In-Domain*), new environments $\mathcal{E}_{\text{ad}}$ (*Adaptation*). Best in **bold**; second underlined.

	LV ($\times 10^{-5}$)		GO ($\times 10^{-4}$)		GS ($\times 10^{-3}$)		NS ($\times 10^{-4}$)	
	In-domain	Adaptation	In-domain	Adaptation	In-domain	Adaptation	In-domain	Adaptation
MAML	60.3±1.3	3150±940	57.3±2.1	1081±62	3.67±0.53	2.25±0.39	68.0±8.0	51.1±4.0
ANIL	381±76	4570±2390	74.5±11.5	1688±226	5.01±0.80	3.95±0.11	61.7±4.3	48.6±3.2
Meta-SGD	32.7±12.6	7220±4580	42.3±6.9	1573±413	2.85±0.54	2.68±0.20	53.9±28.1	44.3±27.1
LEADS	3.70±0.27	47.61±12.47	31.4±3.3	113.8±41.5	2.90±0.76	1.36±0.43	14.0±1.55	28.6±7.23
CAVIA-FiLM	4.38±1.15	8.41±3.20	4.44±1.46	3.87±1.28	2.81±1.15	1.43±1.07	23.2±12.1	22.6±9.88
CAVIA-Concat	2.43±0.66	6.26±0.77	5.09±0.35	2.37±0.23	2.67±0.48	1.62±0.85	25.5±6.31	26.0±8.24
CoDA-ℓ_2	1.52±0.08	<u>1.82±0.24</u>	<u>2.45±0.38</u>	1.98±0.06	<u>1.01±0.15</u>	<u>0.77±0.10</u>	9.40±1.13	10.3±1.48
CoDA-ℓ_1	**1.35**±0.22	**1.24**±0.20	**2.20**±0.26	**1.86**±0.29	**0.90**±0.057	**0.74**±0.10	**8.35**±1.71	**9.65**±1.37

6.5   Experiments

We validate our approach on four classes of challenging nonlinear temporal and spatiotemporal physical dynamics, representative of various fields *e.g.* chemistry, biology and fluid dynamics. We evaluate in-domain and adaptation prediction performance and compare them to related baselines. We also investigate how learned context vectors can be used for system parameter estimation. We consider a few-shot adaptation setting where only few trajectories (N_{ad}) are available at adaptation time on new environments.

6.5.1   Dynamical Systems

We consider four ODEs and PDEs described in Appendix D.6.1. ODEs include *Lotka-Volterra* (LV, Lotka 1925) and *Glycolitic-Oscillator* (GO, Daniels et al. 2015), modelling respectively predator-prey interactions and the dynamics of yeast gly-

colysis. PDEs are defined over a 2D spatial domain and include *Gray-Scott* (GS, Pearson 1993), a reaction-diffusion system with complex spatiotemporal patterns and the challenging *Navier-Stokes* system (NS, Stokes 1851) for incompressible flows. All systems are nonlinear w.r.t. system states and all but GO are linearly parametrized. The analysis in Section 6.3.4 covers all systems but GO. Experiments on the latter show that CoDA also extends to nonlinearly parametrized systems.

6.5.2 Experimental Setting

We consider forecasting: only the initial condition is used for prediction. We perform two types of evaluation: in-domain generalization on $\mathcal{E}_{\mathrm{tr}}$ *(In-domain)* and out-of-domain adaptation to new environments $\mathcal{E}_{\mathrm{ad}}$ *(Adaptation)*. Each environment $e \in \mathcal{E}$ is defined by system parameters and $p^e \in \mathbb{R}^{d_p}$ denotes those that vary across $\mathcal{E}$. d_p represents the degrees of variations in $\mathcal{F}$; $d_p = 2$ for LV, GO, GS and $d_p = 1$ for NS. Appendix D.6.1 defines for each system the number of training and adaptation environments ($\#\mathcal{E}_{\mathrm{tr}}$ and $\#\mathcal{E}_{\mathrm{ad}}$) and the corresponding parameters. Appendix D.6.1 also reports the number of trajectories N_{tr} per training environment in $\mathcal{E}_{\mathrm{tr}}$ and the distribution $p(v_0)$ from which are sampled all initial conditions (including adaptation and evaluation initial conditions). For *Adaptation*, we consider $N_{\mathrm{ad}} = 1$ trajectory per new environment in $\mathcal{E}_{\mathrm{ad}}$ to infer the context vector with Eq. (6.8). We consider more trajectories per adaptation environment in Section 6.5.7.

Evaluation is performed on 32 new test trajectories per environment. We report, in our tables, mean and standard deviation of MSE across test trajectories (Eq. (6.10)) over four different seeds. We report, in our figures, Mean Absolute Percentage Error (MAPE) in % over trajectories, as it allows to better compare performance across environments and systems. We define $\mathrm{MAPE}(z, y)$ between a d-dimensional input z and target y as $\frac{1}{d} \sum_{j=1\ldots d:y_j \neq 0} \frac{|z_j - y_j|}{|y_j|}$. Over a trajectory, it extends into $\int_{t \in \mathcal{T}} \mathrm{MAPE}(\tilde{v}_t, v_t) \mathrm{d}t$, with $\tilde{v}$ defined in Eq. (6.10).

6.5.3 Implementation of CoDA

We used for g_θ MLPs for ODEs, a resolution-dependent ConvNet for GS and a resolution-agnostic FNO (Li et al. 2021c) for NS that can be used on new resolutions. Architecture details are provided in Appendix D.6.2. We tuned d_ξ and observed that $d_\xi = d_p$, the number of system parameters that vary across environments, performed best (*cf.* Section 6.5.6). We use Adam optimizer Kingma et al. 2015 for all datasets; RK4 solver for LV, GS, GO and Euler solver for NS. Optimization and regularization hyperparameters are detailed in Appendix D.6.2.

6.5.4   Baselines

We consider three families of baselines, compared in Appendix Figure D.1 and detailed in Section 2.1. First, GBML methods MAML (Finn et al. 2017), ANIL (Rusu et al. 2019) and Meta-SGD (Li et al. 2017b). Second, the Multi-Task Learning (MTL) method LEADS (Yin et al. 2021a). Finally, the contextual meta-learning method CAVIA (Zintgraf et al. 2019), with conditioning via concatenation (Concat) or linear modulation of final hidden features (FiLM, (Perez et al. 2018)). All baselines are adapted to be dynamics-aware *i.e.* time-continuous: they consider the loss in Eq. (6.10), as CoDA. Moreover, they share the same architecture for g_θ as CoDA.

6.5.5   Generalization Results

In Table 6.1, we observe that CoDA improves significantly test MSE w.r.t. our baselines for both *In-Domain* and *Adaptation* settings. For PDE systems and a given test trajectory, we visualize in Figures D.3 and D.4 in Appendix D.7 the predicted MSE by these models along the ground truth. We also notice improvements for CoDA over our baselines. Across datasets, all baselines are subject to a drop in performance between *In-Domain* and *Adaptation* while CoDA maintains remarkably the same level of performance in both cases. In more details, GBML methods (MAML, ANIL, Meta-SGD) overfit on training *In-Domain* data especially when data is scarce. This is the case for ODEs which include less system states for training than PDEs. LEADS performs better than GBML but overfits for *Adaptation* as it does not adapt efficiently. CAVIA-Concat/FiLM perform better than GBML and LEADS, as they leverage a context, but are less expressive than CoDA. Both variations of CoDA perform best as they combine the benefits of low-rank adaptation and locality constraint. CoDA-ℓ_1 is better than CoDA-ℓ_2 as it induces sparsity, further constraining the hypothesis space.

We evaluate in Figure 6.3 CoDA-ℓ_1 on LV for *Adaptation* over a wider range of adaptation environments ($\#\mathcal{E}_{\mathrm{ad}} = 51 \times 51 = 2601$). We report mean MAPE over $\mathcal{E}_{\mathrm{ad}}$ (top). We observe three regimes: inside the convex hull of training environments $\mathcal{E}_{\mathrm{tr}}$, MAPE is very low; outside the convex-hull, MAPE remains low in a neighborhood of $\mathcal{E}_{\mathrm{tr}}$; beyond this neighborhood, MAPE increases. CoDA thus generalizes efficiently in the neighborhood of training environments and degrades outside this neighborhood. We plot reconstructed phase space portraits (bottom) on four selected environments and observe that the learned solution (green) closely follows the target trajectories (blue).
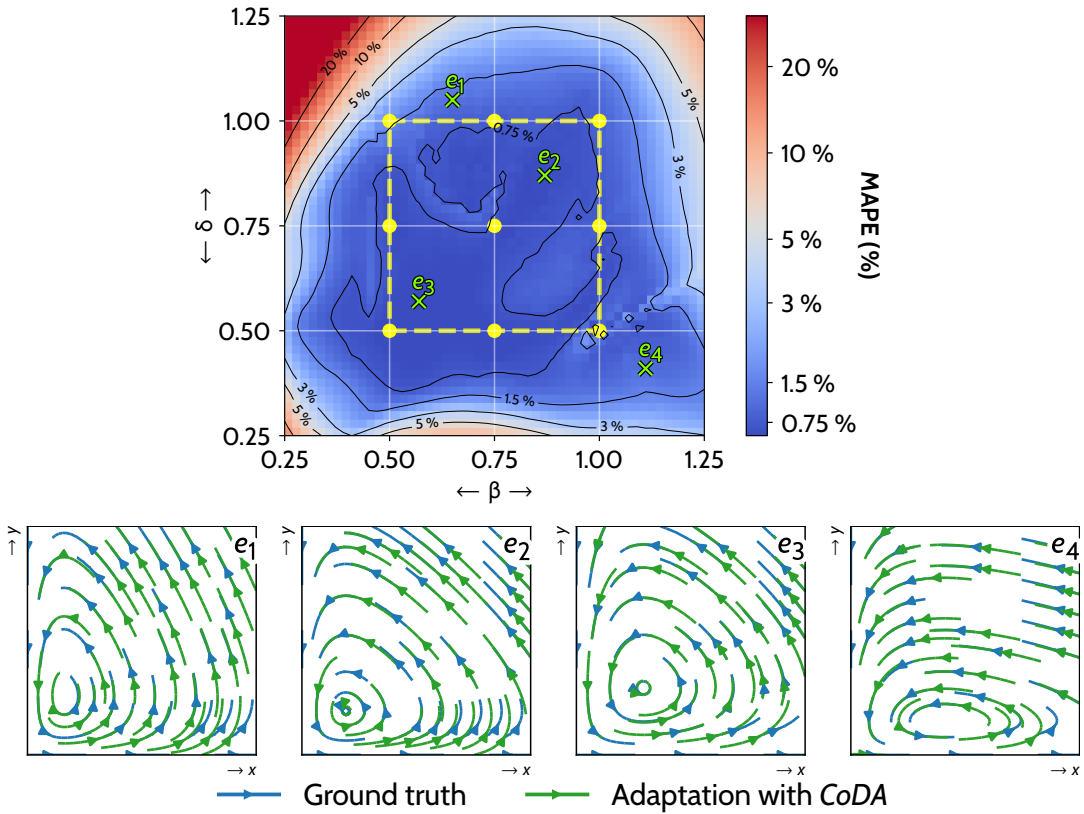
Figure 6.3. – *Adaptation* results with CoDA-ℓ_1 on LV. Parameters (β, δ) are sampled in $[0.25, 1.25]^2$ on a 51×51 uniform grid, leading to 2601 adaptation environments $\mathcal{E}_{\text{ad}}$. ● are training environments $\mathcal{E}_{\text{tr}}$. We report MAPE ($\downarrow$) across $\mathcal{E}_{\text{ad}}$ (top). On the bottom, we choose four of them (✕, e_1–e_4), to show the ground-truth (blue) and predicted (green) phase space portraits. x, y are respectively the quantity of prey and predator in the system in Eq. (D.4).

Table 6.2. – Locality and *In-Domain* test MSE ($\downarrow$). Best in **bold**.

CoDA	LV ($\times 10^{-5}$)		GO ($\times 10^{-4}$)	
	W/o ℓ_2	With ℓ_2	W/o ℓ_2	With ℓ_2
Full	2.28±0.29	1.52±0.08	2.98±0.71	2.45±0.38
FirstLayer	2.25±0.29	2.41±0.23	2.38±0.71	**2.12**±0.55
LastLayer	1.86±0.24	**1.27**±0.03	28.4±0.60	28.4±0.64

6.5.6 Ablation Studies

We perform two studies on LV and GO. In a first study in Table 6.2, we evaluate the gains due to using ℓ_2 locality constraint on *In-Domain* evaluation. On line 1 (Full), we observe that CoDA-ℓ_2 performs better than CoDA without locality constraint.
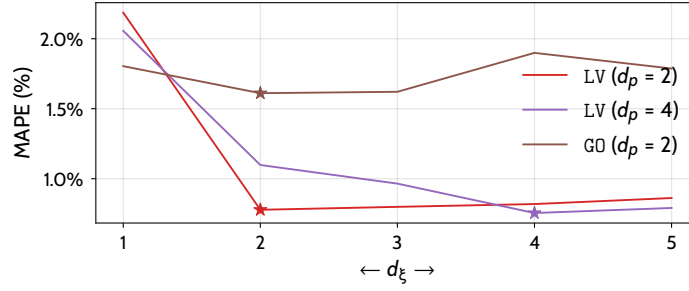
Figure 6.4. – Dimension of the context vectors (d_ξ) and test *In-Domain* MAPE ($\downarrow$) with CoDA-ℓ_1. "$\star$" is the smallest MAPE.

Prior work perform adaptation only on the final layer with some performance improvements on classification or Hamiltonian system modelling (Raghu et al. 2020; Chen et al. 2020a). In order to evaluate this strategy, we manually restrict hypernetwork-decoding to only one layer in the dynamics model g_θ, either the first layer (line 2) or the last layer (line 3). We observe that the importance of the layer depends on the parametrization of the system: for LV, linearly parametrized, the last layer is better while for GO, nonlinearly parametrized, the first layer is better. CoDA-ℓ_1 generalizes this idea by automatically selecting the useful adaptation subspace via $\ell_{1,2}$ regularization, offering a more flexible approach for sparsity.

In a second study in Figure 6.4, we analyze the impact on MAPE of the dimension of context vectors d_ξ for CoDA-ℓ_1. We recall that d_ξ upper-bounds the dimension of the adaptation subspace $\mathcal{W}$ and was cross-validated in Table 6.1. In the following, d_p is the number of parameters that vary across environments. We illustrate the effect of the cross-validation on MAPE for $d_p = 2$ on LV and GO as in Section 6.5.5 and additionally for $d_p = 4$ on LV. We observe in Figure 6.4 that the minimum of MAPE is reached for $d_\xi = d_p$ with two regimes: when $d_\xi < d_p$, performance decreases as some system dimensions cannot be learned; when $d_\xi > d_p$, performance degrades slightly as unnecessary directions of variations are added, increasing the hypothesis search space. This study shows the validity of the low-rank assumption and illustrates how the unknown d_p can be recovered through cross-validation.

6.5.7 Sample Efficiency

We handled originally one-shot adaptation ($N_{\mathrm{ad}} = 1$), the most challenging setting. We vary the number of adaptation trajectories N_{ad} on LV in Table 6.3. With more trajectories, performance improves significantly for MAML; moderately for LEADS; while it remains flat for CoDA. This highlights CoDA's sample-efficiency and meta-overfitting for GBML (Mishra et al. 2018).

Table 6.3. – Test MSE $\times 10^{-5}$ ($\downarrow$) in new environments $\mathcal{E}_\text{ad}$ (*Adaptation*) on *Lotka-Volterra*. Best for each setting in **bold**.

	Number of adaptation trajectories N_ad		
	1	5	10
MAML	3150±940	239±16	173±10
LEADS	47.61±12.47	19.89±7.23	19.42±3.52
CoDA-ℓ_1	**1.24**±0.20	**1.21**±0.18	**1.20**±0.17

Table 6.4. – Parameter estimation MAPE ($\downarrow$) for CoDA-ℓ_1 on LV ($\#\mathcal{E}_\text{tr} = 9$), GS ($\#\mathcal{E}_\text{tr} = 4$) and NS ($\#\mathcal{E}_\text{tr} = 5$).

	In-convex-hull		Out-of-convex-hull		Overall
	MAPE (%)	$\#\mathcal{E}_\text{ad}$	MAPE (%)	$\#\mathcal{E}_\text{ad}$	MAPE (%)
LV	0.15±0.11	625	0.73±1.33	1976	0.59±1.33
GS	0.37±0.25	625	0.74±0.67	1976	0.65±0.62
NS	0.10±0.08	40	0.51±0.35	41	0.30±0.33

6.5.8 Parameter Estimation

We use CoDA to perform parameter estimation, leveraging the links between learned context and system parameters.

6.5.9 Empirical observations

In Figure 6.5a (left), we visualize on LV the learned context vectors ξ^e (red) and the system parameters p^e (black), $\forall e \in \mathcal{E}_\text{tr} \cup \mathcal{E}_\text{ad}$. We observe empirically a linear bijection between these two sets of vectors. Such a correspondence being learned on the training environments, we can use the correspondence to verify if it still applies to new adaptation environments. Said otherwise, we can check if our model is able to infer the true parameters for new environments.

We evaluate in Table 6.4 the parameter estimation MAPE over LV, GS and NS. Figure 6.5 displays estimated parameters along estimation MAPE. Experimentally, we observe low MAPE inside and even outside the convex-hull of training environments. Thus, CoDA identifies accurately the unknown system parameters with little supervision.

(a) LV: $(\beta, \delta) \in [0.25, 1.25]^2$

(b) GS: $F \in$ [2.25, 4.35]e-2, $k \in [5.6, 6.4]$e-2

(c) NS: $\nu \in [0.6, 1.4]$e-3
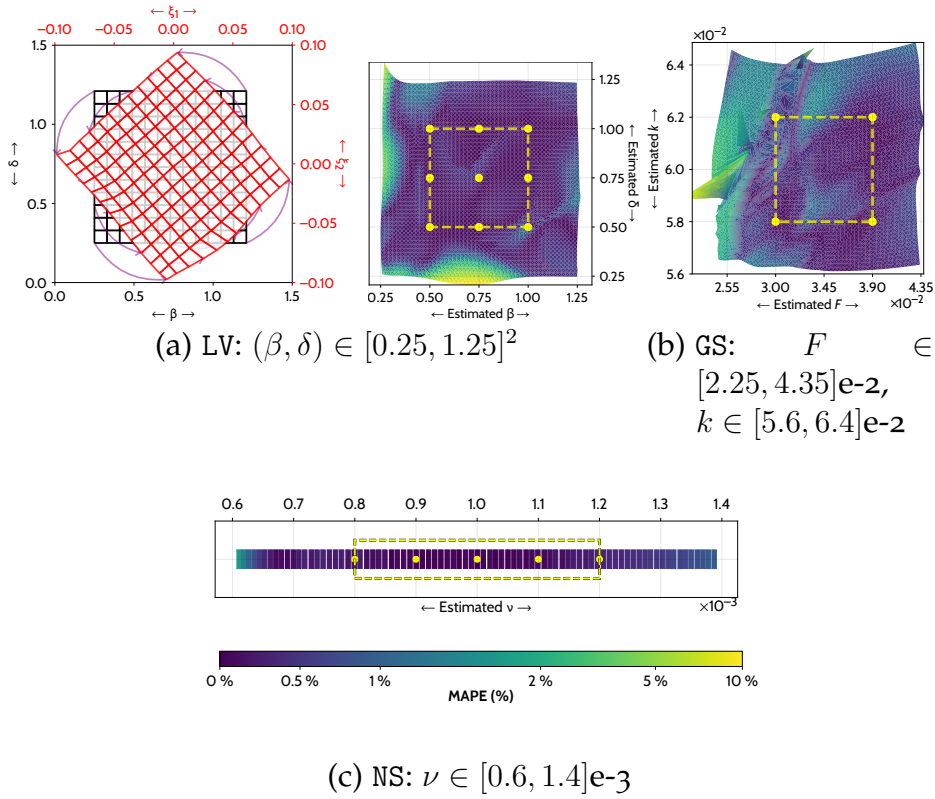
Figure 6.5. – Parameter estimation with CoDA-ℓ_1 in new adaptation environments on (a) LV, (b) GS and (c) NS. In (a), we visualize: on the left, context vectors ξ (red); on the right, true parameters (β, δ) (black). In (b) and (c), we visualize estimated parameters with corresponding estimation MAPE ($\downarrow$). • are training environments $\mathcal{E}_{tr}$ with known parameters. - - delimits the convex hull of $\mathcal{E}_{tr}$.

Theoretical motivation   We justify these empirical observations theoretically in Proposition 6.4 under the following conditions:

Assumption 12. The dynamics in $\mathcal{F}$ are linear w.r.t. inputs and system parameters.

Assumption 13. Dynamics model g, hypernet A are linear.

Assumption 14. $\forall e \in \mathcal{E}$, parameters $p^e \in \mathbb{R}^{d_p}$ are unique.

Assumption 15. Context vectors have dimension $d_\xi = d_p$.

Assumption 16. System parameters p of all dynamics f in a basis $\mathcal{B}$ of $\mathcal{F}$ are known.

Proposition 6.4 (Identification under linearity. Proof in Appendix D.3)**.** *Under Assumptions 12 to 16, system parameters are perfectly identified on new environments if the dynamics model g and hypernetwork A satisfy $\forall f \in \mathcal{B}$ with parameter p, $g_{A(p)} = f$.*

Intuitively, Proposition 6.4 says that given some observations representative of the degrees of variation of the data (a basis of $\mathcal{F}$) and given the system param-

eters for these observations (Assumption 16), we are guaranteed to recover the parameters of new environments for a family systems. This strong guarantee requires strong conditions. Assumptions 12 and 13 state that the systems should be linear w.r.t. inputs and that the dynamics model should be linear too. Linearity of the hypernetwork is not an issue as detailed in Section 6.3.3. Assumption 14 applies to several real-world systems used in our experiments (*cf.* Appendix D.3 Lemmas D.1 and D.2). Assumption 15 is not restrictive as we showed that d_p is recovered through cross-validation (Figure 6.4).

We propose an extension of Proposition 6.4 in Proposition 6.5 to nonlinear systems w.r.t. inputs and nonlinear dynamics model g. This alleviates the linearity assumption in Assumptions 12 and 13 and better fits our experimental setting.

Proposition 6.5 (Local identification under non-linearity. Proof in Appendix D.3)**.**
For linearly parametrized systems, nonlinear w.r.t. inputs and nonlinear dynamics model g_θ with parameters output by a linear hypernetwork A, $\exists \alpha > 0$ s.t. system parameters are perfectly identified $\forall e \in \mathcal{E}$ where $\|\xi^e\| \leq \alpha$ if $\forall f \in \mathcal{B}$ with parameter p, $g_{A(\alpha \frac{p}{\|p\|})} = f$.

Proposition 6.5 states that system parameters are recovered for environments with context vectors of small norm, under a rescaling condition on true system parameters. Proposition 6.5 explains why estimation error increases when system parameters differ greatly from training ones, as these systems are more likely to violate the norm condition.

6.6 Related Work

OOD methods extend the ERM objective to learn domain invariants *e.g.* via robust optimization (Sagawa et al. 2020) or Invariant Risk Minimization (IRM) (Arjovsky et al. 2019b) as reviewed in Section 2.1.3.3. However, they are not adapted to our problem as an unique model is learned. CoDA is closer to meta-learning and MTL. CoDA follows the same objective than contextual meta-learning methods (Zintgraf et al. 2019; Garnelo et al. 2018) of learning a low-dimensional representation of each task but generalizes these approaches with hypernetworks. MTL does not address adaptation to new tasks, which is the focus of CoDA, although some extensions have also considered this problem, mainly for classification (Wang et al. 2021a; Requeima et al. 2019). Only few work have considered adaptation for dynamical systems. LEADS (Yin et al. 2021a) is a MTL approach that performs adaptation in functional space. CoDA operates in parameter space, making adaptation more expressive and efficient, and scales better with the number of environments as it does not require training a full new network per environment as LEADS does.

6.7   Conclusion

We introduced CoDA, a new framework to learn context-informed data-driven dynamics models on multiple environments. CoDA generalizes with little retraining and few data to new related physical systems and outperforms prior methods on several real-world nonlinear dynamics. Many promising applications of CoDA are possible, notably for spatiotemporal problems, *e.g.* partially observed systems, reinforcement learning, or NN-based simulation.

Chapter 7

Spatiotemporal Generalization with Implicit Neural Representations

Chapter abstract

Effective data-driven PDE forecasting methods often rely on fixed spatial and / or temporal discretizations. This raises limitations in real-world applications like weather prediction where flexible extrapolation at arbitrary spatiotemporal locations is required. We address this problem by introducing a new data-driven approach, DINo, that models a PDE's flow with continuous-time dynamics of spatially continuous functions. This is achieved by embedding spatial observations independently of their discretization via Implicit Neural Representation (INR)s in a small latent space temporally driven by a learned ODE. This separate and flexible treatment of time and space makes DINo the first data-driven model to combine the following advantages. It extrapolates at arbitrary spatial and temporal locations; it can learn from sparse irregular grids or manifolds; at test time, it generalizes to new grids or resolutions. DINo outperforms alternative neural PDE forecasters in a variety of challenging generalization scenarios on representative PDE systems.

The work in this chapter has led to the publication of a conference paper:

- Y. Yin*, M. Kirchmeyer*, J-Y Franceschi*, A. Rakotomamonjy, and P. Gallinari (2023). "Continuous PDE Dynamics Forecasting with Implicit Neural Representations". In: *International Conference on Learning Representations (ICLR)*.

7.1 Introduction

Modeling the dynamics and predicting the temporal evolution of physical phenomena is paramount in many fields, *e.g.* climate modeling, biology, fluid mechan-

ics and energy (Willard et al. 2022). Classical solutions rely on a well-established physical paradigm: the evolution is described by differential equations derived from physical first principles, and then solved using numerical analysis tools, *e.g.* finite elements, finite volumes or spectral methods (Olver 2014). The availability of large amounts of data from observations or simulations has motivated data-driven approaches to this problem (Brunton et al. 2022), leading to a rapid development of the field with deep learning.

The main motivations for this research track include developing surrogate or reduced order models that can approximate high-fidelity full order models at reduced computational costs (Kochkov et al. 2021), complementing classical solvers, *e.g.* to account for additional components of the dynamics (Yin et al. 2021b), or improving low fidelity models (De Avila Belbute-Peres et al. 2020).

Most of these attempts rely on workhorses of deep learning like Convolutional Neural Network (CNN)s (Ayed et al. 2020) or Graph Neural Network (GNN)s (Li et al. 2020b; Pfaff et al. 2021; Brandstetter et al. 2022). They all require prior space discretization either on regular or irregular grids, such that they only capture the dynamics on the train grid and cannot generalize outside it. Neural operators, a recent trend, learn mappings between function spaces (Li et al. 2021c; Lu et al. 2021) and thus alleviate some limitations of prior discretization approaches. Yet, they still rely on fixed grid discretization for training and inference: *e.g.*, regular grids for Li et al. 2021c or a free-form but predetermined grid for Lu et al. 2021. Hence, the number and / or location of the sensors has to be fixed across train and test which is restrictive in many situations (Prasthofer et al. 2022). Mesh-agnostic approaches for solving canonical PDEs are another trend (Raissi et al. 2019; Sirignano et al. 2018). In contrast to physics-agnostic grid-based approaches, they aim at solving a known PDE as usual solvers do, and cannot cope with unknown dynamics. This idea was concurrently developed for computer graphics, *e.g.* for learning 3D shapes (Sitzmann et al. 2020; Mildenhall et al. 2020; Tancik et al. 2020) and coined as INRs. When used as solvers, these methods can only tackle a single initial value problem and are not designed for long-term forecasting outside the training horizon.

Because of these limitations, none of the above approaches can handle situations encountered in many practical applications such as: different geometries, *e.g.* phenomena lying on a Euclidean plane or an Earth-like sphere; variable sampling, *e.g.* irregular observation grids that may evolve at train and test time as in adaptive meshing (Berger et al. 1984); scarce training data, *e.g.* when observations are only available at a few spatiotemporal locations; multi-scale phenomena, *e.g.* in large scale-dynamics systems as climate modeling, where integrating intertwinned sub-grid scales a.k.a. the closure problem is ubiquitous (Zanna et al. 2021). These

Table 7.1. – Data-driven approaches to spatiotemporal PDE forecasting.

Model		Ref.	Space			Time		7. PDE agnostic prediction on new initial conditions
			1. Train / test grid independence	2. Free-form grid and topology	4. Evaluation at unobserved locations	5. Time continuous	6. Time extrapolation	
Discretized	NODE	Chen et al. 2018	✗	✗	✗	✓	✓	✓
	MP-PDE	Brandstetter et al. 2022	✗	✓	✗	✗	✓	✓
NO	MNO	Li et al. 2021c	✓	✗	✗	✗	✓	✓
	DeepONet	Lu et al. 2021	✗	✓	✓	✓	✗	✓
INRs	SIREN	Sitzmann et al. 2020	✓	✓	✓	✓	✗	✗
	DINo	Ours	✓	✓	✓	✓	✓	✓

considerations motivate the development of new machine learning models that improve existing approaches on several of these aspects.

In our work, we aim at forecasting PDE-based spatiotemporal physical processes with a versatile model tackling the aforementioned limitations. We adopt an agnostic approach, *i.e.* not assuming any prior knowledge on the physics. We introduce DINo (Dynamics-aware Implicit Neural representations), a model operating continuously in space and time, with the following contributions.

CONTINUOUS FLOW LEARNING. DINo aims at learning the PDE's flow to forecast its solutions, in a continuous manner so that it can be trained on any spatial and temporal discretization and applied to another. To this end, DINo embeds spatial observations into a small latent space via INRs; then it models continuous-time evolution by a learned latent ODE.

SPACE-TIME SEPARATION. To efficiently encode different sequences, we propose a novel INR parameterization, amplitude modulation, implementing a space-time separation of variables. This simplifies the learned dynamics, reduces the number of parameters and greatly improves performance.

SPATIOTEMPORAL VERSATILITY. DINo combines the benefits of prior models (Table 7.1). It handles new sequences via amplitude modulation. Sequential modeling with an ODE makes it extrapolate to unseen times within or beyond the train horizon. With INRs' spatial flexibility, it generalizes to new grids or resolutions, predicts at arbitrary positions and handles sparse irregular grids or manifolds.

EMPIRICAL VALIDATION. We demonstrate DINo's versatility and state-of-the-art performance v.s. prior neural PDE forecasters, representative of grid-based, operator and INR-based methods, via thorough experiments on challenging multi-dimensional PDEs in various spatiotemporal generalization settings.
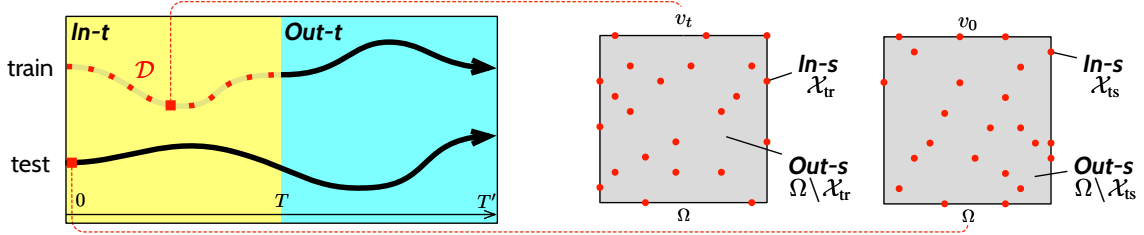
Figure 7.1. – (Left) We represent time contexts. The *train* trajectory consists of training snapshots (■), observed in a train interval $[0, T]$ denoted *In-t*. The line (—) in continuation is a forecasting of this trajectory beyond *In-t*, in $(T, T']$ denoted *Out-t*. The line below (—, *test*) is a forecasting from a new initial condition v_0 (■) on *In-t* and *Out-t*. (Middle and right) We illustrate spatial contexts. (Middle) Dots (•) correspond to the train observation grid $\mathcal{X}_{\text{tr}}$, denoted *In-s*. *Out-s* denotes the complementary domain $\Omega \setminus \mathcal{X}_{\text{tr}}$. (Right) New test observation grid $\mathcal{X}_{\text{ts}}$, used as an initial point for forecasting (left).

7.2 Problem description

Problem setting.   We aim at modeling, via a data-driven approach, the temporal evolution of a continuous fully-observed spatiotemporal phenomenon. It is described by trajectories $v\colon \mathbb{R} \to \mathcal{V}$ in a set Γ; we use $v_t \triangleq v(t) \in \mathcal{V}$. Trajectories share the same dynamics but differ by their initial condition $v_0 \in \mathcal{V}$. $\mathbb{R}$ is the temporal domain and $\mathcal{V}$ is the functional space of the form $\Omega \to \mathbb{R}^n$, where $\Omega \subset \mathbb{R}^p$ is a compact domain of spatial coordinates and n the number of observed values. In other words, v_t is a spatial function of $x \in \Omega$, with vectorial output $v_t(x) \in \mathbb{R}^n$; *cf.* examples of Section 7.4.1. To this end, we consider the setting illustrated in Figure 7.1. We observe a training set of trajectories $\mathcal{D}$, with a free-form spatial observation grid $\mathcal{X}_{\text{tr}} \subset \Omega$ and on discrete times $t \in \mathcal{T} \subset [0, T]$. At test time, we are only given a new initial condition v_0, with observed values $v_{0|\mathcal{X}_{\text{ts}}}$ restricted to a new observation grid $\mathcal{X}_{\text{ts}}$, potentially different from $\mathcal{X}_{\text{tr}}$. Inference is performed on both train and test trajectories given only the initial condition, on a new free-formed grid $\mathcal{X}' \subset \Omega$ and times $t \in \mathcal{T}' \subset [0, T']$. Inference grid $\mathcal{X}'$ comprises observed positions (respectively $\mathcal{X}_{\text{tr}}$ and $\mathcal{X}_{\text{ts}}$ for train and test trajectories) and unobserved positions corresponding to spatial extrapolation. The inference temporal horizon is larger than the train one: $T < T'$. For simplicity, *In-s* refers to data in $\mathcal{X}'$ on the observation grid ($\mathcal{X}_{\text{tr}}$ for *train* / $\mathcal{X}_{\text{ts}}$ for *test*), *Out-s* to data in $\mathcal{X}'$ outside the observation grid; *In-t* refers to times within the train horizon $\mathcal{T} \subset [0, T]$, and *Out-t* to times in $\mathcal{T}' \setminus \mathcal{T} \subset (T, T']$, beyond T, up to inference horizon T'.
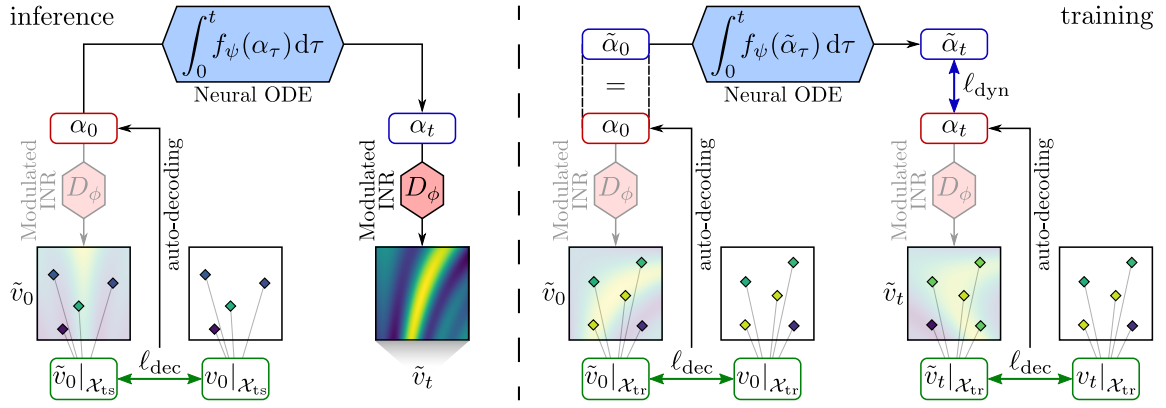
Figure 7.2. – Proposed DINo model. Inference (left): given a new initial condition observed on a grid $\mathcal{X}_{ts}$, $v_{0|\mathcal{X}_{ts}}$, forecasting amounts at decoding α_t to $\tilde{v}_t$, by unrolling α_0 with a time-continuous ODE dynamics model f_ψ. Train (right): given an observation grid $\mathcal{X}_{tr}$ and a space-continuous decoder D_ϕ, α_t is learned by auto-decoding s.t. $D_\phi(\alpha_t)|_{\mathcal{X}_{tr}} = v_t|_{\mathcal{X}_{tr}}$. Its evolution is then modelled with f_ψ.

Evaluation scenarios.   The desired properties in Section 7.1 call for spatiotemporally continuous forecasting models. We select six criteria that our approach should meet; *cf.* column titles of Table 7.1. First, the model should be robust to the change of initial condition v_0, *i.e.* generalize to *test* trajectories (col. 1). Second, it should extrapolate beyond the train conditions: in space, on a test observation grid that differs from the train one, *i.e.* $\mathcal{X}' = \mathcal{X}_{ts} \neq \mathcal{X}_{tr}$ (*In-s*) (col. 2), and outside the observed train and test grid, *i.e.* on $\mathcal{X}' \setminus \mathcal{X}_{ts}, \mathcal{X}' \setminus \mathcal{X}_{tr}$ (*Out-s*, col. 3); in time, between train snapshots (col. 5) and beyond the observed train horizon T (*Out-t*, col. 6). Finally, it should adapt to free-form spatial domains, *i.e.* to various geometries (*e.g.* manifolds) or irregular grids (col. 4). See also Figure 7.1.

Objective.   To satisfy these requirements, we learn the system's flow Φ:

$$\Phi : (\mathcal{V} \times \mathbb{R}) \to \mathcal{V}, \qquad (v_t, \tau) \mapsto \Phi_\tau(v_t) = v_{t+\tau} \quad \forall v \in \Gamma, t \in \mathbb{R}. \qquad (7.1)$$

Learning the flow is a common strategy in sequential models to better generalize beyond the train time horizon. Yet, so far, it has always been learned with discretized models, which poses generalization issues violating our requirements. We describe these issues in Section 2.2.2.

7.3   Model

We present DINo, the first space / time-continuous model that tackles all prediction tasks of Section 7.2, without the above limitations. We specify DINo's

inference procedure (Section 7.3.1), illustrated in Figure 7.2 (left), then introduce each of its components (Section 7.3.2) and how they are trained (Section 7.3.3, Figure 7.2 (right)). Finally, we detail our implementation based on amplitude modulation, a novel INR parameterization for spatiotemporal data which performs separation of variables (Section 7.3.4).

7.3.1   Inference model

As explained in Section 7.2, we aim at estimating the flow Φ in Eq. (7.1), so that our model can be trained on an observed grid $\mathcal{X}_{\mathrm{tr}}$ and perform inference given a new one $\mathcal{X}_{\mathrm{ts}}$, both possibly irregular. To this end, we leverage a space- and time-continuous formulation, independent of a given data discretization.

At inference, DINo starts from an initial condition $v_0 \in \mathcal{V}$ and uses a flow to forecast its dynamics. DINo first embeds spatial observations from v_0 into a latent vector α_0 of small dimension d_α via an encoder of spatial functions $E_\varphi \colon \mathcal{V} \to \mathbb{R}^{d_\alpha}$ (ENC). Then, it unrolls a latent time-continuous dynamics model $f_\psi \colon \mathbb{R}^{d_\alpha} \to \mathbb{R}^{d_\alpha}$ given v_0 (DYN). Finally, it decodes latent vectors via a decoder $D_\phi \colon \mathbb{R}^{d_\alpha} \to \mathcal{V}$ into a function of space (DEC). At any time t, D_ϕ takes as input α_t and outputs a function $\tilde{v}_t \colon \Omega \to \mathbb{R}^n$. This results in the following model (Figure 7.2 left):

$$(\text{ENC})\ \alpha_0 = E_\varphi(v_0), \qquad (\text{DYN})\ \tfrac{\mathrm{d}\alpha_t}{\mathrm{d}t} = f_\psi(\alpha_t), \qquad (\text{DEC})\ \forall t, \tilde{v}_t = D_\phi(\alpha_t). \qquad (7.2)$$

7.3.2   Components

We now further detail each component involved at inference from Eq. (7.2).

Encoder: $\alpha_t = E_\varphi(v_t)$.   The encoder computes a latent vector α_t given observation v_t at any time t. It is used in two different contexts, respectively for train and test. At train time, given an observed trajectory $v_\mathcal{T} = \{v_t\}_{t \in \mathcal{T}}$, it will encode any v_t into α_t (see Section 7.3.3). At inference time, only v_0 is available, and then only α_0 is computed to be used as initial value for the dynamics. Given the decoder D_ϕ, α_t is a solution to the inverse problem $D_\phi(\alpha_t) = v_t$. We solve this inverse problem with auto-decoding (Park et al. 2019). Denoting $\ell_{\mathrm{dec}}(\phi, \alpha_t; v_t) = \|D_\phi(\alpha_t) - v_t\|_2^2$ the decoding loss where $\|\cdot\|_2$ is the euclidean norm of a function and K the number of update steps, auto-decoding defines E_φ as:

$$E_\varphi(v_t) = \alpha_t^K, \ \text{ where } \forall k > 1, \alpha_t^k = \alpha_t^{k-1} - \eta \nabla_{\alpha_t} \ell_{\mathrm{dec}}(\phi, \alpha_t^{k-1}; v_t) \text{ and } \varphi = \phi. \qquad (7.3)$$

In practice, we observe a discretization $(\mathcal{X}_{\text{tr}}, \mathcal{X}_{\text{ts}})$ and accordingly approximate the norm in ℓ_{dec} as in Eq. (7.6). Compared to auto-encoding, auto-decoding underfits less (Kim et al. 2019b) and is more flexible: without requiring specialized encoder architecture, it handles free-formed (irregular or on a manifold) observation grids as long as the decoder shares the same property.

Decoder: $\tilde{v}_t = D_\phi(\alpha_t)$. We define a flexible decoder using a coordinate-based INR network with parameters conditioned on α_t. An INR $I\colon \mathbb{R}^{d_\theta} \to (\Omega \to \mathbb{R}^n)$ is a space-continuous model parameterized by $\theta \in \mathbb{R}^{d_\theta}$ which outputs a spatial function I_θ defined on domain Ω. It approximates functions independently of the observation grid, *e.g.* it handles irregular grids and changing observation positions unlike FNO and DeepONet. Thus, it constitutes a flexible alternative to operators suitable to auto-decoding. To implement the conditioning of the INR's parameters, we use a hypernetwork (Ha et al. 2017) $h_\phi\colon \mathbb{R}^{d_\alpha} \to \mathbb{R}^{d_\theta}$, as illustrated in Figure 7.3. It generates high-dimensional parameters $\theta_t \in \mathbb{R}^{d_\theta}$ of the INR given the low-dimensional latent vector $\alpha_t \in \mathbb{R}^{d_\alpha}$. In summary, the decoder D_ϕ, parameterized as h by ϕ, is defined as:

Figure 7.3. – Decoding via INR Eq. (7.4)

$$\forall x \in \Omega, \quad \tilde{v}_t(x) = D_\phi(\alpha_t)(x) \triangleq I_{h_\phi(\alpha_t)}(x). \qquad (7.4)$$

We provide further details on the precise implementation in Section 7.3.4.

Dynamics model: $\frac{d\alpha_t}{dt} = f_\psi(\alpha_t)$. Finally, the dynamics model $f_\psi\colon \mathbb{R}^{d_\alpha} \to \mathbb{R}^{d_\alpha}$ defines a flow via an ODE in the latent space. The initial condition can be defined at any time t by encoding with E_φ the corresponding input function v_t.

Overall flow. Combined altogether, our components define the following flow in the input space that can approximate the data flow Φ in Eq. (7.1):

$$\forall (t, \tau), \qquad (v_t, \tau) \mapsto D_\phi(E_\varphi(v_t) + \int_t^{t+\tau} f_\psi(\alpha_{\tau'}) d\tau') \quad \text{where } \alpha_t = E_\varphi(v_t). \qquad (7.5)$$

To summarize, DINo defines a time-continuous latent temporal model with a space-continuous emission function D_ϕ, combining the flexibility of space and time continuity. This is fully novel to our knowledge, as prior latent approaches are discretized (*cf.* Fraccaro 2018 for state-space models).

7.3.3   Training

Given these three components (ENC), (DEC), (DYN), we now present their training procedure, illustrated in Figure 7.2 (right). We use a fast and simple two-stage optimization, close to recent works in video prediction (Yan et al. 2021), and provide implementation details in Appendix E.4. Given the train sequences $\mathcal{D}$, we first apply auto-decoding across times to obtain the corresponding latent vectors $\alpha_\mathcal{T} = \{\alpha_t^v\}_{t\in\mathcal{T}, v\in\mathcal{D}}$, as well as the decoder parameters ϕ. We then learn the parameters of the dynamics model ψ by modeling the latent flow over α_t^v for each $v \in \mathcal{D}$. We detail this procedure in Appendix E.4.1, which can be formalized as a bi-level optimization problem solved in parallel:

$$
\begin{aligned}
\min_\psi \quad & \ell_{\text{dyn}}(\psi, \alpha_\mathcal{T}) \triangleq \mathbb{E}_{v\in\mathcal{D}, t\in\mathcal{T}} \| \alpha_t^v - (\alpha_0^v + \textstyle\int_0^t f_\psi(\alpha_\tau^v) d\tau) \|_2^2 \\
\text{s.t. } \alpha_\mathcal{T}, \phi = \arg\min_{\alpha_\mathcal{T}, \phi} \quad & \ell_{\text{dec}}(\phi, \alpha_\mathcal{T}) \triangleq \mathbb{E}_{v\in\mathcal{D}, x\in\mathcal{X}_{\text{tr}}, t\in\mathcal{T}} \| v_t(x) - D_\phi(\alpha_t^v)(x) \|_2^2
\end{aligned}
. \quad (7.6)
$$

7.3.4   Decoder implementation via amplitude-modulated INRs

We now specify our implementation of decoder D_ϕ in Eq. (7.4). This includes the definition of the INR architecture I_θ and of the hypernetwork h_ϕ. We introduce for the latter a new method called amplitude modulation, which implements a space-time separation of variables.

I_θ **as FourierNet.**   We implement I_θ as a FourierNet, a state-of-the-art INR architecture, which instantiates a Multiplicative Filter Network (MFN, Fathony et al. 2021). A FourierNet relies on the recursion in Eq. (7.7), where $x \in \Omega$ is an input spatial location, $z^{(l)}(x)$ is the hidden feature vector at layer l for x and $s_{\omega^{(l)}}(x) = [\cos(\omega^{(l)}x), \sin(\omega^{(l)}x)]$ is a Fourier basis:

$$
\begin{cases}
z^{(0)}(x) = s_{\omega^{(0)}}(x) \\
z^{(l)}(x) = \big(W^{(l-1)} z^{(l-1)}(x) + b^{(l-1)}\big) \odot s_{\omega^{(l)}}(x) & \text{for } l \in [\![1, L-1]\!], \\
z^{(L)}(x) = W^{(L-1)} z^{(L-1)}(x) + b^{(L-1)}
\end{cases}
\quad (7.7)
$$

where we fix $W^{(0)} = 0$, $b^{(0)} = 1$, $s_{\omega^{(0)}}(x) = x$. Denoting $W = [W^{(l)}]_{l=1}^{L-1}, b = [b^{(l)}]_{l=1}^{L-1}, \omega = [\omega^{(l)}]_{l=1}^{L-1}$, we fit a FourierNet to an input function v observed on a grid $\mathcal{X}$ by learning $\{W, b, \omega\}$ s.t. $\forall x \in \mathcal{X}, z^{(L)}(x) = v(x)$. In practice, we observe that fixing ω uniformly sampled performs similarly to learning them, so we exclude them from training parameters.

FourierNets are interpretable, a property we leverage to separate time and space via amplitude modulation. Fathony et al. 2021 show that $\exists M \gg L \in \mathbb{N}, \exists \{c_j^{(m)}\}_{m=1}^{M}$

a set of coefficients that depend individually on $\{W, b\}$ and $\exists \{\gamma^{(m)}\}_{m=1}^{M}$ a set of parameters that depend individually on those of the filters ω s.t. the jth dimension of $z^{(L)}(x)$ can be expressed as:

$$z_j^{(L)}(x) = \sum_{m=1}^{M} c_j^{(m)} s_{\gamma^{(m)}}(x) + \text{bias} \tag{7.8}$$

Eq. (7.8) involves a basis of spatial functions $\{s_{\gamma^{(m)}}\}_{m=1}^{M}$ evaluated on x and the amplitudes of this basis $\{c_j^{(m)}\}_{m=1}^{M}$. Note that Eq. (7.8) can be extended to other choices of $s_{\omega^{(l)}}$ (Fathony et al. 2021).

h as amplitude modulation. h generates the INR's parameters θ_t given α_t to model a target input function v_t. We propose to implement h as elementwise shift and scale transformations (FiLM, Perez et al. 2018) of the linear layers parameters W, b (excluding those of the filters ω). Then, in Eq. (7.8), amplitudes $c_j^{(m)}$ only depend on time while the basis functions $s_{\gamma^{(m)}}$ only depend on space: this corresponds to separation of variable (Le Dret et al. 2016). We call our technique amplitude modulation. In practice, as Dupont et al. 2022, we consider latent shift transformations as illustrated in Figure 7.4 and detailed in Eq. (7.9). Eq. (7.9) extends Eq. (7.7) by introducing a shift term, $\mu_t^{(l-1)} = W'^{(l-1)}\alpha_t$ at each layer l, where $W' = [W'^{(l-1)}]_{l=1}^{L-1}$ is another weight matrix:

Figure 7.4. – Amplitude modulation - Eq. (7.9). Input $z_t^{(l-1)}$ to layer l is combined with $s_{\omega^{(l)}}$ via Hadamard product.

$$z_t^{(l)}(x) = \left(W^{(l-1)} z_t^{(l-1)}(x) + b^{(l-1)} + \mu_t^{(l-1)}\right) \odot s_{\omega^{(l)}}(x). \tag{7.9}$$

The INR's parameters are defined as $h_\phi(\alpha_t) = \{W; b + W'\alpha_t; \omega\}$ where $\phi = \{W, b, W'\}$ are h's parameters. Thus, amplitude modulation separates time and space. We show in Table E.1 that it significantly improves performance, particularly time extrapolation.

7.4   Experiments

We assess the spatiotemporal versality of DINo, following Section 7.2. We introduce our experimental setting (Section 7.4.1), which includes a variety of chal-

lenging PDE datasets, state-of-the-art baselines and forecasting tasks. Then, we present and comment the experimental results (Section 7.4.2).

7.4.1   Experimental setting

Datasets.   We consider the following PDEs defined over a spatial domain Ω, with further details in Appendix E.3. • **2D Wave equation** (*Wave*) is a second-order PDE $\frac{d^2 u_t}{dt^2} = c^2 \Delta u_t$. u is the displacement w.r.t. the rest position and c is the wave traveling speed. We consider its first-order form, so that $v_t = (u_t, \frac{du_t}{dt})$ has a two-dimensional output ($n = 2$). • **2D Navier Stokes** (*Navier-Stokes*, Stokes 1851) corresponds to an incompressible fluid dynamics $\frac{dv_t}{dt} = -u\nabla v + \nu\Delta v + f$, $v = \nabla \times u$, $\nabla u = 0$, where u is the velocity field and v the vorticity. ν is the viscosity and f is a constant forcing term; $n = 1$. • **3D Spherical shallow water** (*Shallow-Water*, Galewsky et al. 2004): it involves the vorticity w, tangent to the sphere's surface, and the thickness of the fluid h. The input is $v_t = (w_t, h_t)$ ($n = 2$).

Baselines.   We reimplement representative neural PDE forecasters from Table 7.1 and adapt them to our multi-dimensional datasets. • **CNODE** (Ayed et al. 2020) combines a CNN and an ODE solver to handle regular grids. • **MP-PDE** (Brandstetter et al. 2022) uses a GNN to handle free-formed grids, yet is unable to predict outside the observation grid. We developed an interpolative extension, **I-MP-PDE**, to handle this limitation; it performs bicubic interpolation on the observed grid and training is done on the resulting interpolation. • **MNO** (Li et al. 2021b): an autoregressive version of FNO (Li et al. 2021c) for regular grids; MNO can be evaluated on new uniform grids. • **DeepONet** (Lu et al. 2021), considered autoregressively (Wang et al. 2021c) where we remove time from the trunk net's input. DeepONet can be evaluated on new spatial locations without interpolation. • **SIREN** (Sitzmann et al. 2020) and **MFN** (Fathony et al. 2021) are two INR methods which we extend to fit our setting. We consider an agnostic setting, *i.e.* without the knowledge of the differential equation and perform sequence conditioning to generalize to more than a trajectory. This is achieved by learning a latent vector with auto-decoding; it is then concatenated to the spatial coordinates.

Tasks.   We evaluate models on various forecasting tasks which combine the evaluation scenarios of Section 7.2. Performance is measured by the prediction MSE given only an initial condition. • **Space and time extrapolation.** We consider a uniform grid $\mathcal{X}'$ for inference. Training is performed on different observations grids $\mathcal{X}_{tr}$ subsampled from $\mathcal{X}'$ with different ratios, $s \in \{5\%, 25\%, 50\%, 100\%\}$ where $s = 100\%$ corresponds to the full inference grid, *i.e.* $\mathcal{X}_{tr} = \mathcal{X}'$. In this setting, we consider that all trajectories (*train* and *test*) share the same observation grid

Table 7.2. – **Space and time extrapolation.** Train and test observation grids are equal and subsampled from an uniform 64×64 grid, used for inference. We report MSE (↓) on the inference time interval $\mathcal{T}'$, divided within training horizon (*In-t*, $\mathcal{T}$) and beyond (*Out-t*, outside $\mathcal{T}$) across subsampling ratios.

| | | Navier-Stokes | | | | Wave | | | |
| | | Train | | Test | | Train | | Test | |
	Model	In-t	Out-t	In-t	Out-t	In-t	Out-t	In-t	Out-t
					$s = 5\%$ subsampling ratio				
Discrete Operator	I-MP-PDE	8.154E−3	8.166E−3	7.926E−3	8.225E−3	7.055E−4	7.097E−4	1.138E−3	1.116E−3
	DeepONet	3.330E−3	7.370E−3	1.346E−2	1.408E−2	8.331E−4	9.295E−3	1.692E−2	3.256E−2
INR	SIREN	8.741E−3	1.767E−1	4.303E−2	2.126E−1	2.738E−3	1.818E−2	3.339E−2	6.964E−2
	DINo	**1.029e−3**	**1.655e−3**	**1.326e−3**	**1.813e−3**	**4.088e−5**	**4.121e−5**	**6.415e−5**	**7.392e−5**
					$s = 25\%$ subsampling ratio				
Discrete Operator	I-MP-PDE	3.135E−4	7.245E−4	3.476E−4	7.658E−4	3.293E−5	1.108E−4	5.142E−5	1.545E−4
	DeepONet	9.016E−4	5.936E−3	9.376E−3	1.328E−2	5.722E−4	1.061E−2	1.757E−2	3.221E−2
INR	SIREN	5.180E−3	2.175E−1	2.436E−1	3.861E−1	8.995E−4	1.292E−2	1.783E−2	5.143E−2
	DINo	**1.020e−4**	**4.504e−4**	**2.646e−4**	**5.951e−4**	**3.949e−6**	**4.436e−6**	**1.089e−5**	**1.174e−5**
					$s = 100\%$ subsampling ratio				
Discrete	CNODE	2.319E−2	9.652E−2	2.305E−2	1.143E−1	2.337E−5	5.280E−4	3.057E−5	7.288E−4
	MP-PDE	1.140E−4	5.500E−4	**1.785e−4**	5.856E−4	**1.718E−7**	1.993E−5	**9.256e−7**	4.261E−5
Operator	MNO	**3.190e−5**	8.678E−4	2.763E−4	8.946E−4	9.381E−6	4.890E−3	1.993E−4	6.128E−3
	DeepONet	1.375E−3	6.573E−3	9.704E−3	1.244E−2	6.431E−4	1.293E−2	1.847E−2	3.317E−2
	SIREN	1.066E−3	4.336E−1	3.874E−1	1.037E0	3.674E−4	9.956E−3	3.013E−2	7.842E−2
INR	MFN	1.651E−3	1.037E0	2.106E−1	1.059E0	1.408E−4	1.763E−1	4.735E−3	2.274E−1
	DINo (no sep.)	3.235E−4	1.593E−3	7.850E−4	1.889E−3	2.641E−6	4.081E−5	5.977E−5	2.979E−4
	DINo	8.339E−5	**3.115e−4**	2.092E−4	**4.311e−4**	3.309E−6	**3.506e−6**	9.495E−6	**9.946e−6**

$\mathcal{X}_{tr} = \mathcal{X}_{ts}$. We evaluate MSE error on $\mathcal{X}'$ over the train time interval (*In-t*) and beyond (*Out-t*) at each subsampling ratio. • **Flexibility w.r.t. input grid.** We vary the test observation grid, *i.e.* $\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$ and perform inference on $\mathcal{X}' = \mathcal{X}_{ts}$, *i.e.* on the test observation grid (*In-s*) under two settings. **Generalizing across grids:** $\mathcal{X}_{tr}, \mathcal{X}_{ts}$ are subsampled differently from the same uniform grid; s_{tr} (resp. s_{ts}) is the train (resp. test) subsampling ratio. **Generalizing across resolutions:** $\mathcal{X}_{tr}, \mathcal{X}_{ts}$ are subsampled with the same ratio s from two uniform grids with different resolutions; the train resolution is fixed to $r_{tr} = 64$ while we vary the test resolution $r_{ts} \in \{32, 64, 256\}$. • **Data on manifold.** We consider a PDE on a sphere and combine several evaluation scenarios, as described later. • **Finer time resolution.** The inference time grid $\mathcal{T}'$ has a finer resolution than the train one $\mathcal{T}$.

7.4.2   Results

Space and time extrapolation.   We report prediction MSE in Table 7.2 for varying subsampling ratios $s \in \{5\%, 25\%, 100\%\}$ on *Navier-Stokes* and *Wave*. Appendix E.1 provides a fine-grained evaluation inside the train observation grid (*In-s*) or outside (*Out-s*) and reports additionally the results for $s = 50\%$. We visualize

Table 7.3. – **Flexibility w.r.t. input grid.** Observed test / train grid differ ($\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$). We report *test* MSE ($\downarrow$) for *Navier-Stokes* on $\mathcal{X}' = \mathcal{X}_{ts}$ (*In-s*). **Green** **Yellow** **Red** mean excellent, good, poor MSE.

Subsampling	Test→	$s_{ts} = 5\%$		$s_{ts} = 50\%$		$s_{ts} = 100\%$	
Train ↓		In-t	Out-t	In-t	Out-t	In-t	Out-t
$s_{tr} = 5\%$	MP-PDE	1.330E−1	3.852E−1	1.859E−1	6.680E−1	2.105E−1	7.120E−1
	DINo	1.494e−3	2.291e−3	1.257e−3	1.883e−3	1.287e−3	1.947e−3
$s_{tr} = 50\%$	MP-PDE	4.494E−2	9.403E−2	4.793E−3	1.997E−2	6.330E−3	3.712E−2
	DINo	2.470e−4	4.697e−4	2.073e−4	4.284e−4	2.058e−4	4.361e−4
$s_{tr} = 100\%$	MP-PDE	1.358E−1	3.355E−1	1.182E−2	2.664E−2	1.785e−4	5.856E−4
	DINo	2.495e−4	4.805e−4	2.109e−4	4.325e−4	2.092E−4	4.311e−4

(a) **Generalization across grids**: $\mathcal{X}_{tr}, \mathcal{X}_{ts}$ are subsampled with ratios $s_{tr} \neq s_{ts}$ among $\{5, 50, 100\}\%$ from the same uniform 64×64 grid.

Test resolution →		$r_{ts} = 32$ - $\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$		$r_{ts} = 64$ - $\mathcal{X}_{ts} = \mathcal{X}_{tr}$		$r_{ts} = 256$ - $\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$	
Subsampling ↓		In-t	Out-t	In-t	Out-t	In-t	Out-t
$s = 5\%$	MP-PDE	3.209E−1	6.472E−1	2.465e−4	1.105E−3	2.239E−1	8.253E−1
	DINo	5.308e−3	9.544e−3	2.533E−4	8.832e−4	1.991e−3	2.942e−3
$s = 100\%$	MNO	4.547E−3	9.281E−3	1.277e−4	8.525E−4	2.174E−3	4.975E−3
	MP-PDE	4.194E−2	9.109E−2	1.597E−4	6.483E−4	4.648E−2	1.381E−1
	DINo	2.321e−4	6.386e−4	2.320E−4	6.385e−4	2.322e−4	6.385e−4

(b) **Generalization across resolutions**: $\mathcal{X}_{ts}$ (resp. $\mathcal{X}_{tr}$) are subsampled at the same ratio $s \in \{5, 100\}\%$ from different uniform grids with resolution $r_{ts} \in \{32, 64, 256\}$ (resp. $r_{tr} = 64$).

some predictions in Appendix E.2. DINo is compared to all baselines when $s = 100\%$, *i.e.* $\mathcal{X}' = \mathcal{X}_{tr} = \mathcal{X}_{ts}$, and otherwise it is compared only to models which handle irregular grids and prediction at arbitrary spatial locations (DeepONet, SIREN, MFN, I-MP-PDE). • **General analysis.** We observe that all models degrade when the subsampling ratio s decreases. DINo performs competitively overall: it achieves the best *Out-t* performance on all subsampling settings, it outperforms all the baselines on low subsampling ratios and performs comparably to the competitive discretized alternatives (MP-PDE, CNODE) and operator (MNO) when $s = 100\%$, *i.e.* when observation and inference grids are equal. Note that this fully observed setting is favorable for CNODE, MP-PDE and MNO, designed to perform inference on the observation grid. This can be seen in Table 7.2, where DINo is slightly outperformed only for few settings. MP-PDE is significantly better only on *Wave* for *In-t*. Overall, CNNs and GNNs exhibit good performance for spatially local dynamics like *Wave*, while INRs (like DINo) and MNO are more adapted to global dynamics like *Navier-Stokes*. • **Analysis per model.** MP-PDE is the most competitive baseline across datasets as it combines a strong and flexible encoder (GNNs) to a good dynamics model; however, it cannot predict outside the observation grid (*Out-s*). To keep a strong competitor, we extend this baseline into

its interpolative version I-MP-PDE on subsampled settings. I-MP-PDE is competitive for high subsampling ratios, *e.g.* $s \in \{50\%, 100\%\}$ but underperforms w.r.t. DINo at lower subsampling ratios due to the accumulated interpolation error. MNO is a competitive baseline on *Navier-Stokes*, performing on par with MP-PDE and DINo inside the training horizon (*In-t*); its performance on *Out-t* degrades more significantly compared to other models, especially DINo. DeepONet is more flexible than MP-PDE as it can predict at arbitrary locations. As no interpolation error is introduced, it outperforms I-MP-PDE for $s = 5\%$ on *train* data. Yet, we observe that it underperforms especially on *Out-t* w.r.t. its alternatives. Finally, we observe that SIREN and MFN fit correctly the train horizon *In-t* on *train*, yet generalize poorly outside this horizon *Out-t* or on new initial conditions (*test*). We highlight that this is not the case for DINo which extrapolates temporally and generalizes to new initial conditions thanks to its sequential modeling of the flow. Thus, DINo *is currently the state-of-the-art* INR *model for spatiotemporal data.* • **Modulation.** We observe that modulating both amplitudes and frequencies (row DINo (no sep.) in Table 7.2) degrades performance w.r.t. DINo (row DINo in Table 7.2) that only modulates amplitudes. Amplitude modulation enables long temporal extrapolation and reduces the number of parameters. Hence, as opposed to DINo (no sep.) which is outperformed by some baselines, time-space variable separation in DINo is an essential ingredient of the model to reach state-of-the-art levels.

Flexibility w.r.t. input grid.   We consider in Table 7.3 *Navier-Stokes* and compare DINo to the most competitive baselines, MP-PDE and MNO (with $s = 100\%$ subsampling ratio). • **Generalizing across grids.** In Table 7.3a, we consider that the test observation grid $\mathcal{X}_{ts}$ is different from the train one $\mathcal{X}_{tr}$. This occurs when sensors differ between two observed trajectories. We vary the subsampling ratio for the train observation grid s_{tr} and the test one s_{ts}. We report *test* MSE on new grids $\mathcal{X}' = \mathcal{X}_{ts}$. We observe that DINo is very robust to changing grids between *train* and *test*, while MP-PDE's performance degrades, especially for low subsampling ratios, *e.g.* 5%. For reference, we report in Table E.2 Appendix E.1 (col. 3) the performance when $\mathcal{X}' = \mathcal{X}_{tr}$, where MP-PDE is substantially better. • **Generalizing across spatial resolutions.** In Table 7.3b we vary the test resolution r_{ts}. We train at a resolution $r_{tr} = 64$ and perform inference at resolutions $r_{ts} \in \{32, 64, 256\}$. For that, we build a high-fidelity 256×256 simulation dataset and downscale it to obtain the other resolutions. We observe that DINo's performance is the stablest across resolutions in the uniform or irregular setting. MNO is also relatively stable but is only applicable to uniform grids while MP-PDE is particularly brittle, especially for a 5% ratio.

Model	In-t	Out-t
I-MP-PDE	1.908E−3	7.240E−3
DINo	**1.063e−4**	**6.466e−4**
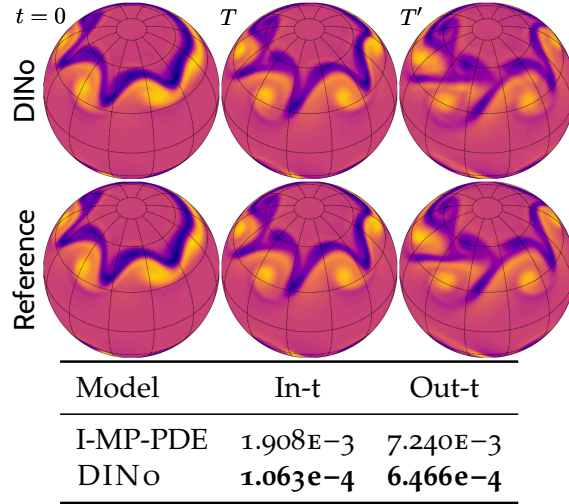
Figure 7.5. – **Data on manifold.** *Shallow-Water* superresolution *test* prediction for DINo (top) and reference (middle); *test* MSE (↓) (bottom).

Data on manifold.   We consider in Figure 7.5 *Shallow-Water* in a super-resolution setting: test resolution is twice the train one, close to weather prediction applications. We observe an irregular 3D Euclidean coordinate grid $\mathcal{X}_{\text{tr}} = \mathcal{X}_{\text{ts}} \subset \mathbb{R}^3$ shared for *train* and *test*. It samples uniformly Euclidean positions on the sphere, via the quasi-uniform skipped latitude-longitude grid (Weller et al. 2012). We predict the PDE on *test* trajectories with a conventional latitude-longitude inference grid $\mathcal{X}'$. At Earth scale, $\mathcal{X}_{\text{tr}}$ corresponds to a resolution of about $300\,\text{km}$, and $\mathcal{X}'$ to $150\,\text{km}$. DINo significantly outperforms I-MP-PDE, making it a viable candidate for this complex setting.

Finer time resolution.   We consider in Table 7.4 a longer and 10 times finer test time grid $\mathcal{T}'$ than the train grid $\mathcal{T}$ on *Navier-Stokes*. We observe the same spatial uniform grid across *train* and *test* and perform inference on this grid. We compare DINo that uses an ODE solver, to interpolating coarser predictions obtained at the train resolution (I-DINo). We report the corresponding *test* MSE.

Table 7.4. – **Finer time resolution.** *Test* MSE (↓) under $\mathcal{T}'$ for *Navier-Stokes*.

Model	In-t	Out-t
I-DINo (linear)	3.459E−4	5.598E−4
I-DINo (quadratic)	2.165E−4	4.473E−4
DINo (ODE solve)	**2.151e−4**	**4.388e−4**

We observe that the ODE solver accurately extrapolates outside the train temporal grid, outperforming interpolation. This confirms that DINo benefits from its continuous-time modeling of the flow, providing consistency and stability across temporal resolutions.

7.5   Related Work

We reviewed in Section 2.2.2 various neural PDE forecasters, based on discretized, neural operator and INR. We highlight the advantages of DINo over existing approaches. The all make restrictive assumptions on the space discretization. Neural operators lack flexibility when encoding spatial observations: FNO is limited to uniform Cartesian observation grids due to Fast Fourier Transform (FFT); GNO does not adapt well to changing observation grids as for the GNN-based models in the previous paragraph; DeepONet is limited to input observations on fixed observation locations. The latter are chosen at random spatial positions but should remain fixed throughout training and testing. Spatiotemporal INR only learn mappings from an initial condition v_0 to a function of time v_t in the train domain, they fail to predict beyond train conditions, as we show in Section 7.4. DINo is a new instance of INR which solves this limitation via a time-continuous dynamics model of the underlying flow.

7.6   Conclusion

We propose DINo, a novel space- and time-continuous data-driven forecasting model for PDEs. DINo handles settings encountered in many applications, where existing methods fail. We assess its extensive spatiotemporal extrapolation abilities on a variety of PDEs and its generalization to unseen sparse irregular meshes and resolutions. Its competitive results over recent PDE forecasters make it a strong alternative for real-world settings with free-formed spatiotemporal conditions. There are many promising extensions *e.g.* improving generalization to new parameters (Kirchmeyer et al. 2022a).

Part V

CONCLUSION AND PERSPECTIVES

Chapter 8

Concluding Remarks

8.1 Overview

We now conclude this work by briefly summarizing the main contributions of this thesis, detailed in Section 1.3.

Problems   In this thesis, we proposed new algorithms for improving the robustness of Neural Network (NN)s to distribution shift. We considered learning settings which consider access to some data of the test domain, at training time, at test-time or not. Observing some target samples allows to better model the target labeling function when it differs from the training one (concept shift setting).

Applications   We tailored our methods to two representative problems with their similarities and differences, classification and forecasting of spatiotemporal dynamics. These two problems cover a variety of applications, from computer vision to numerical simulation in science. They are fundamentally different: while classification was considered for static data, spatiotemporal forecasting requires learning some underlying temporal dynamics, a challenging problem. Yet, the corresponding Machine Learning (ML) approaches share some similarities.

Modeling contributions   Our new models can be grouped into two different categories. A first group proposes some alternatives to the standard Expected Risk Minimization (ERM) inference principle which assumes all data to be Independent and Identically Distributed (IID). It leverages to this end multiple domains for training. We operated on representations in the Unsupervised Domain Adaptation (UDA) setting, where we explored two failure cases of standard methods that learn domain-invariant representations, namely missing data and complex domain shifts. To handle these problems, we respectively introduced a joint imputation-UDA model and proposed to transfer latent representations with Optimal Transport (OT). We also proposed two separate models operating in the NN parameter space. We first adapted parameters in the test-time adapta-

tion setting by performing environmnent-conditioning via linear hypernetworks. This approach defines a low-dimensional parameter subspace to which adaptation is restricted. We then proposed to average independent weights for Out-of-distribution (OOD) generalization, introducing diversity considerations in prior weight averaging techniques. A second direction is to improve the NN architectures at hand. In the context of Partial Differential Equation (PDE) modeling, we better accounted for the continuous space and time nature of PDEs to achieve better generalization to new spatiotemporal discretizations. We leveraged Implicit Neural Representation (INR)s and proposed a new model which extrapolates better in time than existing INRs.

Theoretical contributions   We made various theoretical throughout these work. In the classification setting, we derived new generalization bounds which allowed to rigourously define which assumptions are required for each of our models to achieve OOD generalization. In the UDA setting, we leveraged unlabelled target samples to define generalization guarantees in the challenging settings where some target components are missing and under Generalized Target Shift (GeTarS) (*i.e.* covariate shift combined to concept shift). In the Domain Generalization (DG) setting, we extended the bias-variance decomposition to the OOD setting and related variance to covariate shift and bias to concept shift. We also extended this decomposition to weight averaging strategies, proposing for the first time an empirically verified analysis of weight averaging, centered around the concept of diversity. In the dynamics forecasting setting, we justified the validity of using linear hypernetworks for environment-conditioning when system parameters vary across domains.

Experimental contributions   Experimentally, we were able to outperform the state-of-the-art in all our contributions and open-sourced our implementations. In the classification setting, we compared our models to representative baselines on well-established computer vision image benchmarks. On the competitive OOD DomainBed benchmark (Gulrajani et al. 2021a), our model DiWA is currently the state-of-the-art based on a thorough and fair comparison. In the spatiotemporal setting, we were able to outperform recent approaches on challenging simulations of Ordinary Differential Equation (ODE)/PDEs *e.g.* the Navier-Stokes equations involved in fluid dynamics.

8.2 Other work

Apart from these main contributions, we also explored regression problems in Aggarwal et al. 2019 which proposed to use conditional Generative Adversarial Network (GAN) for regression as an alternative to other approaches.

Karan Aggarwal, Matthieu Kirchmeyer, Pranjul Yadav, S. Sathiya Keerthi, and Patrick Gallinari (2019). "Regression with Conditional GAN". in: *CoRR* abs/1905.12868. arXiv: 1905.12868. URL: http://arxiv.org/abs/1905.12868

8.3 Perspectives

We now discuss various open questions raised by these contributions for our two applications: first classification, then spatiotemporal modeling.

8.3.1 Classification

Architecture vs. dataset vs. learning objective   Classification is one of the most explored ML problems, especially on images where highly curated benchmarks are available. A variety of architectures were proposed, from simple ResNet-50 used in this thesis to complex foundation models *e.g.* the multi-modal CLIP architecture (Radford et al. 2021). Some recent work confirmed that these more complex architectures and the size and variety of the training datasets have a big if not the most important impact on OOD performance (Ruan et al. 2022; Arpit et al. 2021). The question of the relative importance of learning objective over architecture and data curation remains to be answered.

For less explored problems, there is no such choice of large architectures or large and well-curated datasets for pretraining / training these models. The design of new training objectives, which is the focus of this thesis, then plays a more important role for these problems. This calls for new real-world benchmarks to better evaluate Domain Adaptation (DA) and DG methods. This is particularly relevant as current benchmarks such as DomainBed (Gulrajani et al. 2021a) are now reaching a saturation point where performance gains are incremental. With colleagues, we made a first step towards that objective by organizing a challenge at ECML-PKDD 2022 on DG. We introduced a new benchmark for computational advertising (Click-Through-Rate (CTR) prediction) to better assess how existing strategies fare on tabular data, without available pretrained models. We evaluated our UDA model for imputation on a similar dataset, which inspired our initial

research problem of handling missing data in UDA. The definition of such new benchmarks is currently one of the most important future direction for the field.

8.3.2   Spatiotemporal modeling

More real-world applications   We modeled, with NNs, simulated data of complex PDEs involved in many physical phenomena. Most current papers on the topic of ODE / PDE modeling, if not all, also consider simulated data for learning NN-based surrogates. Yet, there is a big gap between models that work on simulations and models usable on real-world data. Exploring applications of NN-based surrogates to more real-world problems *e.g.* weather prediction (Pathak et al. 2022; Lam et al. 2022) is an important future research direction. These real-world applications require scaling up the architectures to handle massive amounts of data, multiple interacting vector and scalar fields and requires additional engineering effort.

Defining neural ODE/PDE surrogates with extensive generalization capabilities Finally, although we made several steps towards building neural ODE/PDE surrogates that generalize, there are still many open generalization problems. Indeed, currently no single model is able to handle all generalization problems, detailed in Section 2.2.3, but only few of these problems. There is still a long way before designing general purpose neural surrogates, able to replace state-of-the-art numerical tools, which are not subject to these generalization problems. This topic is still open and we made some initial attempts at adressing it.

Part VI

APPENDIX

Appendix A

Supplementary Material of Chapter 3

A.1 OT formulation for Adaptation-Imputation

We present here in more details our model using Optimal Transport (OT) as a divergence metric. The formulation is slightly different compared to Adversarial (ADV) models. We replace the $\mathcal{H}$-divergence approximation given by the discriminators D_1 and D_2 by the Wasserstein distance between source and target instances (D_1) and true and imputed feature representations (D_2), following the original ideas in Shen et al. 2018; Damodaran et al. 2018. In practice, we compute the Wasserstein distance using its primal form by finding a joint coupling matrix γ, using a linear programming approach Peyré et al. 2019. In Damodaran et al. 2018; Courty et al. 2017a, the OT problem is formulated on the joint $p(X, Y)$ distributions. Similarly to Shen et al. 2018, in our case, we focus on a plan that acts only on the feature space without taking care of the labels. This leads to:

$$\mathcal{L}_1 = \sum_{ij} \left(\|\mathbf{z}_{\mathbf{S_1}}^{(\mathbf{i})} - \mathbf{z}_{\mathbf{T_1}}^{(\mathbf{j})}\|^2 + \|\widehat{\mathbf{z}}_{\mathbf{S_2}}^{(i)} - \widehat{\mathbf{z}}_{\mathbf{T_2}}^{(j)}\|^2 \right) \gamma_{\mathbf{1}_{ij}} \tag{A.1}$$

where $\gamma_{\mathbf{1}_{ij}}$ is the alignment value between source instance i and target instance j.

For the imputation part, we keep the reconstruction Mean-Squared Error (MSE) component in Eq. (3.6) and derive the distribution matching loss as:

$$\mathcal{L}_{OT} = \sum_{ij} \|\mathbf{z}_{\mathbf{S_2}}^{(\mathbf{i})} - \widehat{\mathbf{z}}_{\mathbf{S_2}}^{(j)}\|^2 \gamma_{\mathbf{2}_{ij}} \tag{A.2}$$

where $\gamma_{\mathbf{2}_{ij}}$ is the alignment value between source instance i and j. The final imputation loss is:

$$\mathcal{L}_2 = \lambda_{OT} \times \mathcal{L}_{OT} + \lambda_{MSE} \times \mathcal{L}_{MSE} \tag{A.3}$$

The classification term in Eq. (3.7) is unchanged.

The optimization problem in Eq. (3.9) is solved in two stages following an alternate optimization strategy:

- We fix all parameters but γ_1 and γ_2 and find the joint coupling matrices γ_1 and γ_2 using EMD $\min_{\gamma_1,\gamma_2} L$

- We fix γ_1 and γ_2 and solve $\min_{g_1,g_2,r,f} L$

In practice, we first minimize $\mathcal{L}_3$ for a couple of epochs (taken to be 10 for `digits`) then minimize $\lambda_1\mathcal{L}_1 + \lambda_2\mathcal{L}_2 + \lambda_3\mathcal{L}_3$ in the remaining epochs. Learning rate and parameters are detailed further in Appendix A.4.

A.2   Proofs

Theorem (3.1). Given $f \in \mathcal{F}, \widehat{g}$ in Eq. (3.2) and $p_S(\widehat{Z})$, $p_T(\widehat{Z})$ the latent marginal distributions obtained with g.

$$\mathcal{E}_T(f \circ \widehat{g}) \leq \underbrace{\left[\mathcal{E}_S(f \circ \widehat{g}) + d_{\mathcal{F}\Delta\mathcal{F}}(p_S(\widehat{Z}), p_T(\widehat{Z})) + \lambda_{\mathcal{H}_{\widehat{g}}} \right]}_{\text{Domain Adaptation } DA}$$

with $\mathcal{E}_S(\cdot), \mathcal{E}_T(\cdot)$ the expected error w.r.t. to the labelling function f_S, f_T on S, T respectively; $\mathcal{F}\Delta\mathcal{F}$ the symmetric difference hypothesis space; $d_{\mathcal{H}}$ the $\mathcal{H}$-divergence for $\mathcal{H} = \mathcal{F}\Delta\mathcal{F}$ and $\lambda_{\mathcal{H}_{\widehat{g}}} = \min_{f' \in \mathcal{F}}[\mathcal{E}_S(f' \circ \widehat{g}) + \mathcal{E}_T(f' \circ \widehat{g})]$, the joint risk of the optimal hypothesis.

Proof. We apply Ben-David et al. 2010 to form the bound in $\mathcal{Z}$ using $\widehat{g}$.   □

Lemma (3.1). For any continuous density distribution p, q defined on an input space $\mathcal{X}$, such that $\forall \mathbf{x} \in \mathcal{X}, q(\mathbf{x}) > 0$, the inequality $\sup_{\mathbf{x} \in \mathcal{X}}[p(\mathbf{x})/q(\mathbf{x})] \geq 1$ holds. Moreover, the minimum is reached when $p = q$.

Proof. Suppose that $\nexists \mathbf{x} \in \mathcal{X} s.t. \sup_{\mathbf{x}} p(\mathbf{x})/q(\mathbf{x}) \geq 1$. This means that $\forall \mathbf{x}, p(\mathbf{x}) < q(\mathbf{x})$. By integrating those positive and continuous functions on their domains, we are lead to the contradiction that the integral of one of them is not equal to 1. Thus, $\exists \mathbf{x} \in \mathcal{X} s.t. p(\mathbf{x})/q(\mathbf{x}) \geq 1$. Thus, $\sup_{\mathbf{x} \in \mathcal{X}}[p(\mathbf{x})/q(\mathbf{x})] \geq 1$, with equality trivially when $p = q$.   □

Proposition (3.2). Under Assumption 3, given $f \in \mathcal{F}, \widehat{g}$ in Eq. (3.2) and g in Eq. (3.1),

$$\mathcal{E}_T(f \circ g) \leq \underbrace{\sup_{\mathbf{z} \sim p(\mathcal{Z})} \left[\frac{p_S(Z_2 = \mathbf{z}_2 | \mathbf{z}_1)}{p_S(\widehat{Z}_2 = \mathbf{z}_2 | \mathbf{z}_1)} \right]}_{\text{Imputation error on S } (I_S)} \times \underbrace{\sup_{\mathbf{z} \sim p(\mathcal{Z})} \left[\frac{p_S(\widehat{Z}_2 = \mathbf{z}_2 | \mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2 | \mathbf{z}_1)} \right]}_{\text{Transfer error of Imputation } T_I} \times \mathcal{E}_T(f \circ \widehat{g}) \tag{3.11}$$

$$\underbrace{}_{\text{Imputation error on T } I_T}$$

Under Lemma 3.1, $I_T = 1$ is the minimal value reached when $p_S(Z_2|Z_1) = p_S(\widehat{Z}_2|Z_1)$ and $p_S(\widehat{Z}_2|Z_1) = p_T(\widehat{Z}_2|Z_1)$. In this case, $\mathcal{E}_T(f \circ g) = \mathcal{E}_T(f \circ \widehat{g})$.

Proof. We denote f_T^z, the latent target labeling function. Moreover, for simplicity, we write $h_{\widehat{g}} = f \circ \widehat{g}$, $h_g = f \circ g$ and $\forall \mathbf{z} \sim p(\mathcal{Z}), S_D(\mathbf{z}) = \dfrac{p_D(Z_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_D(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)}$

$$
\begin{aligned}
\mathcal{E}_T(h_g) &= \mathbb{E}_{\mathbf{x}_T \sim p_T(X)}[\mathbb{I}(h_g(\mathbf{x}_T) \neq f_T(\mathbf{x}_T))] \\
&= \mathbb{E}_{\mathbf{z_{T_1}} \sim p_T(Z_1), \mathbf{z_{T_2}} \sim p_T(Z_2|Z_1)}[\mathbb{I}(f(\mathbf{z_{T_1}}, \mathbf{z_{T_2}}) \neq f_T^z(\mathbf{z_{T_1}}, \mathbf{z_{T_2}}))] \\
&= \mathbb{E}_{\mathbf{z_{T_1}} \sim p_T(Z_1), \widehat{\mathbf{z}}_{\mathbf{T_2}} \sim p_T(\widehat{Z}_2|Z_1)}[\frac{p_T(Z_2 = \widehat{\mathbf{z}}_{\mathbf{T_2}}|\mathbf{z_{T_1}})}{p_T(\widehat{Z}_2 = \widehat{\mathbf{z}}_{\mathbf{T_2}}|\mathbf{z_{T_1}})}\mathbb{I}(f(\mathbf{z_{T_1}}, \widehat{\mathbf{z}}_{\mathbf{T_2}}) \neq f_T^z(\mathbf{z_{T_1}}, \widehat{\mathbf{z}}_{\mathbf{T_2}}))] \\
&\leq \sup_{\mathbf{z} \sim p(\mathcal{Z})}[S_T(\mathbf{z})]\mathbb{E}_{\mathbf{x}_T \sim p_T(X)}[\mathbb{I}(h_{\widehat{g}}(\mathbf{x}_T) \neq f_T(\mathbf{x}_T))] \\
&= \sup_{\mathbf{z} \sim p(\mathcal{Z})}[S_T(\mathbf{z})]\mathcal{E}_T(h_{\widehat{g}})
\end{aligned}
$$

However, $\forall \mathbf{z} \in \mathcal{Z}, S_T(\mathbf{z})$ cannot be computed as there is not supervision possible on T. We will instead apply Assumption 3 and use source data for which we can compute $S_S(\mathbf{z})$.

$$
\begin{aligned}
\forall \mathbf{z} \in \mathcal{Z} \ S_T(\mathbf{z}) &= \frac{p_T(Z_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)} \\
&= \frac{p_S(Z_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)} \qquad\qquad \text{Assumption 3} \\
&= \frac{p_S(Z_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_S(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)} \times \frac{p_S(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)} \\
&= S_S(\mathbf{z}) \times \frac{p_S(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)}
\end{aligned}
$$

Thus by applying $\sup$,

$$
\sup_{\mathbf{z} \sim p(\mathcal{Z})}[S_T(\mathbf{z})] = \sup_{\mathbf{z} \sim p(\mathcal{Z})}[S_S(\mathbf{z})] \times \sup_{\mathbf{z} \sim p(\mathcal{Z})}[\frac{p_S(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)}{p_T(\widehat{Z}_2 = \mathbf{z}_2|\mathbf{z}_1)}]
$$

This yields Eq. (3.11).

If $I_T = 1$ when $p_S(Z_2|Z_1) = p_S(\widehat{Z}_2|Z_1)$ and $p_S(\widehat{Z}_2|Z_1) = p_T(\widehat{Z}_2|Z_1)$ per Lemma 3.1, then $S_T(\mathbf{z}) = 1$ and $\mathcal{E}_T(f \circ g) = \mathcal{E}_T(f \circ \widehat{g})$. □

Proposition (**3.3**). Assume a joint distribution $p_{\widetilde{T}}(X, Y)$ where $p_{\widetilde{T}}(X) = p_T(X)$ and $Y = h_{\widehat{g}}(X)$ where $h_{\widehat{g}} = f \circ \widehat{g} \in \mathcal{H}_{\widehat{g}}$ is a candidate hypothesis. Then,

$$\lambda_{\mathcal{H}_{\widehat{g}}} \leq \min_{h_{\widehat{g}} \in \mathcal{H}_{\widehat{g}}} \left[\mathcal{E}_S(h_{\widehat{g}}) + \mathcal{E}_{\widetilde{T}}(h_{\widehat{g}}) + \mathcal{E}_T(f_{\widetilde{T}}) \right]$$

with $\mathcal{E}_T(f_{\widetilde{T}}) = \Pr_{\mathbf{x} \sim p_T(X)}(f_{\widetilde{T}}(\mathbf{x}) \neq f_T(\mathbf{x}))$ the error of the pseudo-labelling function $f_{\widetilde{T}}$ on T.

Proof. We know that $p_{\widetilde{T}}(X) = p_T(X)$ as instances are not changed by applying the pseudo-labelling function. Thus, given $h_{\widehat{g}} \in \mathcal{H}_{\widehat{g}}$

$$\mathcal{E}_T(h_{\widehat{g}}) = \mathcal{E}_T(h_{\widehat{g}}, f_T) = \mathcal{E}_{\widetilde{T}}(h_{\widehat{g}}, f_T)$$

Applying the triangle inequality for classification error (Crammer et al. 2008),

$$\mathcal{E}_{\widetilde{T}}(h_{\widehat{g}}, f_T) \leq \mathcal{E}_{\widetilde{T}}(h_{\widehat{g}}, f_{\widetilde{T}}) + \mathcal{E}_{\widetilde{T}}(f_{\widetilde{T}}, f_T)$$

Finally, we can rewrite $\mathcal{E}_{\widetilde{T}}(h_{\widehat{g}}, f_{\widetilde{T}}) = \mathcal{E}_{\widetilde{T}}(h_{\widehat{g}})$ and $\mathcal{E}_{\widetilde{T}}(f_{\widetilde{T}}, f_T) = \mathcal{E}_T(f_{\widetilde{T}}, f_T) = \mathcal{E}_T(f_{\widetilde{T}})$. $\square$

A.3 Dataset description

A.3.1 Digits

We scale all images to 32×32 and normalize the input in $[-1, 1]$. When adaptation involves a domain with three channels (SVHN or MNIST-M) and a domain with a single channel, we simply triplicate the channel of the latter domain. As in Damodaran et al. 2018 we use balanced source batches which proves to increase performance especially when the source dataset is imbalanced (e.g. SVHN and USPS datasets) while the target dataset (usually MNIST derived) is balanced. Scaling the input images enables us to use the same architecture across datasets. In practice the embedding size is 2048 after preprocessing. For missing versions, we set pixel values to zero in a given patch as shown in Figure 3.3. The `digits` datasets are provided with a predefined train / test split. We report accuracy results on the target test set and use the source test set as validation set (Appendix A.4.2). The number of instances in each dataset is reported in Table A.1. We run each model five times.
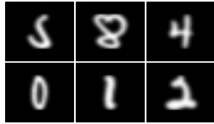
	USPS	MNIST	SVHN	MNIST-M
Train	7438	60k	73257	60k
Test	1860	10k	26032	10k
Size	28 × 28	28 × 28	32 × 32	28 × 28
Channels	1	1	3	3

Table A.1. – Statistics on `digits` datasets

Dataset	ads-kaggle				ads-real			
Domain	Source		Target		Source		Target	
Split	Train	Test	Train	Test	Train	Test	Train	Test
Positive	246 872	61 841	92 333	22 943	X	X	X	X
Negative	699 621	174 783	854 160	213 681	X	X	X	X
Total	946 493	236 624	946 493	236 624	24 465 756	3 760 233	819 073	147 358
$p(Y=1)$	0,2608	0,2613	0,0976	0,0970	X	X	X	X

Table A.2. – Statistics on `ads` datasets

A.3.2 Amazon

Each domain has around 2000 samples and we use pretrained features https://github.com/jindongwang/transferlearning/tree/master/data#amazon-review which follows the data processing pipeline in Chen et al. 2012. Each review is preprocessed as a feature vector of unigrams and bigrams keeping only the 5000 most frequent features. In practice, we consider the dense version of these features after projection onto a low-dimensional sub-space of dimension 400 with PCA as in Chen et al. 2012. Datasets with missing features are built by setting the first half of the features to 0.

A.3.3 Ads

Table A.2 lists statistics on the traffic for the two `ads` datasets; we now describe how they are preprocessed. On both datasets the train and test sets are fixed. We run each model five times.

`ads-kaggle`   The Criteo Kaggle dataset is a reference dataset for Click-Through-Rate (CTR) prediction and groups one week of log data. The objective is to model the probability that a user will click on a given ad given his browsing behaviour. Positives refer to clicked ads and negatives to non-clicked ads. For each datum,

there are 13 continuous and 26 categorical features. We divide the traffic into two domains using feature number 30 corresponding to an engagement feature; for a given value for this categorical feature, all instances have a single missing numeric feature (feature number 5). We then construct an artificial dataset simulating transfer between known and new users. We process the original Criteo Kaggle dataset to have an equal number of source and target data. We then perform train / test split on this dataset keeping 20% of data for testing. We used in our experiments only continuous features; to show the benefit of modelling additional missing features, we extend the missing features list to features 1, 5, 6, 7, 11 and 12 by setting them to zero on the target domain. After these operations, 6 features are missing and 7 are non-missing. Preprocessing consists in normalizing continuous features using a log transform.

ads-real   This private dataset is similar to `ads-kaggle`. We filter out non-clicks and the final task is to model the sale probability for a clicked ad given an user's browsing history. Positives refer to clicked ads which lead to a sale; negatives to clicked ads which did not lead to a sale. We use one week of sampled logs as a training set and use the following day data as the test set. This train / test definition is used so to better correlate with the performance of a production model. Features are aggregated across user timelines and describe the clicking and purchase behavior of a user. In comparison to `ads-kaggle` more continuous features are used. The count features can be User-centric i.e. describe the global activity of the user (number of clicks, displays, sales done globally across partners) or User-partner features i.e. describing the history of the user on the given partner (number of clicks, sales... on the partner). The latter are missing for prospecting users. Counts are aggregated across varying windows of time and categories of partner catalog. We bucketize these count features using log transforms and project the features into an embedding space of size 596 with 29 features. 12 features are missing and 17 are non-missing.

A.4   Implementation details

A.4.1   Neural Net architecture

digits   We use the ADV and OT versions of our imputation model. For ADV models, we use the DANN model description in Ganin et al. 2015; for OT we use the DeepJDOT model description in Damodaran et al. 2018. Both models can be considered as simplified instances of our corresponding ADV and OT imputation models when no imputation is performed. Performance of the adaptation models
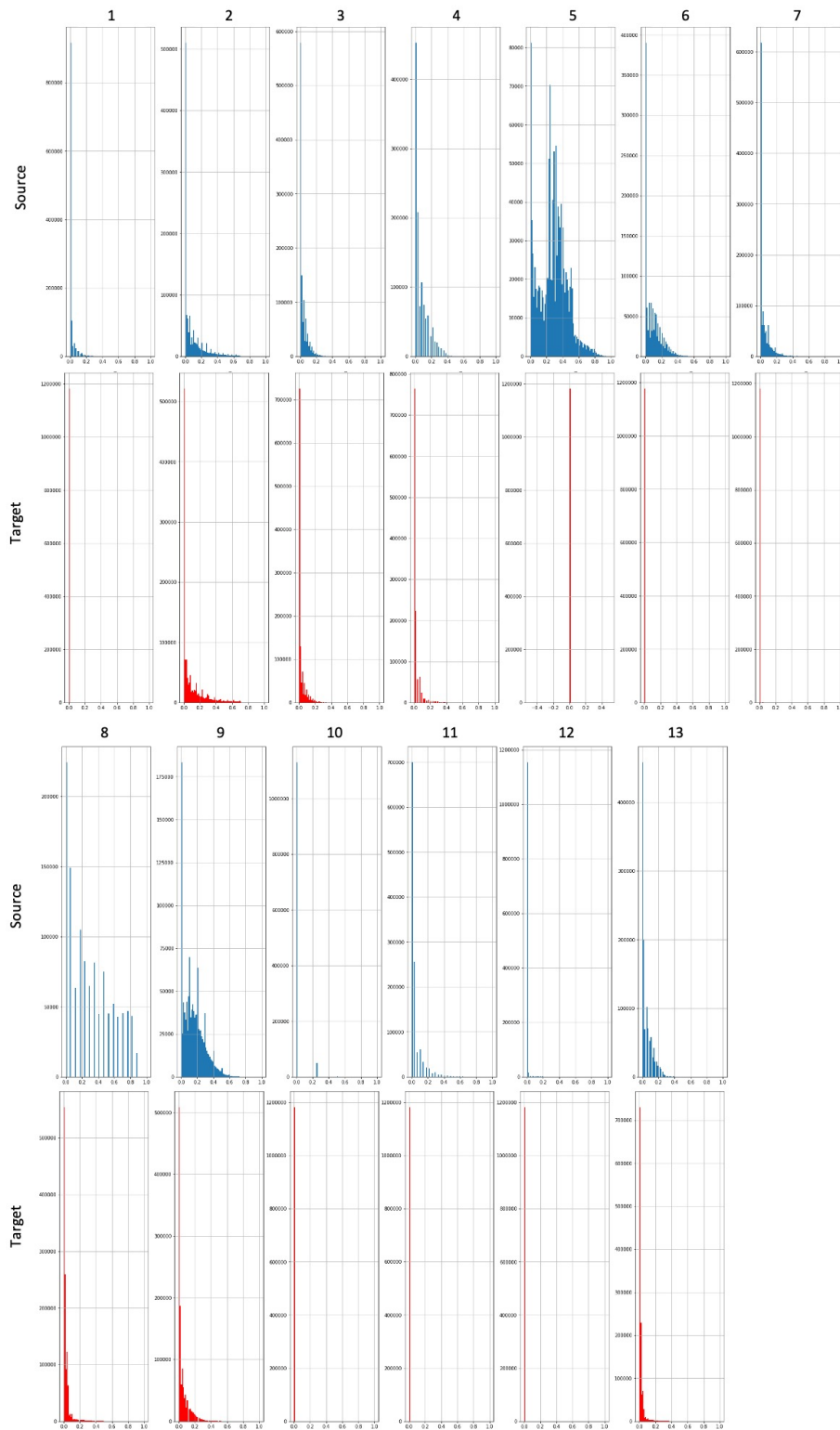
Figure A.1. – Source (blue) and Target (red) distributions on `ads-kaggle` for each feature (1 to 13)

Table A.3. – Feature mean and standard deviation on `ads-kaggle`. We set features 1, 6, 7, 11, 12 to zero on T.

Domain	Source	Target
feature 1	0.80±2.21	4.4×10^{-4}±0.041
feature 2	9.16±13.04	9.01±13.42
feature 3	4.40±6.32	3.44±6.19
feature 4	2.58±3.27	0.94±2.31
feature 5	61.09±37.67	0.0±0.0
feature 6	11.26±12.24	0.090±1.69
feature 7	4.10±6.23	0.0034±0.13
feature 8	5.12±4.50	1.91±4.26
feature 9	14.32±11.57	3.273±5.36
feature 10	0.046±0.22	1.35×10^{-5}±0.0037
feature 11	1.08±2.11	4.25×10^{-4}±0.029
feature 12	0.083±0.78	6.68×10^{-5}±0.018
feature 13	2.74±3.59	1.21±3.36

is highly dependent on the Neural Network (NN) architectures used for adaptation and classification. In order to perform fair comparisons and since our goal is to evaluate the potential of joint Adaptation-imputation-classification, we selected these architectures through preliminary tests and use them for both the ADV and OT models. The two models are described below and illustrated in Figure A.2.

- Feature extractors g_1 and g_2 consists of three convolutional layers with 5×5 kernel and 64 filters interleaved with max pooling layers with a stride of 2 and 2×2 kernel. The final layer has 128 filters. We use batch norm on convolutional layers and ReLU as an activation after max pooling. As in Damodaran et al. 2018 we find that adding a sigmoid activation layer as final activation is helpful.

- Classifier f consists of two fully connected layers with 100 neurons with batch norm and ReLU activation followed by the final softmax layer. We add Dropout as an activation for the first layer of the classifier.

- Discriminator D_1 and D_2 is a single layer NN with 100 neurons, batch norm and ReLU followed by the final softmax layer. On USPS → MNIST and MNIST → USPS dataset we use a stronger discriminator network which consists of two fully connected layers with 512 neurons.

- Generator r consists of two fully connected layers with 512 neurons, batch norm and ReLU activation. This architecture is used for ADV and OT imputation models. In practice using wider and deeper networks increases classification performance with the more complicated classification tasks (SVHN → MNIST, MNIST → MNIST-M); in these cases we add an additional fully connected network with 512 neurons. The final activation function is a sigmoid.

We use the same architecture described above for all our models to guarantee fair comparison. As a side note, the input to the imputation model's classifier is twice bigger as in the standard adaptation models.
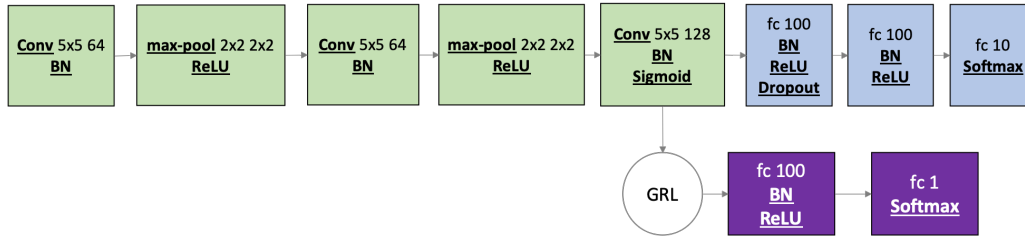
Figure A.2. – Base architecture for the ADV DANN model

`ads-kaggle` **and** `amazon`   We experiment with ADV models only. As input data is numeric and low dimensional, architectures are simpler than in `digits`. Our feature extractor is a three layered NN with 128 neurons and with a final sigmoid activation. The classifier is taken to be a single layered NN with 128 neurons and a final softmax layer. Activations are taken to be ReLUs. The domain discriminator is taken to be a two layered NN with 128 neurons and a final softmax layer. Finally the reconstructor is taken to be a two-layered NN with 256 neurons and final sigmoid activation.

`ads-real`   We experiment with ADV models only. Input features after processing are fed directly into the feature extractors g_1, g_2 consisting of two fully connected layers with 128 neurons. The classifier and discriminator is taken to be single-layered NN with 25 neurons. The reconstructor is taken to be a two-layered NN with 128 neurons. Inner activations are taken to be ReLUs and the final activation of the feature extractor is taken to be a sigmoid.

A.4.2   Network parameters

Hyperparameter tuning   Tuning hyperparameters for Unsupervised Domain Adaptation (UDA) is tricky as we do not have access to target labels and thus cannot choose parameters minimizing the target risk on a validation set. Several papers set hyperparameters through reverse cross-validation Ganin et al. 2015. Other approaches developed for model selection are based on risk surrogates obtained by estimating an approximation of the risk value on the source based on the similarity of source and target distributions (without the labels). In the experiments, we used a recent estimator, Deep Embedded Validation (DEV) You et al. 2019 for tuning the initial learning rate and for the OT imputation model, tuning

λ_1 and λ_{OT}. For other parameters, we used heuristics and typical hyperparameter values from UDA papers (such as batch size) without further tuning. We use a cross entropy link function on the source validation set; this value provides a proxy for the target test risk. Using parameters from the original paper, this estimator helps select parameter ranges which perform reasonably well. We keep the estimator unchanged for our baseline models. In the imputation case, the discriminator used for computing importance sampling weights discriminates between $\widehat{z}_S$ and $\widehat{z}_T$ i.e. D_1 (Figure 3.2).

Digits   We find that the results are highly dependent on the NN architecture and the training parameter setting. In order to evaluate the gain obtained with *Adaptation-Imputation*, we use the same NN architecture for all models (ADV and OT) but fine tune the learning rates for each model using the DEV estimator (other parameters do not have a significant impact on the classification performance).

ADV   We use an adaptive approach as in Ganin et al. 2015 for decaying the learning rate lr and updating the gradient's scale s between 0 and 1 for the domain discriminators. We choose the decay values used in Ganin et al. 2015 ie. $s = \dfrac{2}{1 + \exp(-10 \times p)} - 1$ and $lr = \dfrac{lr_i}{(1 + 10 \times p)^{0,75}}$ where p is ratio of current batches processed over the total number of batches to be processed without further tuning. We tune the initial learning rate lr_i, chosen in the range $\{10^{-2}, 10^{-2.5}, 10^{-3}, 10^{-3.5}, 10^{-4}\}$ following Appendix A.4.2. In practice we take $lr_i = 10^{-2}$ for ADV *Adaptation-Imputation*, *Adaptation-Full*, *Adaptation-IgnoreComponent* and $lr_i = 10^{-2.5}$ for ADV *Adaptation-ZeroImputation*. We use Adam as the optimizer with momentum parameters $\beta_1 = 0.8$ and $\beta_2 = 0.999$ and use the same decay strategy and initial learning rate for all components (feature extractor, classifier, reconstructor). Batch size is chosen to be 128; we see in practice that initializing the adaptation models with a source model with smaller batch size (such as 32) can be beneficial.

OT   We choose parameter $\lambda_{OT} = 0.1$ in Eq. (A.3) after tuning in the range $\{10^{-1}, 10^{-2}, 10^{-3}\}$ using DEV. We weight $\mathcal{L}_1$ in Eq. (3.8) by $\lambda_1 = 0.1$. Following Damodaran et al. 2018, batch size is taken to be 500 and we use EMD a.k.a. Wasserstein-2 distance. We initialize adaptation models with a source model in the first 10 epochs and divide the initial learning rate by two as adaptation starts for non-imputation models. For *Adaptation-Imputation* we follow a decaying strategy on the learning rate and on the adaptation weight as explained in the next item. We choose lr_i in the range $\{10^{-2}, 10^{-2.5}, 10^{-3}, 10^{-3.5}, 10^{-4}\}$. In practice we fix $lr_i = 10^{-2}$ for all models.

Imputation parameters   Ablation studies are conducted in Section 3.6.6 on weights in Eq. (3.4); in `digits` experiments we choose $\mathcal{L}_2 = \mathcal{L}_{MSE} + \mathcal{L}_{ADV}$ for ADV and OT to reduce the burden of additional feature tuning. For ADV model, we fix $\lambda_1 = \lambda_2 = \lambda_3 = 1$ in Eq. (3.8). In the OT model, we vary λ_1 between 0 and 0.1 and λ_2 between 0 and 1 following the same schedule as the gradient scale update for ADV models to reduce variance.

Ads   We use an adaptive strategy for updating the gradient scale and the learning rate with the same parameters as in the `digits` dataset. Optimizer is taken to be Adam. Batch size is taken to be big so that target batches include sufficient positive instances.

`ads-kaggle`   The initial learning rate is chosen in the range $\{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}\}$ using DEV and fixed to be 10^{-6} for all models. Batch size is taken to be 500 and we initialize models with a simple classification loss for five epochs. We run models for 50 epochs after which we notice that models reach a plateau. We find that adding a weighted MSE term allows to achieve higher stability (as measured by variance) as further studied in Section 3.6.6. In a similar fashion to Pathak et al. 2016, we tune this weight in the range $\{1, 10^{-1}, 10^{-2}, 7.5 \times 10^{-3}, 5 \times 10^{-3}, 10^{-3}\}$. We find that 0.005 offers the best compromise between mean loss and variance. Moreover on this dataset we use a faster decaying strategy for the discriminator's D_2 and the reconstructor's r learning rate, $lr = \dfrac{lr_i}{(1 + 30 \times p)^{0,75}}$ to achieve higher stability in the training curves while the feature extractor g_1, g_2 and D_1's learning rate are unchanged.

`ads-real`   The initial learning rate is chosen in the range $\{10^{-4}, 10^{-5}, 10^{-6}\}$ and fixed to be 10^{-6} for all models. The learning rate is decayed with the same parameters as `digits` for all models. We run models for ten epochs which provides a good trade-off between learning time and classification performance. Batch size is taken to be 500. We choose $\mathcal{L}_2 = \mathcal{L}_{MSE} + \mathcal{L}_{ADV}$ without further tuning; this achieves already good results.

A.4.3 Amazon

We use the same hyperparameters as `ads-kaggle`. λ_{MSE} is set to 1 without further tuning.

A.5 Latent space visualization on `digits`

In this section we visualize the embeddings $\widehat{\mathbf{z}} = \widehat{g}(\mathbf{z})$ learned by the various models on `digits` by projecting the embeddings in a 2D space using $\widehat{g}$ with t-SNE (the original embedding size being 2048). Figure A.3 represents the embeddings learned for ADV models on MNIST $\rightarrow$ MNIST-M. Figures A.4 and A.5 represent these embeddings for OT models respectively on MNIST $\rightarrow$ MNIST-M and MNIST $\rightarrow$ USPS. On these figures, we see that *Adaptation-Imputation* generates feature representations that overlap better between source and target examples per class than the adaptation counterparts (although *Adaptation-IgnoreComponent* does a good job at overlapping feature representations). This correlates with the accuracy performance on the target test set. Moreover we notice, as expected, that *Adaptation-IgnoreComponent* and *Adaptation-ZeroImputation* perform badly compared to *Adaptation-Full* which justifies the use of *Adaptation-Imputation* when confronted to missing non-stochastic data.
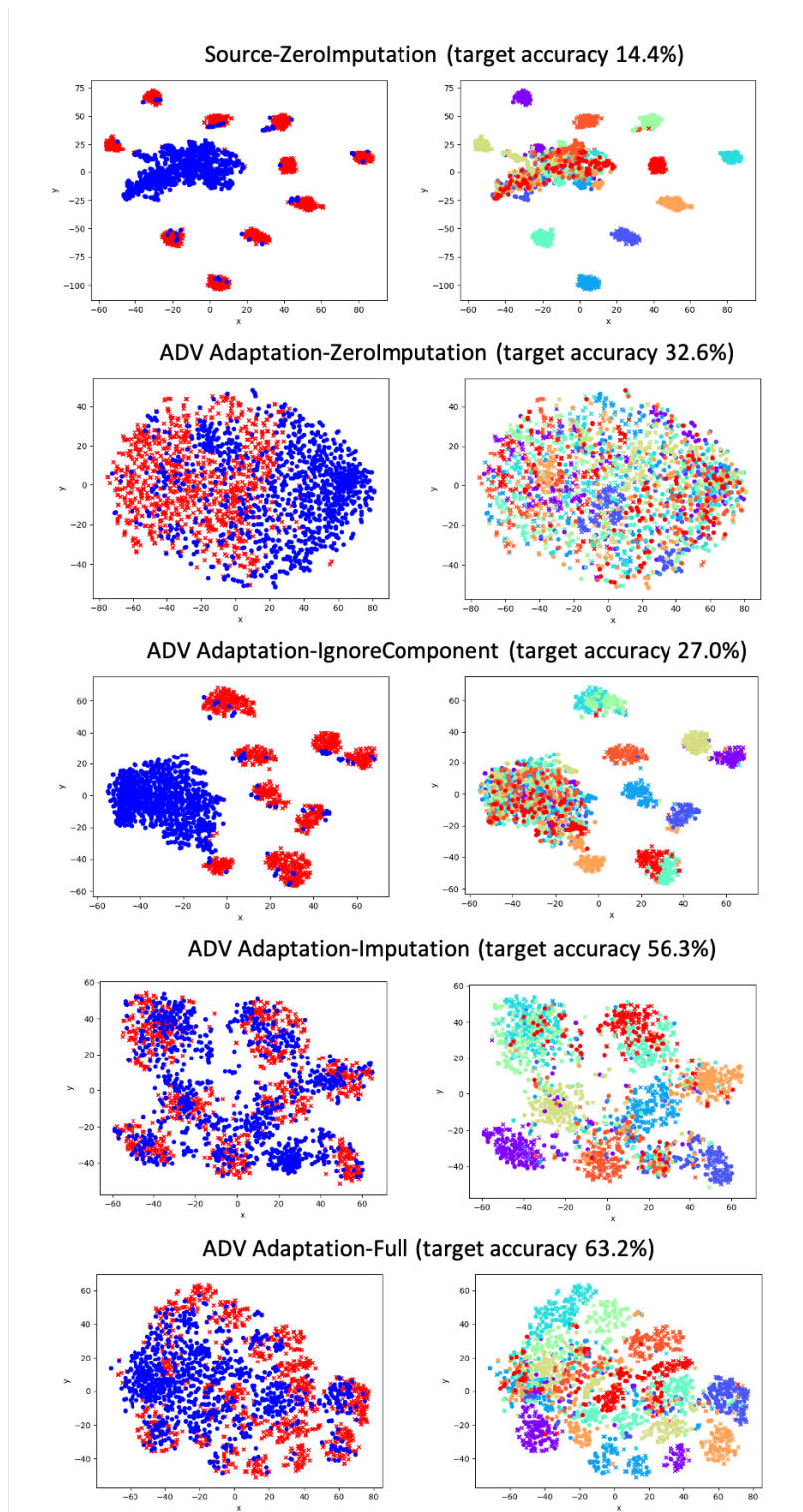
Figure A.3. – Embeddings for MNIST → MNIST-M dataset for ADV models on a batch. Figures on the left represent the source (red) and target (blue) clusters; Figures on the right represent the classes on source and target.

Source-ZeroImputation (target accuracy 14.4%)

OT Adaptation-ZeroImputation (target accuracy 18.8%)

OT Adaptation-IgnoreComponent (target accuracy 27.8%)

OT Adaptation-Imputation (target accuracy 28.6%)

OT Adaptation-Full (target accuracy 45.9%)
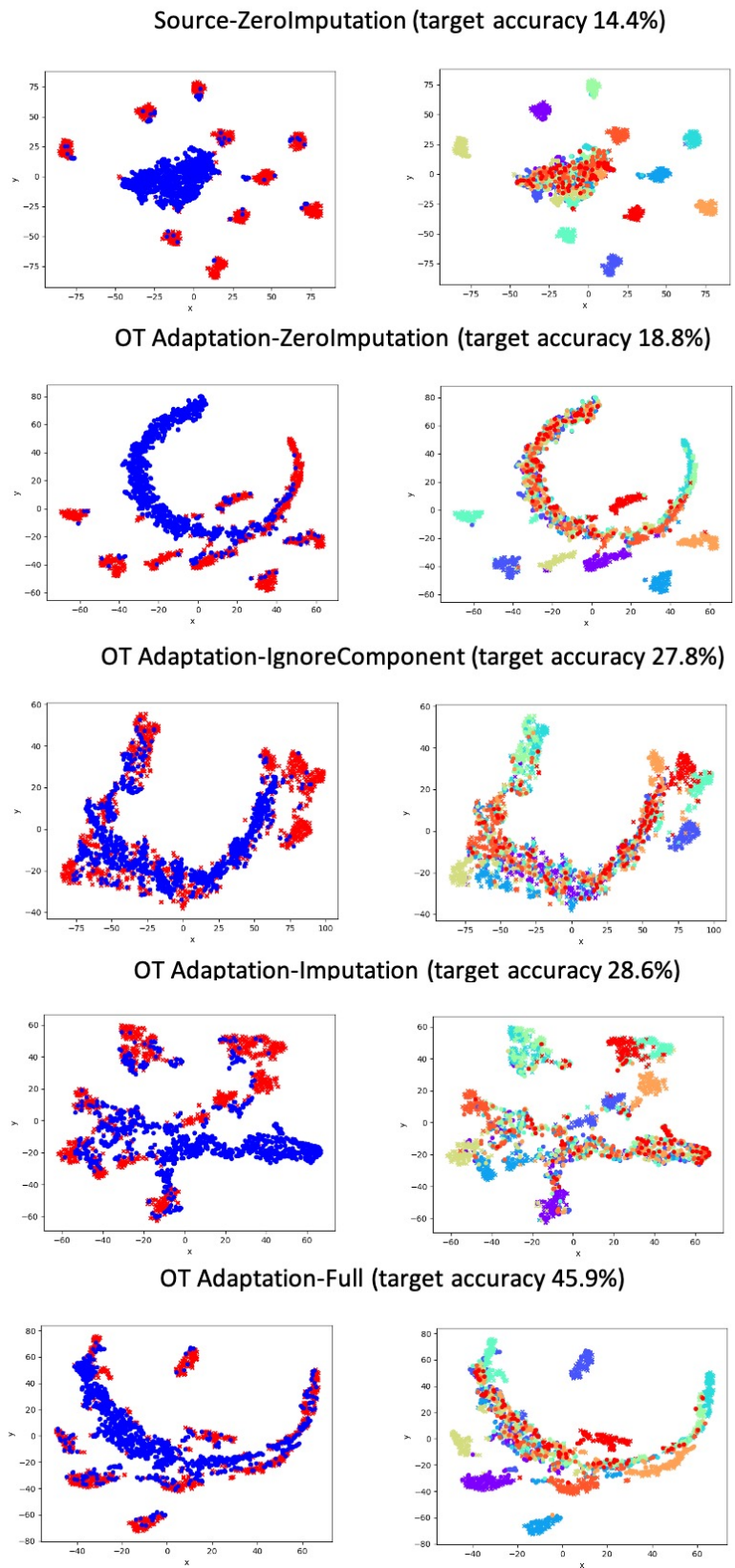
Figure A.4. – Embeddings for MNIST → MNIST-M dataset for OT models on a batch. Figures on the left represent the source (red) and target (blue) clusters; Figures on the right represent the classes on source and target.

Source-ZeroImputation (target accuracy 25.0%)

OT Adaptation-ZeroImputation (target accuracy 58.0%)

OT Adaptation-IgnoreComponent (target accuracy 62.4%)

OT Adaptation-Imputation (target accuracy 65.2%)

OT Adaptation-Full (target accuracy 91.5%)
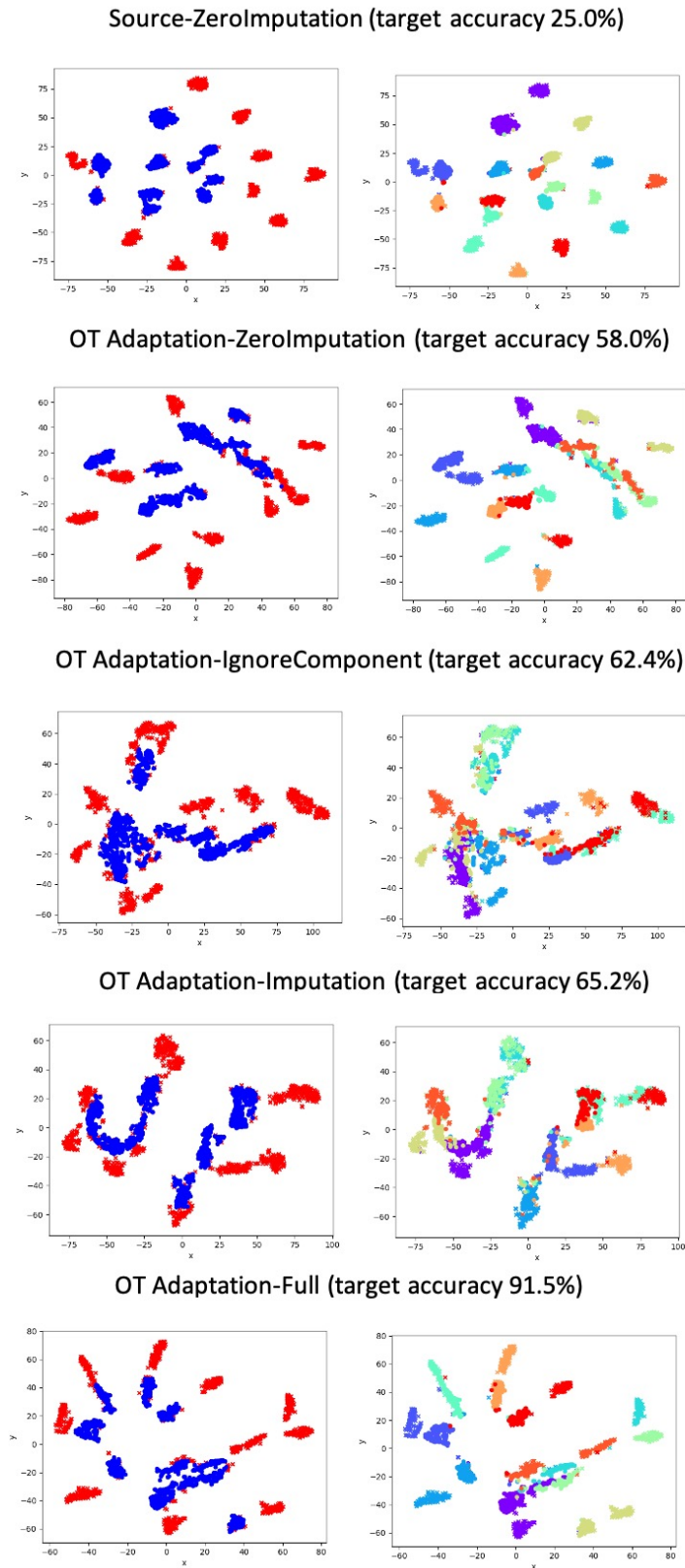
Figure A.5. – Embeddings for MNIST → USPS dataset for OT models on a batch. Figures on the left represent the source (red) and target (blue) clusters; Figures on the right represent the classes on source and target.

Appendix B

Supplementary Material of Chapter 4

B.1   Additional visualisation

We visualize in Figure B.2 how `OSTAR` maps source representations onto target ones under Generalized Target Shift (GeTarS). We note that `OSTAR` (i) maps source conditionals to target ones (blue and green points are matched c.f. left), (ii) matches conditionals of the same class ("v" and "o" of the same colour are matched c.f. right).

B.2   Additional results

We report our full results below. Aggregated results for balanced accuracy over a dataset and imbalance scenarios are reported in Table 4.1. Prefix "s" refers to subsampled datasets defined in Table B.6.

Additional ablation studies   We detail the full results for our ablation studies.

Table B.1. – Balanced accuracy (↑) over 10 runs. The best performing model is indicated in **bold**.

| Setting	Source	DANN	$WD_{\beta=0}$	$WD_{\beta=1}$	$WD_{\beta=2}$	MARSg	MARSc	IW-WD	OSTAR+IM	DI-Oracle
					Digits - MNIST→USPS					
balanced	86.94 ± 1.1	93.97 ± 0.9	94.42 ± 0.6	82.05 ± 6.3	75.14 ± 6.7	94.19 ± 1.8	$\mathbf{96.44 \pm 0.3}$	96.10 ± 0.3	$\mathbf{96.91 \pm 0.3}$	96.95 ± 0.2
mild	87.10 ± 2.0	93.23 ± 1.2	91.52 ± 0.6	84.07 ± 5.0	78.29 ± 7.2	94.78 ± 1.0	95.18 ± 0.9	94.72 ± 0.4	$\mathbf{96.18 \pm 1.0}$	97.22 ± 0.2
high	86.23 ± 2.8	93.03 ± 1.0	91.14 ± 0.7	85.15 ± 2.9	78.90 ± 3.8	94.50 ± 1.3	95.07 ± 0.6	94.60 ± 0.8	$\mathbf{96.06 \pm 0.6}$	96.87 ± 0.4
					Digits - USPS→MNIST					
balanced	63.01 ± 6.5	87.64 ± 1.7	90.84 ± 1.3	83.54 ± 3.0	77.00 ± 7.6	90.16 ± 2.5	93.37 ± 2.5	95.68 ± 0.6	$\mathbf{98.11 \pm 0.2}$	96.84 ± 0.2
mild	62.81 ± 2.3	87.00 ± 2.1	88.07 ± 1.2	77.83 ± 5.4	79.41 ± 7.4	89.93 ± 2.4	93.20 ± 2.8	92.73 ± 1.5	$\mathbf{97.44 \pm 0.5}$	95.75 ± 0.3
high	64.04 ± 5.4	86.37 ± 1.4	87.04 ± 1.4	79.17 ± 6.0	74.47 ± 7.4	88.24 ± 3.3	91.54 ± 0.9	90.81 ± 1.5	$\mathbf{97.08 \pm 0.6}$	95.87 ± 0.3
					VisDA12					
VisDA	48.63 ± 1.0	53.72 ± 0.9	57.40 ± 1.1	47.56 ± 0.8	36.21 ± 1.8	55.62 ± 1.6	55.33 ± 0.8	51.88 ± 1.6	$\mathbf{59.24 \pm 0.5}$	57.61 ± 0.3
sVisDA	42.46 ± 1.4	47.57 ± 0.9	47.32 ± 1.4	41.48 ± 1.6	31.83 ± 3.0	55.00 ± 1.9	51.86 ± 2.0	50.65 ± 1.5	$\mathbf{58.84 \pm 1.0}$	55.77 ± 1.1
					Office31					
sA-D	80.71 ± 0.5	82.39 ± 0.4	81.76 ± 0.4	75.98 ± 1.2	68.64 ± 2.4	$\mathbf{84.54 \pm 1.0}$	84.10 ± 0.8	81.83 ± 0.5	84.17 ± 0.7	87.74 ± 0.6
sD-W	89.08 ± 0.4	88.70 ± 0.2	88.98 ± 0.2	88.53 ± 0.2	88.97 ± 0.1	91.03 ± 0.4	90.76 ± 0.4	88.17 ± 0.3	$\mathbf{94.13 \pm 0.2}$	91.31 ± 0.2
sW-A	58.91 ± 0.2	58.87 ± 0.1	59.18 ± 0.2	60.70 ± 0.3	60.95 ± 0.2	63.94 ± 0.1	63.80 ± 0.3	60.25 ± 0.2	$\mathbf{69.99 \pm 0.1}$	63.92 ± 0.2
sW-D	95.64 ± 0.2	97.26 ± 0.3	97.13 ± 0.3	95.99 ± 0.3	95.57 ± 0.5	97.96 ± 0.1	$\mathbf{98.16 \pm 0.2}$	97.53 ± 0.2	98.47 ± 0.2	98.35 ± 0.0
sD-A	53.41 ± 0.9	57.45 ± 0.2	57.81 ± 0.2	58.24 ± 0.2	58.61 ± 0.3	62.12 ± 0.2	62.13 ± 0.4	60.03 ± 0.2	$\mathbf{65.00 \pm 0.5}$	62.57 ± 0.3
sA-W	69.23 ± 0.5	72.09 ± 0.5	72.60 ± 0.3	65.94 ± 0.9	61.64 ± 7.2	81.60 ± 0.5	81.05 ± 0.7	75.84 ± 0.7	$\mathbf{83.91 \pm 0.5}$	82.51 ± 0.5
					OfficeHome					
sA-C	44.44 ± 0.3	46.08 ± 0.3	41.74 ± 1.7	40.90 ± 0.8	39.22 ± 1.1	47.19 ± 0.3	46.94 ± 0.2	45.29 ± 0.1	$\mathbf{48.43 \pm 0.2}$	48.09 ± 0.2
sA-P	58.96 ± 0.3	59.96 ± 0.2	54.67 ± 1.8	52.18 ± 2.3	46.29 ± 1.4	62.17 ± 0.2	61.97 ± 0.2	59.46 ± 0.3	$\mathbf{69.52 \pm 0.4}$	63.59 ± 0.2
sA-R	67.10 ± 0.2	67.42 ± 0.2	65.40 ± 0.6	62.52 ± 1.7	60.51 ± 1.9	68.66 ± 0.3	68.62 ± 0.3	67.76 ± 0.2	$\mathbf{73.29 \pm 0.3}$	69.85 ± 0.1
sC-A	35.54 ± 2.3	35.47 ± 1.7	37.34 ± 2.0	36.81 ± 1.5	33.15 ± 2.3	46.03 ± 0.2	46.10 ± 0.2	44.18 ± 0.1	$\mathbf{46.47 \pm 0.3}$	46.94 ± 0.2
sC-P	52.48 ± 2.1	50.56 ± 0.9	53.53 ± 0.1	49.96 ± 1.4	44.67 ± 1.5	59.82 ± 0.1	59.82 ± 0.1	58.67 ± 0.1	$\mathbf{63.37 \pm 0.1}$	60.14 ± 0.1
sC-R	54.99 ± 1.7	54.22 ± 0.8	54.69 ± 0.5	51.34 ± 2.5	45.16 ± 3.5	62.69 ± 0.1	62.41 ± 0.1	60.74 ± 0.2	$\mathbf{63.12 \pm 0.2}$	62.80 ± 0.2
sP-A	38.10 ± 4.5	36.36 ± 3.3	48.24 ± 0.2	47.24 ± 0.3	48.20 ± 0.3	47.78 ± 0.3	46.10 ± 0.9	45.68 ± 0.3	$\mathbf{50.84 \pm 0.3}$	50.11 ± 0.4
sP-C	34.16 ± 4.6	33.00 ± 1.9	39.10 ± 0.2	40.36 ± 0.4	37.41 ± 0.3	42.41 ± 0.3	41.92 ± 0.3	38.22 ± 0.2	$\mathbf{44.15 \pm 0.3}$	43.10 ± 0.2
sP-R	66.28 ± 3.4	59.19 ± 0.8	70.01 ± 0.1	68.78 ± 0.2	66.61 ± 0.3	70.00 ± 0.4	69.37 ± 0.9	69.43 ± 0.4	$\mathbf{73.95 \pm 0.3}$	71.26 ± 0.4
sR-P	66.67 ± 5.4	70.97 ± 0.6	73.47 ± 0.3	72.66 ± 0.7	71.76 ± 0.7	72.62 ± 0.9	72.72 ± 1.1	72.90 ± 0.7	$\mathbf{75.58 \pm 0.6}$	74.17 ± 0.7
sR-A	48.59 ± 6.7	51.90 ± 1.1	$\mathbf{56.97 \pm 0.3}$	57.02 ± 0.5	55.38 ± 1.1	54.02 ± 0.7	53.37 ± 1.3	53.44 ± 0.6	56.28 ± 0.5	57.68 ± 0.6
sR-C	39.36 ± 2.5	45.33 ± 0.8	46.47 ± 0.4	47.11 ± 0.5	45.38 ± 1.3	45.81 ± 1.2	45.30 ± 1.2	42.66 ± 1.0	$\mathbf{49.07 \pm 0.9}$	47.86 ± 0.3

Table B.2. – Semi-supervised learning for OSTAR and balanced accuracy (↑). Best results are in **bold**.

Setting \ Objective	Eq. (CAL)	+ Eq. (SS)	+ Eq. (SSg)
	MNIST→USPS		
balanced	95.12 ± 0.6	96.68 ± 0.1	$\mathbf{96.91 \pm 0.3}$
mild	91.77 ± 1.2	95.39 ± 1.4	$\mathbf{96.18 \pm 1.0}$
high	88.55 ± 1.1	95.70 ± 0.8	$\mathbf{96.06 \pm 0.6}$
	USPS→MNIST		
balanced	88.19 ± 1.1	97.16 ± 0.3	$\mathbf{98.11 \pm 0.2}$
mild	88.34 ± 1.3	96.34 ± 0.2	$\mathbf{97.44 \pm 0.5}$
high	84.87 ± 2.3	95.61 ± 0.4	$\mathbf{97.08 \pm 0.6}$
	VisDA12		
original	50.37 ± 0.6	52.54 ± 0.3	$\mathbf{59.24 \pm 0.5}$
subsampled	49.05 ± 0.9	53.37 ± 0.6	$\mathbf{58.84 \pm 1.0}$
	Office31		
sA-D	81.52 ± 0.7	83.18 ± 0.2	$\mathbf{84.17 \pm 0.7}$
sD-W	89.94 ± 0.8	89.50 ± 0.8	$\mathbf{94.13 \pm 0.2}$
sW-A	59.62 ± 0.6	60.06 ± 0.4	$\mathbf{69.99 \pm 0.1}$
sW-D	96.39 ± 0.6	97.44 ± 0.2	$\mathbf{98.47 \pm 0.2}$
sD-A	54.38 ± 1.1	56.58 ± 0.6	$\mathbf{65.00 \pm 0.5}$
sA-W	75.30 ± 1.0	81.32 ± 0.8	$\mathbf{83.91 \pm 0.5}$
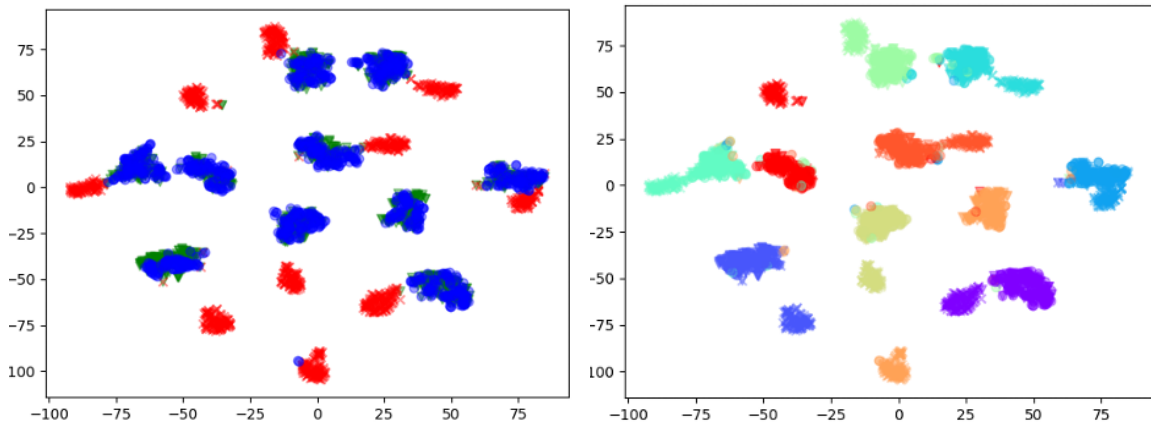
(a) MNIST→USPS balanced
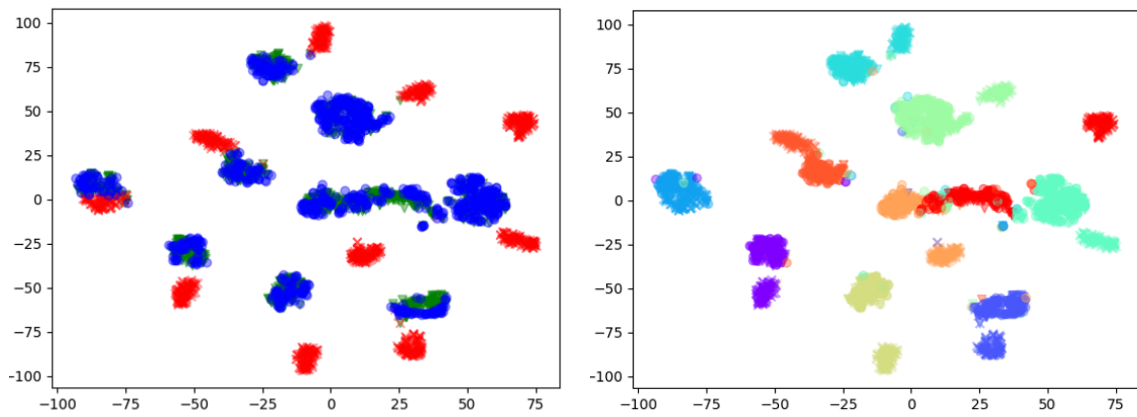
(b) MNIST→USPS subsampled high imbalance
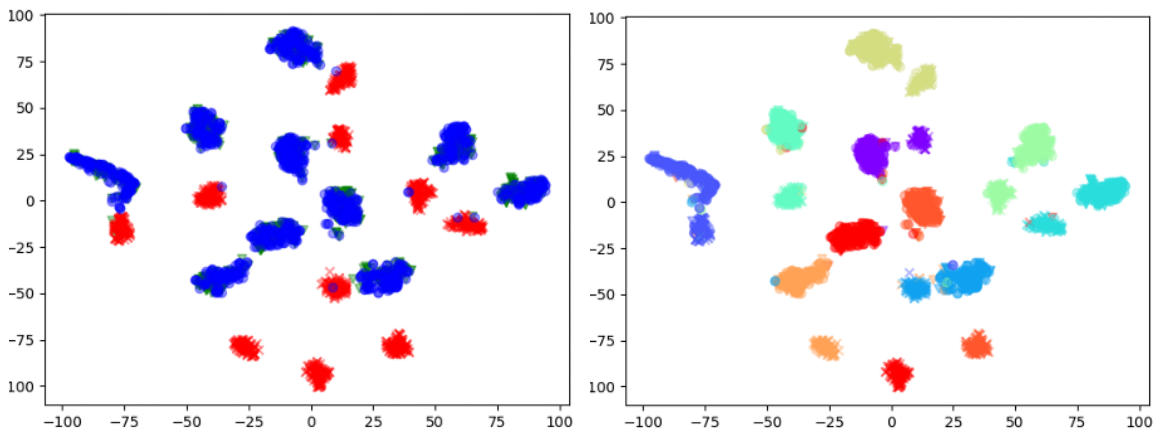
(c) USPS→MNIST balanced
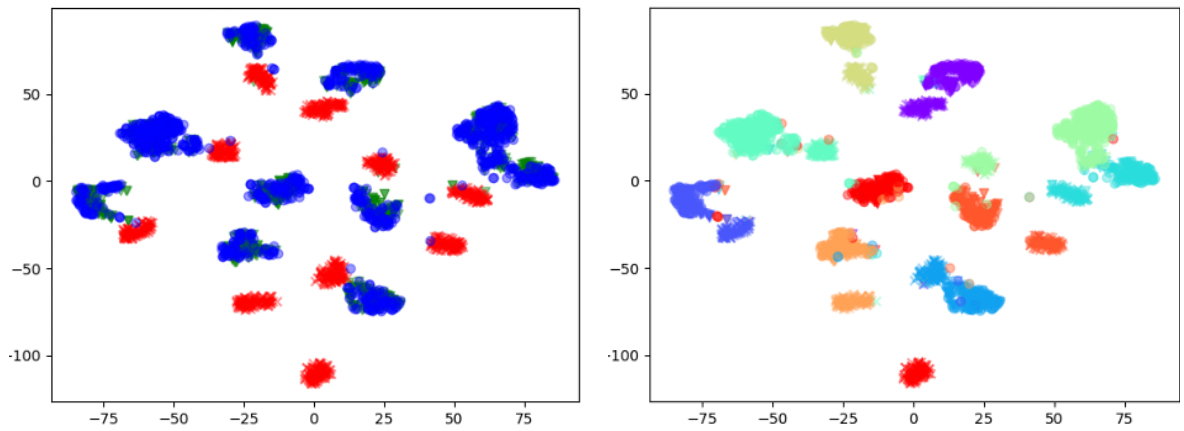
(a) USPS→MNIST subsampled high imbalance
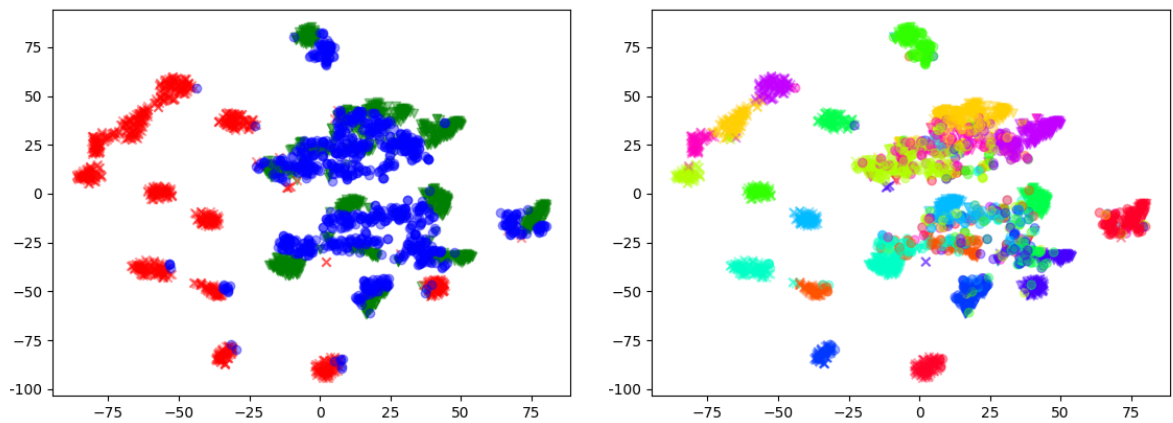
(b) VisDA
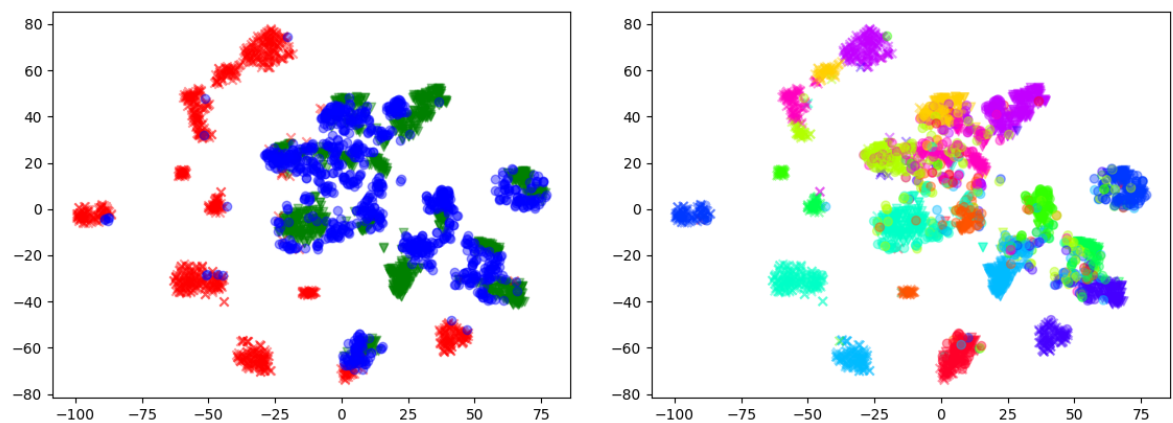
(c) VisDA subsampled

Figure B.2. – t-SNE feature visualizations for OSTAR on various datasets and label imbalance. Crosses "x" denote source samples, circles "o" target samples and triangles "v" transported source samples. On the left, source samples are red, target samples blue and transported source samples green. On the right, samples from the same class have the same colour regardless of domain.
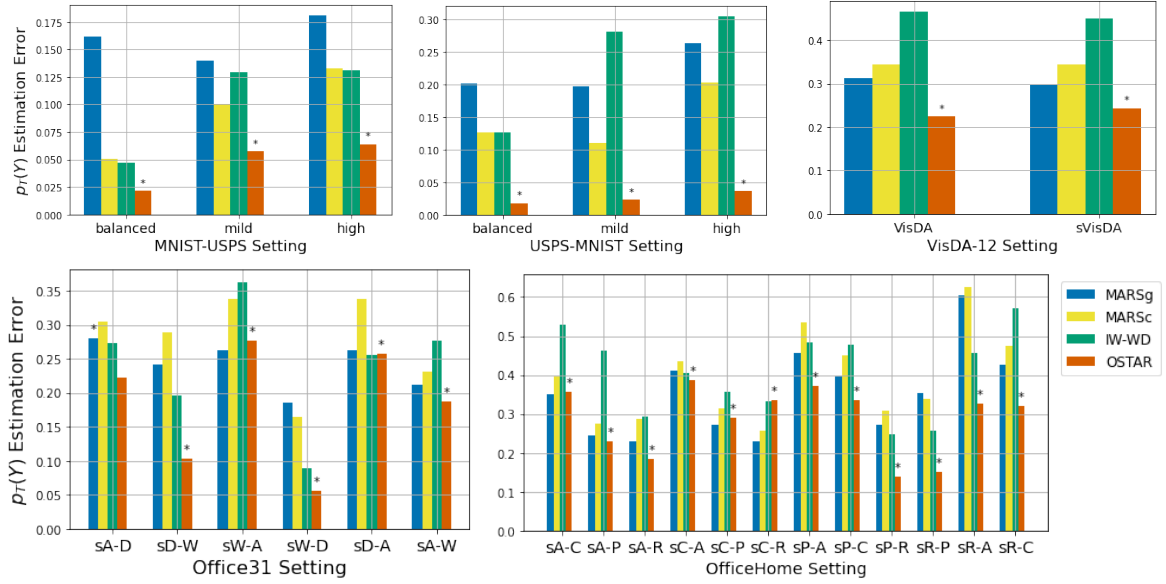
Figure B.3. – ℓ_1 estimation error of $\boldsymbol{p}_T^Y$ ($\downarrow$). The best model for balanced accuracy is indicated with "$\star$".

Table B.3. – IM for `MARSc`, `IW-WD` and `OSTAR` on balanced accuracy ($\uparrow$). Best results are in **bold**.

Setting \ Model	MARSc	MARSc+IM	IW-WD	IW-WD+IM	OSTAR	OSTAR+IM
			MNIST→USPS			
balanced	96.44 ± 0.3	$\mathbf{97.92 \pm 0.2}$	96.10 ± 0.3	$\mathbf{97.91 \pm 0.1}$	95.12 ± 0.6	96.91 ± 0.3
mild	95.18 ± 0.9	95.47 ± 1.1	94.72 ± 0.4	95.74 ± 0.6	91.77 ± 1.2	$\mathbf{96.18 \pm 1.0}$
high	95.07 ± 0.6	93.76 ± 0.5	94.60 ± 0.8	91.73 ± 0.6	88.55 ± 1.1	$\mathbf{96.06 \pm 0.6}$
			USPS→MNIST			
balanced	93.37 ± 2.5	93.03 ± 1.9	95.68 ± 0.6	96.17 ± 0.5	88.19 ± 1.1	$\mathbf{98.11 \pm 0.2}$
mild	93.20 ± 2.8	94.60 ± 1.7	92.73 ± 1.5	92.65 ± 1.0	88.34 ± 1.3	$\mathbf{97.44 \pm 0.5}$
high	91.54 ± 0.9	90.16 ± 2.0	90.81 ± 1.5	91.26 ± 1.1	84.87 ± 2.3	$\mathbf{97.08 \pm 0.6}$
			VisDA12			
VisDA	55.33 ± 0.8	57.57 ± 0.8	51.88 ± 1.6	57.63 ± 0.1	50.37 ± 0.6	$\mathbf{59.24 \pm 0.5}$
sVisDA	51.86 ± 2.0	57.06 ± 0.8	50.65 ± 1.5	57.62 ± 0.7	49.05 ± 0.9	$\mathbf{58.84 \pm 1.0}$
			Office31			
sW-A	63.80 ± 0.3	68.12 ± 0.5	60.25 ± 0.2	67.42 ± 0.8	59.62 ± 0.6	$\mathbf{69.99 \pm 0.1}$
sA-W	81.05 ± 0.7	81.83 ± 1.9	75.84 ± 0.7	82.34 ± 1.6	75.30 ± 1.0	$\mathbf{83.91 \pm 0.5}$
			OfficeHome			
sR-P	72.72 ± 1.1	$\mathbf{75.17 \pm 0.6}$	72.90 ± 0.7	74.94 ± 0.6	71.77 ± 0.4	$\mathbf{75.58 \pm 0.6}$
sR-A	53.37 ± 1.3	54.20 ± 1.3	53.44 ± 0.6	54.50 ± 1.1	55.15 ± 0.8	$\mathbf{56.28 \pm 0.5}$
sR-C	45.30 ± 1.2	48.17 ± 1.2	42.66 ± 1.0	47.93 ± 1.7	43.02 ± 3.1	$\mathbf{49.07 \pm 0.9}$

Table B.4. – Best value over training epochs of term (A) (↓), term (C) (↓) and term (L) (↓) without and with IM in `OSTAR`. Best results are in **bold**. Terms (A) and (L) are computed with the primal formulation of OT using the POT package `https://pythonot.github.io/`.

	Alignment				Discriminativity	
	Term (A)		Term (L)		Term (C)	
Setting	`OSTAR`	`OSTAR+IM`	`OSTAR`	`OSTAR+IM`	`OSTAR`	`OSTAR+IM`
	MNIST→USPS					
balanced	49.83	**16.56**	1.45	**0.18**	2.19×10^{-3}	$\mathbf{0.918 \times 10^{-3}}$
mild	39.31	**19.13**	10.78	**0.24**	1.65×10^{-3}	$\mathbf{0.931 \times 10^{-3}}$
high	38.60	**21.10**	12.78	**0.74**	1.76×10^{-3}	$\mathbf{0.510 \times 10^{-3}}$
	USPS→MNIST					
balanced	235.43	**86.65**	7.08	**0.52**	4.46×10^{-3}	$\mathbf{0.495 \times 10^{-3}}$
mild	188.67	**104.66**	20.99	**1.05**	3.98×10^{-3}	$\mathbf{0.399 \times 10^{-3}}$
high	181.64	**123.83**	21.62	**0.82**	4.51×10^{-3}	$\mathbf{0.616 \times 10^{-3}}$

Table B.5. – Detailed analysis of the impact of λ_{OT} on balanced accuracy (↑). Best results are in **bold**.

	MNIST→USPS - initialization gain 0.02						
Shift \ λ_{OT}	$\lambda_{OT} = 0$	$\lambda_{OT} = 10^{-3}$	$\lambda_{OT} = 10^{-2}$	$\lambda_{OT} = 10^{-1}$	$\lambda_{OT} = 1$	$\lambda_{OT} = 10^4$	Source
balanced	94.92 ± 0.6	$\mathbf{96.02 \pm 0.2}$	95.12 ± 0.6	89.76 ± 1.2	91.95 ± 1.2	87.03 ± 1.8	86.02 ± 1.4
mild	88.28 ± 1.5	88.63 ± 1.3	$\mathbf{91.77 \pm 1.2}$	90.17 ± 1.7	88.51 ± 1.2	88.42 ± 1.6	89.08 ± 0.5
high	85.24 ± 1.6	85.38 ± 1.4	88.55 ± 1.1	$\mathbf{89.10 \pm 1.2}$	88.82 ± 1.1	86.94 ± 1.1	86.73 ± 1.9

	MNIST→USPS high imbalance					
Gain \ λ_{OT}	$\lambda_{OT} = 0$	$\lambda_{OT} = 10^{-3}$	$\lambda_{OT} = 10^{-2}$	$\lambda_{OT} = 10^{-1}$	$\lambda_{OT} = 1$	$\lambda_{OT} = 10^4$
0.02	85.24 ± 1.6	85.38 ± 1.4	88.55 ± 1.1	$\mathbf{89.10 \pm 1.2}$	88.82 ± 1.1	86.94 ± 1.1
0.1	84.62 ± 2.3	85.84 ± 1.1	$\mathbf{88.41 \pm 1.3}$	$\mathbf{88.79 \pm 1.2}$	87.90 ± 1.2	87.10 ± 1.0
0.3	83.11 ± 2.4	84.45 ± 1.4	89.41 ± 1.6	$\mathbf{91.00 \pm 1.3}$	89.65 ± 0.7	86.23 ± 1.8

B.3 Details on OT

Background OT was introduced to find a transportation map minimizing the cost of displacing mass from one configuration to another (Villani 2008). For a comprehensive introduction, we refer to Peyré et al. 2019. Formally, let α and β be absolutely continuous distributions compactly supported in $\mathbb{R}^d$ and $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ a cost function. Consider a map $\phi : \mathbb{R}^d \to \mathbb{R}^d$ that satisfies $\phi_\#\alpha = \beta$, i.e. that pushes α to β. We remind that for a function f, $f_\#\rho$ is the push-forward measure $f_\#\rho(B) = \rho(f^{-1}(B))$, for all measurable set B. The total transportation cost depends on the contributions of costs for transporting each point $\mathbf{x}$ to $\phi(\mathbf{x})$ and the Monge OT problem is:

$$
\min_\phi \mathcal{C}_{\text{monge}}(\phi) = \int_{\mathbb{R}^d} c(\mathbf{x}, \phi(\mathbf{x}))d\alpha(\mathbf{x}) \tag{B.1}
$$
$$
\text{s.t. } \phi_\#\alpha = \beta
$$

$c(\mathbf{x}, \mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_2^p$ induces the p-Wasserstein distance, $\mathcal{W}_p(\alpha, \beta) = \min_{\phi_\#\alpha=\beta} \mathcal{C}_{\text{monge}}(\phi)^{1/p}$. When $p = 1$, $\mathcal{W}_1$ can be expressed in the dual form $\mathcal{W}_1(\alpha, \beta) = \sup_{\|v\|_L \leq 1} \mathbb{E}_{\mathbf{x}\sim\alpha}v(\mathbf{x}) - \mathbb{E}_{\mathbf{y}\sim\beta}v(\mathbf{y})$ where $\|v\|_L$ is the Lipschitz constant of function v.

Relationship between Eq. (OT) and the Monge OT problem Eq. (B.1) Eq. (OT) is the extension of Eq. (B.1) to the setting where $\boldsymbol{p}_S^Y \neq \boldsymbol{p}_T^Y$ with $\boldsymbol{p}_T^Y$ unknown. This extension aims at matching conditional distributions regardless of how their mass differ and learns a weighting term $\boldsymbol{p}_N^Y$ for source conditional distributions which addresses label shift settings. When $\boldsymbol{p}_S^Y = \boldsymbol{p}_T^Y$, these two formulations are equivalent under Assumption 5 if we fix $\boldsymbol{p}_N^Y = \boldsymbol{p}_S^Y = \boldsymbol{p}_T^Y$.

B.4 Discussion

Our four assumptions are required for deriving our theoretical guarantees. All GeTarS papers that propose theoretical guarantees also rely on a set of assumptions, either explicit or implicit. Assumptions 5 and 8 are common to several papers. Assumption 5 is easily met in practice since it could be forced by training a classifier on the source labels. Assumption 8 is said to be met in high dimensions (Redko et al. 2019; Garg et al. 2020). Assumption 7 says that source and target clusters from the same class are globally (there is a sum in the condition) closer one another than clusters from two different classes. This assumption is required to cope with the absence of target labels. Because of the sum, it could be considered as a reasonable hypothesis. It is milder than the hypotheses made in

papers offering guarantees similar to ours (Zhang et al. 2013; Gong et al. 2016). Assumption 6 is new to this paper. It states that ϕ maps a j source conditional to a unique k target conditional i.e. it guarantees that mass of a source conditional will be entirely transferred to a target conditional and will not be split across several target conditionals. This property is key to show that OSTAR matches source conditionals and their corresponding target conditionals, which is otherwise very difficult to guarantee under GeTarS as both target conditionals and proportions are unknown. This assumption has some restrictions, despite being milder than existing assumptions in GeTarS. First, mass from a source conditional might in practice be split across several target conditionals. In practice, our optimization problem in Eq. (CAL) mitigates this problem by learning how to reweight source conditionals to improve alignment. We could additionally apply Laplacian regularization (Ferradans et al. 2013; Carreira-Perpiñán et al. 2014) into our OT map ϕ but this was not developed in the paper. Second, it assumes a closed-set UDA problem i.e. label domains are the same $\mathcal{Y}_S = \mathcal{Y}_T$. The setting where $\mathcal{Y}_S \neq \mathcal{Y}_T$ is more realistic in large scale image pretraining and is another interesting follow-up. Note that OSTAR can address empirically open-set UDA ($\mathcal{Y}_T \subset \mathcal{Y}_S$) by simply applying L1 regularization on $\boldsymbol{p}_N^Y$ in our objective function Eq. (CAL). This forces sparsity and allows "loosing" mass when mapping a source conditional. It avoids the unwanted negative transfer setting when source clusters, with labels absent on the target, are aligned with target clusters.

B.5   Proofs

Proposition (4.2). For any encoder g which defines $\mathcal{Z}$ satisfying Assumption 5, 6, 7, 8, there is an unique solution $(\phi, \boldsymbol{p}_N^Y)$ to Eq. (OT) and $\phi_\#(p_S(Z|Y)) = p_T(Z|Y)$ and $\boldsymbol{p}_N^Y = \boldsymbol{p}_T^Y$.

Proof. Fixing $\mathcal{Z}$ satisfying Assumptions 5, 7 and 8, we first show that there exists a solution $(\phi, \boldsymbol{p}_N^Y)$ to Eq. (OT). Following Brenier 1991 as $\mathcal{Z} \subset \mathbb{R}^d$, we can find K unique Monge maps $\{\widehat{\phi}^{(k)}\}_{k=1}^K$ s.t. $\forall k \ \widehat{\phi}_\#^{(k)}(p_S(Z|Y = k)) = p_T(Z|Y = k)$ with respective transport costs $\mathcal{W}_1(p_S(Z|Y = k), p_T(Z|Y = k))$. Let's define $\widehat{\phi}$ as $\forall k \ \widehat{\phi}_{|\mathcal{Z}_S^{(k)}} = \widehat{\phi}^{(k)}$ where $\cup_{k=1}^K \mathcal{Z}_S^{(k)}$ is the partition of $\mathcal{Z}_S$ in Assumption 5. $(\widehat{\phi}, \boldsymbol{p}_T^Y)$ satisfies the equality constraint to Eq. (OT), thus we easily deduce existence.

Now, let $(\phi, \boldsymbol{p}_N^Y)$ be a solution to Eq. (OT), let's show unicity. We first show that $\phi = \widehat{\phi}$. Under Assumption 6, Eq. (OT) is the Monge formulation of the optimal assignment problem between $\{p_S(Z|Y = k)\}_{k=1}^K$ and $\{p_T(Z|Y = k)\}_{k=1}^K$ with $\mathbf{C}$ the cost matrix defined by $\mathbf{C}_{ij} = \mathcal{W}_2(p_S(Z|Y = i), p_T(Z|Y = j))$. At the optimum, the transport cost is related to the Wasserstein distance between source conditionals and their corresponding target conditionals i.e. $\mathcal{C}(\phi) = \sum_{k=1}^K \mathcal{W}_2(p_S(Z|Y =$

k), $\phi_{\#}(p_S(Z|Y = k)))$. Suppose $\exists(i,j), j \neq i$ s.t. $\phi_{\#}(p_S(Z|Y = i)) = p_T(Z|Y = j)$ and $\phi_{\#}(p_S(Z|Y = j)) = p_T(Z|Y = i)$ and $\forall k \neq i, j, \phi_{\#}(p_S(Z|Y = k)) = p_T(Z|Y = k)$. Assumption 7 implies $\sum_{k=1}^{K} \mathcal{W}_2(p_S(Z|Y = k), p_T(Z|Y = k)) \leq \sum_{k \neq i,j} \mathcal{W}_2(p_S(Z|Y = k), p_T(Z|Y = k)) + \mathcal{W}_2(p_S(Z|Y = i), p_T(Z|Y = j)) + \mathcal{W}_2(p_S(Z|Y = j), p_T(Z|Y = i))$. Thus $C(\widehat{\phi}, \mathcal{Z}) \lneq C(\phi, \mathcal{Z})$ whereas ϕ, solution to Eq. (OT), has minimal transport cost. Thus $\phi = \widehat{\phi}$.

Now let's show $\boldsymbol{p}_N^Y = \boldsymbol{p}_T^Y$ under Assumption 8. We inject $\phi_{\#}(p_S(Z|Y)) = p_T(Z|Y)$ into Eq. (OT),

$$\sum_{k=1}^{K} \boldsymbol{p}_N^{Y=k} p_T(Z|k) = \sum_{k=1}^{K} \boldsymbol{p}_T^{Y=k} p_T(Z|k) \Leftrightarrow \sum_{k=1}^{K} (\boldsymbol{p}_N^{Y=k} - \boldsymbol{p}_T^{Y=k}) p_T(Z|k) = 0 \Leftrightarrow \boldsymbol{p}_N^Y = \boldsymbol{p}_T^Y$$

$\square$

Theorem (4.3). Given a fixed encoder g defining a latent space $\mathcal{Z}$, two domains N and T satisfying cyclical monotonicity in $\mathcal{Z}$, assuming that we have $\forall k, \boldsymbol{p}_N^{Y=k} > 0$, then $\forall f_N \in \mathcal{H}$ where $\mathcal{H}$ is a set of M-Lipschitz continuous functions over $\mathcal{Z}$,

$$\mathcal{E}_T^g(f_N) \leq \underbrace{\mathcal{E}_N^g(f_N)}_{\text{Classification } (C)} + \underbrace{\frac{2M}{\min_{k=1}^{K} \boldsymbol{p}_N^{Y=k}} \mathcal{W}_1\Big(p_N(Z), p_T(Z)\Big)}_{\text{Alignment } (A)}$$

$$+ \underbrace{2M\Big(1 + \frac{1}{\min_{k=1}^{K} \boldsymbol{p}_N^{Y=k}}\Big) \mathcal{W}_1\Big(\sum_{k=1}^{K} \boldsymbol{p}_N^{Y=k} p_T(Z|Y = k), \sum_{k=1}^{K} \boldsymbol{p}_T^{Y=k} p_T(Z|Y = k)\Big)}_{\text{Label } (L)}$$

$$(4.2)$$

Proof. We first recall that $\mathcal{E}_D^g(f_N) = \mathbb{E}_{(\mathbf{z},y) \in p_D(Z,Y)} \mathcal{L}(f_N(\mathbf{z}), y)$ where $\mathcal{L}$ is the 0/1 loss. For conciseness, $\forall \mathbf{z} \in \mathcal{Z}, \boldsymbol{p}_T^{\mathbf{z}|Y}, \boldsymbol{p}_N^{\mathbf{z}|Y}, \mathcal{L}^{\mathbf{z},k}$ will refer to the vector of $[p_T(\mathbf{z}|k)]_{k=1}^{K}$,

$[p_N(\mathbf{z}|k)]_{k=1}^K$, $[\mathcal{L}(f_N(\mathbf{z}), k)]_{k=1}^K$ respectively. In the following, $\odot$ denotes the element-wise product operator between two vectors of the same size.

$$\forall f_N, \mathcal{E}_T^g(f_N) = \mathcal{E}_N^g(f_N) + \mathcal{E}_T^g(f_N) - \mathcal{E}_N^g(f_N)$$

$$\leq \mathcal{E}_N^g(f_N) + \int_{\mathcal{Z}} \sum_{k=1}^K \Big(p_T(\mathbf{z}, k) - p_N(\mathbf{z}, k) \Big) \times \mathcal{L}\Big(f_N(\mathbf{z}), k \Big) d\mathbf{z} dy$$

$$\leq \mathcal{E}_N^g(f_N) + \int_{\mathcal{Z}} \sum_{k=1}^K \Big[\boldsymbol{p}_T^{Y=k} p_T(\mathbf{z}|k) - \boldsymbol{p}_N^{Y=k} p_N(\mathbf{z}|k) \Big] \times \mathcal{L}\Big(f_N(\mathbf{z}), k \Big) d\mathbf{z}$$

$$\leq \mathcal{E}_N^g(f_N) + \int_{\mathcal{Z}} \boldsymbol{p}_T^{Y\top} \Big(\boldsymbol{p}_T^{\mathbf{z}|Y} \odot \mathcal{L}^{\mathbf{z},k} \Big) - \boldsymbol{p}_N^{Y\top} \Big(\boldsymbol{p}_N^{\mathbf{z}|Y} \odot \mathcal{L}^{\mathbf{z},k} \Big) d\mathbf{z}$$

$$\leq \mathcal{E}_N^g(f_N) + \int_{\mathcal{Z}} \boldsymbol{p}_T^{Y\top} \Big(\boldsymbol{p}_T^{\mathbf{z}|Y} \odot \mathcal{L}^{\mathbf{z},k} \Big) - \boldsymbol{p}_N^{Y\top} \Big(\boldsymbol{p}_T^{\mathbf{z}|Y} \odot \mathcal{L}^{\mathbf{z},k} \Big) + \boldsymbol{p}_N^{Y\top} \Big(\boldsymbol{p}_T^{\mathbf{z}|Y} \odot \mathcal{L}^{\mathbf{z},k} \Big) - \boldsymbol{p}_N^{Y\top} \Big(\boldsymbol{p}_N^{\mathbf{z}|Y} \odot \mathcal{L}^{\mathbf{z},k} \Big) d\mathbf{z}$$

$$\leq \mathcal{E}_N^g(f_N) + \int_{\mathcal{Z}} \Big(\boldsymbol{p}_T^{Y\top} - \boldsymbol{p}_N^{Y\top} \Big) \Big(\boldsymbol{p}_T^{\mathbf{z}|Y} \odot \mathcal{L}^{\mathbf{z},k} \Big) + \boldsymbol{p}_N^{Y\top} \Big((\boldsymbol{p}_T^{\mathbf{z}|Y} - \boldsymbol{p}_N^{\mathbf{z}|Y}) \odot \mathcal{L}^{\mathbf{z},k} \Big) d\mathbf{z}$$

$$\leq \mathcal{E}_N^g(f_N) + \int_{\mathcal{Z}} \Big(\boldsymbol{p}_T^{Y\top} - \boldsymbol{p}_N^{Y\top} \Big) \Big(\boldsymbol{p}_T^{\mathbf{z}|Y} \odot \mathcal{L}^{\mathbf{z},k} \Big) d\mathbf{z} + \int_{\mathcal{Z}} \boldsymbol{p}_N^{Y\top} \Big((\boldsymbol{p}_T^{\mathbf{z}|Y} - \boldsymbol{p}_N^{\mathbf{z}|Y}) \odot \mathcal{L}^{\mathbf{z},k} \Big) d\mathbf{z}$$

We now introduce a preliminary result from Shen et al. 2018. $\forall f_N \in \mathcal{H}$ M-Lipschitz continuous,

$$\mathcal{E}_N^g(f_N) - \mathcal{E}_T^g(f_N) \leq 2M \cdot \mathcal{W}_1(p_N^g(Z), p_T^g(Z))$$

Assuming that h is M-Lipschitz continuous we apply this result in the following

$$\int_{\mathcal{Z}} \Big(\boldsymbol{p}_T^{Y\top} - \boldsymbol{p}_N^{Y\top} \Big) \Big(\boldsymbol{p}_T^{\mathbf{z}|Y} \odot \mathcal{L}^{\mathbf{z},k} \Big) d\mathbf{z} = \int_{\mathcal{Z}} \sum_{k=1}^K \Big(\boldsymbol{p}_T^{Y=k} - \boldsymbol{p}_N^{Y=k} \Big) p_T(\mathbf{z}|k) \times \mathcal{L}\Big(f_N(\mathbf{z}), k \Big) d\mathbf{z}$$

$$= \mathcal{E}_T^g(f_N) - \mathcal{E}_{\widetilde{T}}^g(f_N) \quad \text{where } p_{\widetilde{T}}(Z) = \sum_{k=1}^K \boldsymbol{p}_N^{Y=k} p_T(Z|k)$$

$$\leq 2M \cdot \mathcal{W}_1(p_{\widetilde{T}}(Z), p_T(Z))$$

$$\int_{\mathcal{Z}} \boldsymbol{p}_N^{Y\top} \Big((\boldsymbol{p}_T^{\mathbf{z}|Y} - \boldsymbol{p}_N^{\mathbf{z}|Y}) \odot \mathcal{L}^{\mathbf{z},k} \Big) d\mathbf{z} = \int_{\mathcal{Z}} \sum_{k=1}^K \boldsymbol{p}_N^{Y=k} \Big(p_T(\mathbf{z}|k) - p_N(\mathbf{z}|k) \Big) \times \mathcal{L}\Big(f_N(\mathbf{z}), k \Big) d\mathbf{z}$$

$$\leq \sum_{k=1}^K \int_{\mathcal{Z}} \Big(p_T(\mathbf{z}|k) - p_N(\mathbf{z}|k) \Big) \times \mathcal{L}\Big(f_N(\mathbf{z}), k \Big) d\mathbf{z} \quad \forall k \ \boldsymbol{p}_N^{Y=k} \leq 1$$

$$\leq 2M \sum_{k=1}^K \mathcal{W}_1\Big(p_T(Z|k), p_N(Z|k) \Big)$$

Thus, $\forall f_N$ M-Lipschitz continuous

$$\mathcal{E}_T^g(f_N) \le \mathcal{E}_N^g(f_N) + 2M \times \sum_{k=1}^K \mathcal{W}_1\Big(p_T(Z|k), p_N(Z|k)\Big) + 2M \times \mathcal{W}_1\Big(p_{\widetilde{T}}(Z), p_T(Z)\Big)$$

We rewrite the second term to involve latent marginals. Proposition 2 in Rakotomamonjy et al. 2021 shows that under cyclical monotonicity, if $\forall k, \boldsymbol{p}_N^{Y=k} > 0$,

$$\mathcal{W}_1\Big(\sum_{k=1}^K \boldsymbol{p}_N^{Y=k} p_T(Z|k), p_N(Z)\Big) = \sum_{k=1}^K \boldsymbol{p}_N^{Y=k} \mathcal{W}_1\Big(p_N(Z|k), p_T(Z|k)\Big)$$

This allows to write

$$\min_{k=1}^K \boldsymbol{p}_N^{Y=k} \sum_{k=1}^K \mathcal{W}_1\Big(p_N(Z|k), p_T(Z|k)\Big) \le \sum_{k=1}^K \boldsymbol{p}_N^{Y=k} \mathcal{W}_1\Big(p_N(Z|k), p_T(Z|k)\Big)$$

$$= \mathcal{W}_1\Big(\sum_{k=1}^K \boldsymbol{p}_N^{Y=k} p_T(Z|k), p_N(Z)\Big) = \mathcal{W}_1\Big(p_{\widetilde{T}}(Z), p_N(Z)\Big)$$

We then use the triangle inequality for the Wasserstein distance $\mathcal{W}_1$

$$\mathcal{E}_T^g(f_N) \le \mathcal{E}_N^g(f_N) + \frac{2M}{\min_{k=1}^K \boldsymbol{p}_N^{Y=k}} \mathcal{W}_1\Big(p_{\widetilde{T}}(Z), p_N(Z)\Big) + 2M \times \mathcal{W}_1\Big(p_{\widetilde{T}}(Z), p_T(Z)\Big)$$

$$\le \mathcal{E}_N^g(f_N) + \frac{2M}{\min_{k=1}^K \boldsymbol{p}_N^{Y=k}} \mathcal{W}_1\Big(p_N(Z), p_T(Z)\Big) + 2M(1 + \frac{1}{\min_{k=1}^K \boldsymbol{p}_N^{Y=k}}) \mathcal{W}_1\Big(p_{\widetilde{T}}(Z), p_T(Z)\Big)$$

$\square$

Derivation of the reweighted classification loss (C) $\mathcal{E}_N^g(f_N)$ Let ℓ_{ce} be the Cross Entropy (CE) loss. Given a classifier h, feature extractor g and domain N, the mapping of domain S by $(\phi, \boldsymbol{p}_N^Y)$,

$$\mathcal{E}_N^g(f_N) = \int_{\mathcal{Z},\mathcal{Y}} p_N^\phi(\mathbf{z}, y) \ell_{ce}\Big(f_N(\mathbf{z}), y\Big) d\mathbf{z} dy = \int_{\mathcal{Z},\mathcal{Y}} \boldsymbol{p}_N^{Y=y} p_N^\phi(\mathbf{z}|y) \ell_{ce}\Big(f_N(\mathbf{z}), y\Big) d\mathbf{z} dy$$

$$= \int_{\mathcal{Z},\mathcal{Y}} \frac{\boldsymbol{p}_N^{Y=y}}{\boldsymbol{p}_S^{Y=y}} \boldsymbol{p}_S^{Y=y} p_N^\phi(\mathbf{z}|y) \ell_{ce}\Big(f_N(\mathbf{z}), y\Big) d\mathbf{z} dy = \int_{\mathcal{Z},\mathcal{Y}} \frac{\boldsymbol{p}_N^{Y=y}}{\boldsymbol{p}_S^{Y=y}} \boldsymbol{p}_S^{Y=y} \phi_\#(p_S(\mathbf{z}|y)) \ell_{ce}\Big(f_N(\mathbf{z}), y\Big) d\mathbf{z} dy$$

B.6   Pseudo-code and runtime / complexity analysis

We detail in Algorithm B.1 our pseudo-code and in Algorithm B.2 how we minimize Eq. (CAL) with respect to (ϕ, f_N) using the dual form of Wasserstein-1 distance Eq. (4.6). Our method is based on a standard backpropagation strategy with gradient descent and uses gradient penalty (Gulrajani et al. 2017).

Algorithm B.1 Training and inference procedure for OSTAR

Training:

$\mathcal{D}_S = \{\mathbf{x_S^{(i)}}, y_S^{(i)}\}_{i=1}^n$, $\mathcal{D}_T = \{\mathbf{x_T^{(i)}}\}_{i=1}^m$, $\mathcal{Z}_{\mathcal{D}_N} = \{\phi \circ g(\mathbf{x_S^{(i)}}), y_S^{(i)}\}_{i=1}^n$, $\mathcal{Z}_{\mathcal{D}_T} = \{g(\mathbf{x_T^{(i)}})\}_{i=1}^m$

$f_S, f_N \in \mathcal{H}$ classifiers; g feature extractor; ϕ latent domain-mapping, v critic.

N_e: number of epochs, N_u: epoch to update $\boldsymbol{p}_N^Y$, N_g: epoch to update g

1: Train f_S, g on $\mathcal{D}_S$ to minimize source classification loss   ▷ Eq. (4.3)

2: Initialize $\boldsymbol{p}_N^Y = \dfrac{1}{K}\mathbf{1}_K$

3: **for** $n_{epoch} \leq N_e$ **do**

4:     **if** $n_{epoch} \mod N_u = 0$ **then**

5:         Compute $\boldsymbol{p}_N^Y$ with estimator in Lipton et al. 2018 on $(\mathcal{Z}_{\mathcal{D}_N}, \mathcal{Z}_{\mathcal{D}_T})$ ▷ Eq. (CAL) w.r.t.. $\boldsymbol{p}_N^Y$

6:         Average $\boldsymbol{p}_N^Y$ with cumulative moving average

7:     **end if**

8:     **if** $n_{epoch} \leq N_g$ **then** Train ϕ, v, f_N with $(\mathcal{D}_S, \mathcal{D}_T)$ ▷ Eq. (CAL) + Eq. (SS) w.r.t.. ϕ, f_N

9:     **else** Train ϕ, v, f_N, g with $(\mathcal{D}_S, \mathcal{D}_T)$ ▷ Eq. (CAL) + Eq. (SSg) w.r.t.. ϕ, f_N

10:     **end if**

11: **end for**

Inference: Score $\mathbf{x_T}$ with $f_N \circ g(\mathbf{x_T})$

Complexity / Runtime analysis   In practice on USPS → MNIST, the runtimes in seconds on a NVIDIA Tesla V100 GPU machine are the following: DANN: 22.75s, WD$_{\beta=0}$: 59.25s, MARSc: 72.87s, MARSg: 2769.06s, IW-WD: 74.17s, OSTAR+IM: 89.72s. We observe that (i) computing Wasserstein distance (WD$_{\beta=0}$) is slower than computing $\mathcal{H}$-divergence (DANN), (ii) runtimes for domain-invariant GeTarS baselines are slightly higher than for WD$_{\beta=0}$ as proportions are additionally estimated, (iii) domain-invariant GeTarS baselines have similar runtimes apart from MARSg which uses GMM, (iv) our model, OSTAR+IM, has slightly higher runtime than GeTarS baselines other than MARSg. We now provide some further analysis on computational cost and memory in OSTAR. We denote K the number of classes, d the dimension of latent representations, n and m the number of source and target samples.

Algorithm B.2 Minimize Eq. (CAL) w.r.t.. (ϕ, f_N)

$\mathcal{D}_S = \{\mathbf{x_S^{(i)}}, y_S^{(i)}\}_{i=1}^n$, $\mathcal{D}_T = \{\mathbf{x_T^{(i)}}\}_{i=1}^m$, g feature extractor, ϕ domain-mapping, v critic, f_N classifier.

Parameters of ϕ, v, f_N: $\theta_\phi, \theta_v, \theta_{f_N}$ and learning rates $\alpha_\phi, \alpha_v, \alpha_{f_N}$.

N_{iter}: batches per epoch, N_b: batch size, N_v: critic iterations

1: **for** $n_{iter} < N_{iter}$ **do**
2:     Sample minibatches $\mathbf{x_S^B}, y_S^B = \{\mathbf{x_S^{(i)}}, y_S^{(i)}\}_{i=1}^{N_b}$, $\mathbf{x_T^B} = \{\mathbf{z_T^{(i)}}\}_{i=1}^{N_b}$ from $\mathcal{D}_S, \mathcal{D}_T$
3:     Compute $\mathbf{z_S^B} = g(\mathbf{x_S^B})$, $\mathbf{z_N^B} = \phi \circ g(\mathbf{x_S^B})$ and $\mathbf{z_T^B} = g(\mathbf{x_T^B})$
4:     Compute class ratios: $\mathbf{w}_Y = \boldsymbol{p}_N^Y / \boldsymbol{p}_S^Y$
5:     **for** $n_v < N_v$ **do**
6:         Sample random points $\mathbf{z^{B\prime}}$ from the lines between $(\mathbf{z_N^B}, \mathbf{z_T^B})$ pairs
7:         Compute gradient penalty $\mathcal{L}_{grad}$ with $\mathbf{z_N^B}, \mathbf{z_T^B}, \mathbf{z^{B\prime}}$ (Gulrajani et al. 2017)
8:         Compute $\mathcal{L}_{wd}^g = \sum_{i=1}^{N_b} \mathbf{w}_{y_S^{(i)}} v(\mathbf{z_N^{(i)}}) - \frac{1}{N_b} \sum_{i=1}^{N_b} v(\mathbf{z_T^{(i)}})$ Eq. (4.6)
9:         $\theta_v \leftarrow \theta_v - \alpha_v \nabla_{\theta_v} \left[\mathcal{L}_{wd}^g - \mathcal{L}_{grad} \right]$
10:     **end for**
11:     Compute $\mathcal{L}_{OT}^g = \sum_{k=1}^{K} \frac{1}{\#\{y_S^{(i)} = k\}_{i \in [\![1,N_b]\!]}} \sum_{y_S^{(i)} = k,\ i \in [\![1,N_b]\!]} \|\phi(\mathbf{z_S^{(i)}}) - \mathbf{z_S^{(i)}}\|_2^2$
12:     $\theta_\phi \leftarrow \theta_\phi - \alpha_\phi \nabla_{\theta_\phi} \left[\mathcal{L}_{wd}^g + \mathcal{L}_{OT}^g \right]$
13:     Compute $\mathcal{L}_c^g(f_N, N) = \frac{1}{N_b} \sum_{i=1}^{N_b} \mathbf{w}_{y_S^{(i)}} \ell_{ce}(f_N \circ \phi \circ g(\mathbf{x_S^{(i)}}), y_S^{(i)})$
14:     $\theta_{f_N} \leftarrow \theta_{f_N} - \alpha_{f_N} \nabla_{\theta_{f_N}} \mathcal{L}_c^g(f_N, N)$
15: **end for**

- Memory: Proportion estimation is based on the method in Lipton et al. 2018 and requires storing the confusion matrix $\hat{C}$ and predictions $\hat{\boldsymbol{p}}_T^Y$ with a memory cost of $O(K^2)$. Encoding is performed by a deep NN g into $\mathbb{R}^d$ and classification by a shallow classifier from $\mathbb{R}^d$ to $\mathbb{R}^K$. Alignment is performed with a ResNet $\phi : \mathbb{R}^d \to \mathbb{R}^d$ in the latent space which has an order magnitude less parameters than g. Globally, memory consumption is mostly defined by the number of parameters in the encoder g.

- Computational cost comes from: (i) proportion estimation, which depends on $K, n + m$ and is solved once every 5 epochs with small runtime; (ii) alignment between source and target representations, which depends on $d, n + m$ and N_v (the number of critic iterations). This step updates less parameters than domain-invariant methods which align with g instead of ϕ; this may lead to speedups for large encoders. The transport cost $\mathcal{L}_{OT}^g$ depends on n, d and adds small additional runtime; (iii) classification with f_N using labelled source samples, which depends on d, K, n. In a second stage, we improve target discriminativity by updating the encoder g with semi-supervised learning; this depends on $d, K, n + m$.

B.7 Experimental setup

Datasets   We consider the following UDA problems:

- `Digits` is a synthetic binary adaptation problem. We consider adaptation between MNIST and USPS datasets. We consider a subsampled version of the original datasets with the following number of samples per domain: 10000-2609 for MNIST→USPS, 5700-20000 for USPS→MNIST. The feature extractor is learned from scratch.

- `VisDA12` is a 12-class adaptation problem between simulated and real images. We consider a subsampled version of the original problem using 9600 samples per domain and use pre-trained ImageNet ResNet-50 features `http://csr.bu.edu/ftp/visda17/clf/`.

- `Office31` is an object categorization problem with 31 classes. We do not sample the original dataset. There are 3 domains: Amazon (A), DSLR (D) and WebCam (W) and we consider all pairwise source-target domains. We use pre-trained ImageNet ResNet-50 features `https://github.com/jindongwang/transferlearning/blob/master/data/dataset.md`.

- `OfficeHome` is another object categorization problem with 65 classes. We do not sample the original dataset. There are 4 domains: Art (A), Product (P), Clipart

(C), Realworld (R) and we consider all pairwise source-target domains. We use pre-trained ImageNet ResNet-50 features https://github.com/jindongwang/transferlearning/blob/master/data/dataset.md.

Imbalance settings   We consider different class-ratios between domains to simulate label-shift and denote with a "s" prefix, the subsampled datasets. For `Digits`, we explicitly provide the class-ratios as Rakotomamonjy et al. 2021 (e.g. for high imbalance, class 2 accounts for the 7% of target samples while class 4 accounts for 22% of target samples). For `Visda12`, `Office31` and `OfficeHome`, subsampled datasets only consider a small percentage of source samples for the first half classes as Combes et al. 2020 (e.g. `Office31` considers 30% of source samples in classes below 15 and uses all source samples from other classes and all target samples).

Table B.6. – Label imbalance settings

Dataset	Configuration	p_S^Y	p_T^Y
Digits	balanced	$\{\frac{1}{10}, \cdots, \frac{1}{10}\}$	$\{\frac{1}{10}, \cdots, \frac{1}{10}\}$
	subsampled mild	$\{\frac{1}{10}, \cdots, \frac{1}{10}\}$	$\{0, 1, 2, 3, 6\} = 0.06, \{4, 5\} = 0.2, \{7, 8, 9\} = 0.1$
	subsampled high	$\{\frac{1}{10}, \cdots, \frac{1}{10}\}$	$\{0, 1, 2, 3, 6, 7, 8, 9\} = 0.07, \{4, 5\} = 0.22$
VisDA12	original	$\{0 - 11\} : 100\%$	$\{0 - 11\} : 100\%$
	subsampled	$\{0 - 5\} : 30\% \, \{5 - 11\} : 100\%$	$\{0 - 11\} : 100\%$
Office31	subsampled	$\{0 - 15\} : 30\% \, \{15 - 31\} : 100\%$	$\{0 - 31\} : 100\%$
OfficeHome	subsampled	$\{0 - 32\} : 30\% \, \{33 - 64\} : 100\%$	$\{0 - 64\} : 100\%$

Hyperparameters   Domain-invariant methods weight alignment over classification; we tuned the corresponding hyperparameter for $\text{WD}_{\beta=0}$ in the range $[10^{-4}, 10^{-2}]$ and used the one that achieves the best performance on other models. We also tuned λ_{OT} in problem Eq. (CAL) and fixed it to 10^{-2} on `Digits` and 10^{-5} on `VisDA12`, `Office31` and `OfficeHome`. Batch size is $N_b = 200$ and all models are trained using Adam with learning rate tuned in the range $[10^{-4}, 10^{-3}]$. We initialize NN for classifiers and feature extractors with a normal prior with zero mean and gain 0.02 and ϕ with orthogonal initialization with gain 0.02.

Training procedure   We fix N_e the number of epochs to 50 on `Digits`, 150 on `VisDA12` and 100 on `Office31`, `OfficeHome`; `OSTAR` requires smaller N_e to converge. Critic iterations are fixed to $N_v = 5$ which worked best for all baseline models; for `OSTAR` higher values performed better. For all models, we initialize f_S, g for 10 epochs with Eq. (4.3). Then, we perform alignment either through domain-invariance or with a domain-mapping until we reach the total number of epochs. GeTarS models `IW-WD`, `MARSc`, `MARSg`, `OSTAR` perform reweighting with estimates refreshed every $N_u = 2$ epochs in the first 10 alignment epochs, every $N_u = 5$

epochs after. `OSTAR` minimizes Eq. (CAL) + Eq. (SS) for 10 epochs on `Digits`, `Office31`, `OfficeHome` and 5 epochs on `VisDA12`, then minimizes Eq. (CAL) + Eq. (SSg) for remaining epochs.

Architectures   For `Digits`, our feature extractor g is composed of three convolutional layers with respectively 64, 64, 128 filters of size 5×5 interleaved with batch norm, max-pooling and ReLU. Our classifiers (f_S, f_N) are three-layered fully-connected networks with 100 units interleaved with batch norm, ReLU. Our discriminators are three-layered NN with 100 units and ReLU activation. For `VisDA12` and `Office31`, `OfficeHome`, we consider pre-trained 2048 features obtained from a ResNet-50 followed by 2 fully-connected networks with ReLU and 100 units for `VisDA12`, 256 units for `Office31`, `OfficeHome`. Discriminators are 2-layer fully-connected networks with respectively 100/1 units on `VisDA12`, 256/1 units on `Office31`, `OfficeHome` interleaved with ReLU. Classifiers are 2-layer fully-connected networks with $100/K$ units on `VisDA12`, single layer fully-connected network with K units on `Office31`, `OfficeHome`. ϕ is a ResNet with 10 blocks of two fully-connected layers with ReLU and batch-norm.

Implementation of target proportion estimators   `OSTAR` and `IW-WD` use the confusion based estimator in Lipton et al. 2018 and solve a convex optimization problem ((4) in Combes et al. 2020 and CAL w.r.t. $\boldsymbol{p}_N^Y$ for `OSTAR`) which has an unique solution if the soft confusion matrix $\mathbf{C}$ is of full rank. We implement the same optimization problem using the parallel proximal method from Pustelnik et al. 2011 instead of `cvxopt`[1] used in Combes et al. 2020. `MARSc` and `MARSg` (Rakotomamonjy et al. 2021) use linear programming with POT[2] to estimate proportions with optimal assignment between conditional distributions. Target conditionals are obtained with hierarchical clustering or with a Gaussian Mixture Model (GMM) using `sklearn`[3]. `MARSg` has some computational overhead due to the GMM.

---

1. http://cvxopt.org/
2. https://pythonot.github.io/
3. https://scikit-learn.org/

Appendix C

Supplementary Material of Chapter 5

This supplementary material complements the main paper. It is organized as follows:

1. Appendix C.1 points out the limitations of existing flatness-based analysis of Weight Averaging (WA) and shows how our analysis solves these limitations.

2. Appendix C.2 details all the proofs of the propositions and lemmas found in our work. Appendices C.2.1 and C.2.2 derive the bias-variance-covariance-locality decomposition for WA (Proposition 5.1). Appendix C.2.3 establishes the link between bias and correlation shift (Proposition 5.2). Appendix C.2.4 establishes the link between variance and diversity shift (Proposition 5.3). Appendix C.2.5 compares WA with one of its member (Lemma C.2).

3. Appendix C.3 empirically compares WA to Ensembles (ENS).

4. Appendix C.4 presents some additional diversity results on OfficeHome and PACS.

5. Appendix C.5 ablates the importance of the number of training runs.

6. Appendix C.6 describes our experiments on DomainBed and our per-domain results.

7. Appendix C.7 empirically confirms a limitation of WA approaches expected from our theoretical analysis: they do not tackle correlation shift on ColoredMNIST.

8. Appendix C.8 suggests DiWA's potential when some target data is available for training (Kirichenko et al. 2022).

C.1   Limitations of OOD flatness-based analysis

Theorem C.1 (Equation 21 from Cha et al. 2021, simplified version of their Theorem 1). *Consider a set of N covers $\{\Theta_k\}_{k=1}^{N}$ s.t. the parameter space $\Theta \subset \cup_k^N \Theta_k$ where*

$\mathrm{diam}(\Theta) \triangleq \sup_{\theta,\theta' \in \Theta} \|\theta - \theta'\|_2$, $N \triangleq \lceil (\mathrm{diam}(\Theta)/\gamma)^d \rceil$ *and d is the dimension of Θ. Then, $\forall \theta \in \Theta$ with probability at least $1 - \delta$:*

$$
\begin{aligned}
\mathcal{E}_T(\theta) &\leq \frac{1}{2}\mathrm{Div}(p_S, p_T) + \mathcal{E}_S(\theta) \\
&\leq \frac{1}{2}\mathrm{Div}(p_S, p_T) + \mathcal{E}_{\mathcal{D}_S}^\gamma(\theta) + \max_k \sqrt{\frac{(v_k[\ln(n_S/v_k) + 1] + \ln(N/\delta))}{2n_S}},
\end{aligned}
\tag{C.1}
$$

where:

- $\mathcal{E}_T(\theta) \triangleq \mathbb{E}_{(x,y)\sim p_T(X,Y)}[\ell(f_\theta(x); y)]$ *is the expected risk on the target domain,*

- $\mathrm{Div}(p_S, p_T) \triangleq 2\sup_A |p_S(A) - p_T(A)|$ *is a divergence between the source and target marginal distributions p_S and p_T: it measures diversity shift.*

- $\mathcal{E}_S(\theta) \triangleq \mathbb{E}_{(x,y)\sim p_S(X,Y)}[\ell(f_\theta(x); y)]$ *is the expected risk on the source domain,*

- $\mathcal{E}_{\mathcal{D}_S}^\gamma(\theta) \triangleq \max_{\|\Delta\|\leq\gamma} \mathcal{E}_{\mathcal{D}_S}(\theta + \Delta)$ *(where $\mathcal{E}_{\mathcal{D}_S}(\theta + \Delta) \triangleq \mathbb{E}_{(x,y)\in\mathcal{D}_S}[\ell(f_{\theta+\Delta}(x); y)]$) is the robust empirical loss on source training dataset $\mathcal{D}_S$ from S of size n_S,*

- v_k *is a VC dimension of each Θ_k.*

Previous understanding of WA's success in Out-of-distribution (OOD) relied on this upper-bound, where $\mathcal{E}_{\mathcal{D}_S}^\gamma(\theta)$ involves the solution's flatness. This is usually empirically analyzed by the trace of the Hessian (Dinh et al. 2017; Petzka et al. 2021; Yao et al. 2020): indeed, with a second-order Taylor approximation around the local minima θ and h the Hessian's maximum eigenvalue, $\mathcal{E}_{\mathcal{D}_S}^\gamma(\theta) \approx \mathcal{E}_{\mathcal{D}_S}(\theta) + h \times \gamma^2$.

In the following subsections, we show that this inequality does not fully explain the exceptional performance of WA on DomainBed (Gulrajani et al. 2021b). Moreover, we illustrate that our bias-variance-covariance-locality addresses these limitations.

C.1.1 Flatness does not act on distribution shifts

The flatness-based analysis is not specific to OOD. Indeed, the upper-bound in Eq. (C.1) sums up two noninteracting terms: a domain divergence $\mathrm{Div}(p_S, p_T)$ that grows in OOD and $\mathcal{E}_{\mathcal{D}_S}^\gamma(\theta)$ that measures the Independent and Identically Distributed (IID) flatness. The flatness term can indeed be reduced empirically with WA: yet, it does not tackle the domain gap. In fact, Eq. (C.1) states that additional flatness reduces the upper bound of the error similarly no matter the strength of the distribution shift, thus as well OOD than IID. In contrast, our analysis shows that variance (which grows with diversity shift, see Section 5.2.3)

(a) Hessian trace (↓) in train.

(b) Hessian trace (↓) in test OOD.
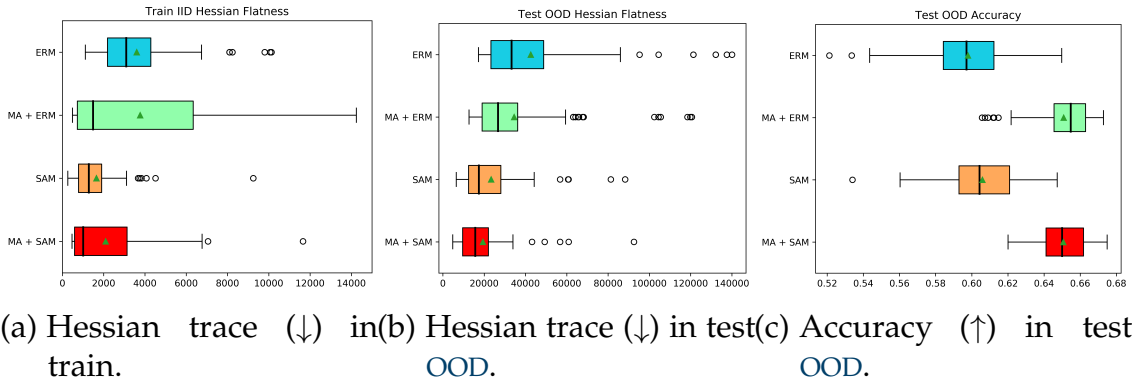
(c) Accuracy (↑) in test OOD.

Figure C.1. – MA (Arpit et al. 2021) (a WA strategy) and SAM (Foret et al. 2021) similarly improve flatness. When combined, they further improve flatness. Yet, MA outperforms SAM and beats MA + SAM in OOD accuracy on domain "Art" from OfficeHome.

is tackled for large M: our error is controlled even under large diversity shift. This is consistent with our experiments in Table 5.1. Our analysis also explains why WA cannot tackle correlation shift (where bias dominates, see Appendix C.7), a limitation Cha et al. 2021 does not illustrate.

C.1.2   SAM leads to flatter minimas but worse OOD performance

The flatness-based analysis does not explain why WA outperforms other flatness-based methods in OOD. We consider Sharpness-Aware Minimizer (SAM) Foret et al. 2021, another popular method to find flat minima based on minimax optimization: it minimizes the maximum loss around a neighborhood of the current weights θ. In Figure C.1, we compare the flatness (*i.e.* the Hessian trace computed with the package in Yao et al. 2020) and accuracy of Expected Risk Minimization (ERM), MA (Arpit et al. 2021) (a WA strategy) and SAM (Foret et al. 2021) when trained on the "Clipart", "Product" and "Photo" domains from OfficeHome Venkateswara et al. 2017b: they are tested OOD on the fourth domain "Art". Analyzing the second and the third rows of Figure C.1a and Figure C.1b, we observe that SAM indeed finds flat minimas (at least comparable to MA), both in training (IID) and test (OOD). However, this is not reflected in the OOD accuracies in Figure C.1c, where MA outperforms SAM. As reported in Table C.1, similar experiments across more datasets lead to the same conclusions in Cha et al. 2021. In conclusion, flatness is not sufficient to explain why WA works so well in OOD, because SAM has similar flatness but worse OOD results. In contrast, we highlight in this paper that WA succeeds in OOD by reducing the impact of variance thanks to its similarity with prediction ensembling Lakshminarayanan et al. 2017 (see Lemma 5.1), a privileged link that SAM does not benefit from.

Table C.1. – **Accuracy (↑) on DomainBed for SWAD**, taken from Table 4 in Cha et al. 2021

	PACS	VLCS	OfficeHome	TerraInc	DomainNet	Avg. (Δ)
ERM	85.5 ± 0.2	77.5 ± 0.4	66.5 ± 0.3	46.1 ± 1.8	40.9 ± 0.1	63.3
SWAD + ERM	$\mathbf{88.1} \pm 0.1$	79.1 ± 0.1	$\mathbf{70.6} \pm 0.2$	$\mathbf{50.0} \pm 0.3$	$\mathbf{46.5} \pm 0.1$	$\mathbf{66.9}(+3.6)$
SAM	85.8 ± 0.2	$\mathbf{79.4} \pm 0.1$	69.6 ± 0.1	43.3 ± 0.7	44.3 ± 0.0	64.5
SWAD + SAM	87.1 ± 0.2	78.5 ± 0.2	69.9 ± 0.1	45.3 ± 0.9	$\mathbf{46.5} \pm 0.1$	$65.5(+1.0)$

Table C.2. – **Accuracy (↑) impact of including SAM** (Foret et al. 2021) on domain "Art" from OfficeHome.

Algorithm	Weight selection	ERM	SAM
No DiWA	N/A	62.9 ± 1.3	63.5 ± 0.5
DiWA	Restricted: $M \leq 20$	66.7 ± 0.1	65.4 ± 0.1
DiWA	Uniform: $M = 20$	67.3 ± 0.3	66.7 ± 0.2
DiWA†	Uniform: $M = 60$	$\mathbf{67.7}$	$\mathbf{67.4}$

C.1.3   WA and SAM are not complementary in OOD when variance dominates

We investigate a similar inconsistency when combining these two flatness-based methods. As argued in Kaddour et al. 2022, we confirm in Figures C.1a and C.1b that MA + SAM leads to flatter minimas than MA alone (*i.e.* with ERM) or SAM alone. Yet, MA does not benefit from SAM in Figure C.1c. Cha et al. 2021 showed an even stronger result in Table C.1: SWAD + ERM performs better than SWAD + SAM. We recover similar findings in Table C.2: DiWA performs worse when SAM is applied in each training run.

This behavior is not explained by Theorem C.1, which states that more flatness should improve OOD generalization. Yet it is explained by our diversity-based analysis. Indeed, we observe in Figure C.2 that the diversity across two checkpoints along a SAM trajectory is much lower than along a standard ERM trajectory (with SGD). We speculate that this is related to the recent empirical observation made in Ramesh et al. 2022: "the rank of the CLIP representation space is drastically reduced when training CLIP with SAM". Under diversity shift, variance dominates (see Eq. (5.3)): in this setup, the gain in accuracy of models trained with SAM cannot compensate the decrease in diversity. This explains why WA and SAM are not complementary under diversity shift where variance is large.
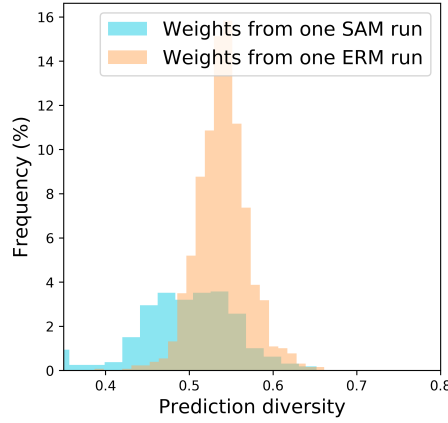
Figure C.2. – Prediction diversity in ratio-error (Aksela 2003) (↑) on domain "Art" from OfficeHome. Checkpoints along a SAM run are less diverse than along an ERM run.

C.2 Proof

C.2.1 WA loss derivation

Lemma (5.1). Given $\{\theta_m\}_{m=1}^{M}$ with learning procedures $L_S^M \triangleq \{l_S^{(m)}\}_{m=1}^{M}$. Denoting $\Delta_{L_S^M} = \max_{m=1}^{M}\|\theta_m - \theta_{\text{WA}}\|_2$, $\forall (x, y) \in \mathcal{X} \times \mathcal{Y}$:

$$f_{\text{WA}}(x) = f_{\text{ENS}}(x) + O(\Delta_{L_S^M}^2) \text{ and } \ell(f_{\text{WA}}(x), y) = \ell(f_{\text{ENS}}(x), y) + O(\Delta_{L_S^M}^2).$$

Proof. This proof has two components:

- to establish the functional approximation, as Izmailov et al. 2018, it performs Taylor expansion of the models' predictions at the first order.

- to establish the loss approximation, as Wortsman et al. 2022a, it performs Taylor expansion of the loss at the first order.

Functional approximation With a Taylor expansion at the first order of the models' predictions w.r.t. parameters θ:

$$f_{\theta_m} = f_{\text{WA}} + \nabla f_{\text{WA}}^{\mathsf{T}}\Delta_m + O\big(\|\Delta_m\|_2^2\big)$$

$$f_{\text{ENS}} - f_{\text{WA}} = \frac{1}{M}\sum_{m=1}^{M}\big(\nabla f_{\text{WA}}^{\mathsf{T}}\Delta_m + O\big(\|\Delta_m\|_2^2\big)\big)$$

Therefore, because $\sum_{m=1}^{M} \Delta_m = 0$,

$$f_{\text{ENS}} - f_{\text{WA}} = O(\Delta^2) \text{ where } \Delta = \max_{m=1}^{M} \|\Delta_m\|_2. \tag{C.2}$$

Loss approximation   With a Taylor expansion at the zeroth order of the loss w.r.t. its first input and injecting Eq. (C.2):

$$\ell(f_{\text{ENS}}(x); y) = \ell(f_{\text{WA}}(x); y) + O(\|f_{\text{ENS}}(x) - f_{\text{WA}}(x)\|_2)$$
$$\ell(f_{\text{ENS}}(x); y) = \ell(f_{\text{WA}}(x); y) + O(\Delta^2).$$

$\square$

C.2.2   Bias-variance-covariance-locality decomposition

Remark. Our result in Proposition 5.1 is simplified by leveraging the fact that the learning procedures $L_S^M = \{l_S^{(m)}\}_{m=1}^{M}$ are Identically Distributed (ID). This assumption naturally holds for DiWA which selects weights from different runs with i.ID hyperparameters. It may be less obvious why it applies to MA (Arpit et al. 2021) and SWAD (Cha et al. 2021). It is even false if the weights $\{\theta(l_S^{(m)})\}_{m=1}^{M}$ are defined as being taken sequentially along a training trajectory, *i.e.* when $0 \leq i < j \leq M$ implies that $l_S^{(i)}$ has fewer training steps than $l_S^{(j)}$. We propose an alternative indexing strategy to respect the ID assumption. Given M weights selected by the weight selection procedure, we draw without replacement the M weights, *i.e.* $\theta(l_S^{(i)})$ refers to the ith sampled weights. With this procedure, all weights are ID as they are uniformly sampled. Critically, their WA are unchanged for the two definitions.

Proposition (5.1). Denoting $\bar{f}_S(x) = \mathbb{E}_{l_S}[f(x, \theta(l_S))]$, under ID learning procedures $L_S^M \triangleq \{l_S^{(m)}\}_{m=1}^{M}$, the expected generalization error on domain T of $\theta_{\text{WA}}(L_S^M) \triangleq \frac{1}{M}\sum_{m=1}^{M}\theta_m$ over the joint distribution of L_S^M is:

$$\mathbb{E}_{L_S^M}\mathcal{E}_T(\theta_{\text{WA}}(L_S^M)) = \mathbb{E}_{(x,y)\sim p_T}\left[\text{bias}^2(x, y) + \frac{1}{M}\text{var}(x) + \frac{M-1}{M}\text{cov}(x)\right] + O(\bar{\Delta}^2),$$

$$\text{where bias}(x, y) = y - \bar{f}_S(x),$$

$$\text{and var}(x) = \mathbb{E}_{l_S}\left[\left(f(x, \theta(l_S)) - \bar{f}_S(x)\right)^2\right],$$

$$\text{and cov}(x) = \mathbb{E}_{l_S, l'_S}\left[\left(f(x, \theta(l_S)) - \bar{f}_S(x)\right)\left(f(x, \theta(l'_S)) - \bar{f}_S(x)\right)\right],$$

$$\text{and } \bar{\Delta}^2 = \mathbb{E}_{L_S^M}\Delta_{L_S^M}^2 \text{ with } \Delta_{L_S^M} = \max_{m=1}^{M}\|\theta_m - \theta_{\text{WA}}\|_2.$$

$$\tag{BVCL}$$

cov is the prediction covariance between two member models whose weights are averaged. The locality term $\bar{\Delta}^2$ is the expected squared maximum distance between weights and their average.

Proof. This proof has two components:

- it follows the bias-variance-covariance decomposition from Ueda et al. 1996; Brown et al. 2005 for functional ensembling. It is tailored to WA by assuming that learning procedures are ID.

- it injects the obtained equation into Lemma 5.1 to obtain the Proposition 5.1 for WA.

BVC for ensembling with ID learning procedures   With $\bar{f}_S(x) = \mathbb{E}_{l_S}[f(x, \theta(l_S))]$, we recall the bias-variance decomposition Kohavi et al. 1996 (Eq. (BV)):

$$\mathbb{E}_{l_S}\mathcal{E}_T(\theta(l_S)) = \mathbb{E}_{(x,y)\sim p_T}\left[\text{bias}(x,y)^2 + \text{var}(x)\right],$$
$$\text{where bias}(x,y) = \text{Bias}\{f|(x,y)\} = y - \bar{f}_S(x),$$
$$\text{and var}(x) = \text{Var}\{f|x\} = \mathbb{E}_{l_S}\left[\left(f(x, \theta(l_S)) - \bar{f}_S(x)\right)^2\right].$$

Using $f_{\text{ENS}} \triangleq f_{\text{ENS}}(\cdot, \{\theta(l_S^{(m)})\}_{m=1}^M) \triangleq \frac{1}{M}\sum_{m=1}^M f(\cdot, \theta(l_S^{(m)}))$ in this decomposition yields,

$$\mathbb{E}_{L_S^M}\mathcal{E}_T(\{\theta(l_S^{(m)})\}_{m=1}^M) = \mathbb{E}_{x\sim p_T}\left[\text{Bias}\{f_{\text{ENS}} \mid (x,y)\}^2 + \text{Var}\{f_{\text{ENS}} \mid x\}\right]. \qquad \text{(C.3)}$$

As f_{ENS} depends on L_S^M, we extend the bias into:

$$\text{Bias}\{f_{\text{ENS}} \mid (x,y)\} = y - \mathbb{E}_{L_S^M}\left[\frac{1}{M}\sum_{m=1}^M f(x, \theta(l_S^{(m)}))\right] = y - \frac{1}{M}\sum_{m=1}^M \mathbb{E}_{l_S^{(m)}}\left[f(x, \theta(l_S^{(m)}))\right]$$

Under ID $L_S^M \triangleq \{l_S^{(m)}\}_{m=1}^M$,

$$\frac{1}{M}\sum_{m=1}^M \mathbb{E}_{l_S^{(m)}}\left[y - f(x, \theta(l_S^{(m)}))\right] = \mathbb{E}_{l_S}[y - f(x, \theta(l_S))] = \text{Bias}\{f|(x,y)\}.$$

Thus the bias of ENS is the same as for a single member of the WA.

Regarding the variance:

$$\text{Var}\{f_{\text{ENS}} \mid x\} = \mathbb{E}_{L_S^M}\left[\left(\frac{1}{M}\sum_{m=1}^{M} f(x, \theta(l_S^{(m)})) - \mathbb{E}_{L_S^M}\left[\frac{1}{M}\sum_{m=1}^{M} f(x, \theta(l_S^{(m)}))\right]\right)^2\right].$$

Under ID $L_S^M \triangleq \{l_S^{(m)}\}_{m=1}^{M}$,

$$\text{Var}\{f_{\text{ENS}} \mid x\} = \frac{1}{M^2}\sum_{m=1}^{M} \mathbb{E}_{l_S}\left[(f(x, \theta(l_S)) - \mathbb{E}_{l_S}[f(x, \theta(l_S))])^2\right] +$$

$$\frac{1}{M^2}\sum_{m}\sum_{m'\neq m} \mathbb{E}_{l_S, l_S'}\left[(f(x, \theta(l_S)) - \mathbb{E}_{l_S}[f(x, \theta(l_S))])(f(x, \theta(l_S')) - \mathbb{E}_{l_S'}[f(x, \theta(l_S'))])\right]$$

$$= \frac{1}{M}\mathbb{E}_{l_S}\left[(f(x, \theta(l_S)) - \mathbb{E}_{l_S}[f(x, \theta(l_S))])^2\right] +$$

$$\frac{M-1}{M}\mathbb{E}_{l_S, l_S'}\left[(f(x, \theta(l_S)) - \mathbb{E}_{l_S}[f(x, \theta(l_S))])(f(x, \theta(l_S')) - \mathbb{E}_{l_S'}[f(x, \theta(l_S'))])\right]$$

$$= \frac{1}{M}\text{var}(x) + \left(1 - \frac{1}{M}\right)\text{cov}(x).$$

The variance is split into the variance of a single member (divided by M) and a covariance term.

Combination with Lemma 5.1   We recall that per Lemma 5.1,

$$\ell(f_{\text{WA}}(x), y) = \ell(f_{\text{ENS}}(x), y) + O(\Delta_{L_S^M}^2).$$

Then we have:

$$\mathcal{E}_T(\theta_{\text{WA}}(L_S^M)) = \mathbb{E}_{(x,y)\sim p_T}[\ell(f_{\text{WA}}(x), y)]$$

$$= \mathbb{E}_{(x,y)\sim p_T}[\ell(f_{\text{ENS}}(x), y)] + O(\Delta_{L_S^M}^2) = \mathcal{E}_T(\{\theta(l_S^{(m)})\}_{m=1}^{M}) + O(\Delta_{L_S^M}^2),$$

$$\mathbb{E}_{L_S^M}\mathcal{E}_T(\theta_{\text{WA}}(L_S^M)) = \mathbb{E}_{L_S^M}\mathcal{E}_T(\{\theta(l_S^{(m)})\}_{m=1}^{M}) + O(\mathbb{E}_{L_S^M}[\Delta_{L_S^M}^2]).$$

We eventually obtain the result:

$$\mathbb{E}_{L_S^M}\mathcal{E}_T(\theta_{\text{WA}}(L_S^M)) = \mathbb{E}_{(x,y)\sim p_T}\left[\text{bias}(x, y)^2 + \frac{1}{M}\text{var}(x) + \frac{M-1}{M}\text{cov}(x)\right] + O(\bar{\Delta}^2).$$

$$\square$$

C.2.3   Bias, correlation shift and support mismatch

We first present in Appendix C.2.3.1 a decomposition of the OOD bias without any assumptions. We then justify in Appendix C.2.3.2 the simplifying Assumption 9 from Section 5.2.3.

C.2.3.1   OOD bias

Proposition C.2 (OOD bias). *Denoting* $\bar{f}_S(x) = \mathbb{E}_{l_s}[f(x, \theta(l_S))]$, *the bias is:*

$$\mathbb{E}_{(x,y)\sim p_T}[\text{bias}^2(x,y)] = \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T(x) - f_S(x))^2 p_T(x) dx \qquad \text{(Correlation shift)}$$

$$+ \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_S(x) - \bar{f}_S(x))^2 p_T(x) dx \qquad \text{(Weighted IID bias)}$$

$$+ \int_{\mathcal{X}_T \cap \mathcal{X}_S} 2(f_T(x) - f_S(x))(f_S(x) - \bar{f}_S(x)) p_T(x) dx \qquad \text{(Interaction IID bias and corr. shift)}$$

$$+ \int_{\mathcal{X}_T \setminus \mathcal{X}_S} (f_T(x) - \bar{f}_S(x))^2 p_T(x) dx. \qquad \text{(Support mismatch)}$$

Proof. This proof is original and based on splitting the OOD bias in and out of $\mathcal{X}_S$:

$$\mathbb{E}_{(x,y)\sim p_T}[\text{bias}^2(x,y)] = \mathbb{E}_{(x,y)\sim p_T}(y - \bar{f}_S(x))^2$$

$$= \int_{\mathcal{X}_T} (f_T(x) - \bar{f}_S(x))^2 p_T(x) dx$$

$$= \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T(x) - \bar{f}_S(x))^2 p_T(x) dx + \int_{\mathcal{X}_T \setminus \mathcal{X}_S} (f_T(x) - \bar{f}_S(x))^2 p_T(x) dx.$$

To decompose the first term, we write $\forall x \in \mathcal{X}_S$, $-\bar{f}_S(x) = -f_S(x) + (f_S(x) - \bar{f}_S(x))$.

$$\int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T(x) - \bar{f}_S(x))^2 p_T(x) dx = \int_{\mathcal{X}_T \cap \mathcal{X}_S} ((f_T(x) - f_S(x)) + (f_S(x) - \bar{f}_S(x)))^2 p_T(x) dx$$

$$= \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T(x) - f_S(x))^2 p_T(x) dx + \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_S(x) - \bar{f}_S(x))^2 p_T(x) dx$$

$$+ \int_{\mathcal{X}_T \cap \mathcal{X}_S} 2(f_T(x) - f_S(x))(f_S(x) - \bar{f}_S(x)) p_T(x) dx.$$

$\square$

The four terms can be qualitatively analyzed:

- The first term measures differences between train and test labelling function. By rewriting $\forall x \in \mathcal{X}_T \cap \mathcal{X}_S$, $f_T(x) \triangleq \mathbb{E}_{p_T}[Y|X = x]$ and $f_S(x) \triangleq \mathbb{E}_{p_S}[Y|X = x]$, this

term measures whether conditional distributions differ. This recovers a similar expression to the correlation shift formula from Ye et al. 2022.

- The second term is exactly the IID bias, but weighted by the marginal distribution $p_T(X)$.

- The third term $\int_{\mathcal{X}_T \cap \mathcal{X}_S} 2(f_T(x) - f_S(x))\big(f_S(x) - \bar{f}_S(x)\big)p_T(x)dx$ measures to what extent the IID bias compensates the correlation shift. It can be negative if (by chance) the IID bias goes in opposite direction to the correlation shift.

- The last term measures support mismatch between test and train marginal distributions. It lead to the "No free lunch for learning representations for DG" in Ruan et al. 2022. The error is irreducible because "outside of the source domain, the label distribution is unconstrained": "for any domain which gives some probability mass on an example that has not been seen during training, then all [...] labels for that example" are possible.

C.2.3.2   Discussion of the small IID bias Assumption 9

Assumption 9 states that $\exists \epsilon > 0$ small s.t. $\forall x \in \mathcal{X}_S, |f_S(x) - \bar{f}_S(x)| \leq \epsilon$ where $\bar{f}_S(x) = \mathbb{E}_{l_S}[f(x, \theta(l_S))]$. $\bar{f}_S$ is the expectation over the possible learning procedures $l_S = \{\mathcal{D}_S, c\}$. Thus Assumption 9 involves:

- the network architecture f which should be able to fit a given dataset $\mathcal{D}_S$. This is realistic when the network is sufficiently parameterized, *i.e.* when the number of weights $|\theta|$ is large.

- the expected datasets $\mathcal{D}_S$ which should be representative enough of the underlying domain S; in particular the dataset size n_S should be large.

- the sampled configurations c which should be well chosen: the network should be trained for enough steps, with an adequate learning rate ...

For DiWA, this is realistic as it selects the weights with the highest training validation accuracy from each run. For SWAD (Cha et al. 2021), this is also realistic thanks to their overfit-aware weight selection strategy. In contrast, this assumption may not perfectlty hold for MA (Arpit et al. 2021), which averages weights starting from batch 100 until the end of training: indeed, 100 batches are not enough to fit the training dataset.

C.2.3.3   OOD bias when small IID bias

We now develop our equality under Assumption 9.

Proposition (5.2. OOD bias when small IID bias). With a bounded difference between the labeling functions $f_T - f_S$ on $\mathcal{X}_T \cap \mathcal{X}_S$, under Assumption 9, the bias on domain T is:

$$\mathbb{E}_{(x,y)\sim p_T}[\text{bias}^2(x,y)] = \text{Correlation shift} + \text{Support mismatch} + O(\epsilon),$$

$$\text{where Correlation shift} = \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T(x) - f_S(x))^2 p_T(x)dx, \tag{5.2}$$

$$\text{and Support mismatch} = \int_{\mathcal{X}_T \setminus \mathcal{X}_S} \left(f_T(x) - \bar{f}_S(x)\right)^2 p_T(x)dx.$$

Proof. We simplify the second and third terms from Proposition C.2 under Assumption 9.

The second term is $\int_{\mathcal{X}_T \cap \mathcal{X}_S} \left(f_S(x) - \bar{f}_S(x)\right)^2 p_T(x)dx$. Under Assumption 9, $|f_S(x) - \bar{f}_S(x)| \leq \epsilon$. Thus the second term is $O(\epsilon^2)$.

The third term is $\int_{\mathcal{X}_T \cap \mathcal{X}_S} 2(f_T(x) - f_S(x))\left(f_S(x) - \bar{f}_S(x)\right) p_T(x)dx$. As $f_T - f_S$ is bounded on $\mathcal{X}_S \cap \mathcal{X}_T$, $\exists K \geq 0$ such that $\forall x \in \mathcal{X}_S$,

$$|(f_T(x) - f_S(x))\left(f_S(x) - \bar{f}_S(x)\right) p_T(x)| \leq K|f_S(x) - \bar{f}_S(x)|p_T(x) = O(\epsilon)p_T(x).$$

Thus the third term is $O(\epsilon)$.

Finally, note that we cannot say anything about $\bar{f}_S(x)$ when $x \in \mathcal{X}_T \setminus \mathcal{X}_S$.   □

To prove the previous equality, we needed a bounded difference between labeling functions $f_T - f_S$ on $\mathcal{X}_T \cap \mathcal{X}_S$. We relax this bounded assumption to obtain an inequality in the following Proposition C.3.

Proposition C.3 (OOD bias when small IID bias without bounded difference between labeling functions). *Under Assumption 9,*

$$\mathbb{E}_{(x,y)\sim p_T}[\text{bias}^2(x,y)] \leq 2 \times \text{Correlation shift} + \text{Support mismatch} + O(\epsilon^2) \tag{C.4}$$

Proof. We follow the same proof as in Proposition C.2, except that we now use: $(a+b)^2 \leq 2(a^2 + b^2)$. Then,

$$\int_{\mathcal{X}_T \cap \mathcal{X}_S} \left(f_T(x) - \bar{f}_S(x)\right)^2 p_T(x)dx = \int_{\mathcal{X}_T \cap \mathcal{X}_S} \left((f_T(x) - f_S(x)) + \left(f_S(x) - \bar{f}_S(x)\right)\right)^2 p_T(x)dx$$

$$\leq 2 \times \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T(x) - f_S(x))^2 + \left(f_S(x) - \bar{f}_S(x)\right)^2 p_T(x)dx$$

$$\leq 2 \times \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T(x) - f_S(x))^2 p_T(x)dx + 2 \times \int_{\mathcal{X}_T \cap \mathcal{X}_S} \epsilon^2 p_T(x)dx$$

$$\leq 2 \times \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T(x) - f_S(x))^2 p_T(x)dx + O(\epsilon^2)$$

☐

C.2.4   Variance and diversity shift

We prove the link between variance and diversity shift. Our proof builds upon the similarity between Neural Network (NN)s and Gaussian Process (GP)s in the kernel regime, detailed in Appendix C.2.4.1. We discuss our simplifying Assumption 11 in Appendix C.2.4.2. We present our final proof in Appendix C.2.4.3. We discuss the relation between variance and initialization in Appendix C.2.4.4.

C.2.4.1   Neural Networks as Gaussian Processes

We fix $\mathcal{D}_S, \mathcal{D}_T$ and denote $X_{\mathcal{D}_S} = \{x_S\}_{(x_S,y_S)\in\mathcal{D}_S}$, $X_{\mathcal{D}_T} = \{x_T\}_{(x_T,y_T)\in\mathcal{D}_T}$ their respective input supports. We fix the initialization of the network. l_S encapsulates all other sources of randomness.

Lemma C.1 (Inspired from Rasmussen 2003). *Given a* NN *$f(\cdot, \theta(l_S))$ under Assumption 10, we denote K its neural tangent kernel and $K(X_{\mathcal{D}_S}, X_{\mathcal{D}_S}) \triangleq (K(x_S, x'_S))_{x_S,x'_S\in X^2_{\mathcal{D}_S}} \in \mathbb{R}^{n_S\times n_S}$. Given $x \in \mathcal{X}$, we denote $K(x, X_{\mathcal{D}_S}) \triangleq [K(x, x_S)]_{x_S\in X_{\mathcal{D}_S}} \in \mathbb{R}^{n_S}$. Then:*

$$\mathrm{var}(x) = K(x, x) - K(x, X_{\mathcal{D}_S})K(X_{\mathcal{D}_S}, X_{\mathcal{D}_S})^{-1}K(x, X_{\mathcal{D}_S})^\intercal. \tag{C.5}$$

Proof. Under Assumption 10, NNs are equivalent to GPs. $\mathrm{var}(x)$ is the formula of the variance of the GP posterior given by Eq. (2.26) in Rasmussen 2003, when conditioned on $\mathcal{D}_S$. This formula thus also applies to the variance $f(\cdot, \theta(l_S))$ when l_S varies (at fixed $\mathcal{D}_S$ and initialization). ☐
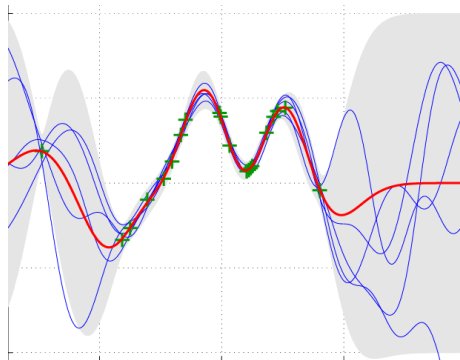

Figure C.3. – **Mean and variance of a Gaussian process's prediction**. Image from Pérez-Cruz et al. 2013. Intuitively, variance grows when samples are distant from training samples.

C.2.4.2 Discussion of the same norm and low similarity Assumption 11 on source dataset

Lemma C.1 shows that the variance only depends on the input distributions $p(X)$ without involving the label distributions $p(Y|X)$. This formula highlights that the variance is related to shifts in input similarities (measured by K) between $X_{\mathcal{D}_S}$ and $X_{\mathcal{D}_T}$. Yet, a more refined analysis of the variance requires additional assumptions, in particular to obtain a closed-form expression of $K(X_{\mathcal{D}_S}, X_{\mathcal{D}_S})^{-1}$. Assumption 11 is useful because then $K(X_{\mathcal{D}_S}, X_{\mathcal{D}_S})$ is diagonally dominant and can be approximately inverted (see Appendix C.2.4.3).

The first part of Assumption 11 assumes that $\exists \lambda_S$ such that all training inputs $x_S \in X_{\mathcal{D}_S}$ verify $K(x_S, x_S) = \lambda_S$. Note that this equality is standard in some kernel machine algorithms Ah-Pine 2010; Ghojogh et al. 2021; Rennie 2005 and is usually achieved by replacing $K(x, x')$ by $\lambda_S \frac{K(x,x')}{\sqrt{K(x,x)}\sqrt{K(x',x')}}, \forall (x, x') \in (X_{\mathcal{D}_S} \cup X_{\mathcal{D}_T})^2$. In the Neural Tangent Kernel (NTK) literature, this equality is achieved without changing the kernel by normalizing the samples of $X_{\mathcal{D}_S}$ such that they lie on the hypersphere; this input preprocessing was used in Lee et al. 2017. This is theoretically based: for example, the NTK $K(x, x')$ for an architecture with an initial fully connected layer only depends on $\|x\|, \|x'\|, \langle x, x' \rangle$ Yang et al. 2019. Thus in the case where all samples from $X_{\mathcal{D}_S}$ are preprocessed to have the same norm, the value of $K(x_S, x_S)$ does not depend on $x_S \in X_{\mathcal{D}_S}$; we denote λ_S the corresponding value.

The second part of Assumption 11 states that $\exists 0 \leq \epsilon \ll \lambda_S$, s.t. $\forall x_S, x'_S \in X_{\mathcal{D}_S}^2, x_S \neq x'_S \Rightarrow |K(x_S, x'_S)| \leq \epsilon$, *i.e.* that training samples are dissimilar and do not interact. This diagonal structure of the NTK Jacot et al. 2018, with diagonal values larger than non-diagonal ones, is consistent with empirical observations from Seleznova et al. 2022 at initialization. Theoretically, this is reasonable if K is close to the RBF kernel $K_h(x, x') = \exp(-\|x - x'\|_2^2/h)$ where h would be the bandwidth: in this case, Assumption 11 is satisfied when training inputs are distant in pixel space.

We now provide an analysis of the variance where the diagonal assumption is relaxed. Specifically, we provide the sketch for proving an upper-bound of the variance when the NTK has a block-diagonal structure. This is indeed closer to the empirical observations in Seleznova et al. 2022 at the end of training, consistently with the local elasticity property of NNs He et al. 2020. We then consider the dataset $d_{S'} \subset \mathcal{D}_S$ made of one sample per block, to which Assumption 11 applies. As decreasing the size of a training dataset empirically reduces variance Brain et al. 1999, the variance of f trained on $\mathcal{D}_S$ is upper-bounded by the variance of f trained on $d_{S'}$; the latter is given by applying Proposition 5.3 to $d_{S'}$. We believe that the proper formulation of this idea is beyond the scope of this article and best left for future theoretical work.

C.2.4.3 Expression of OOD variance

Proposition (5.3). Given f trained on source dataset $\mathcal{D}_S$ (of size n_S) with NTK K, under Assumptions 10 and 11, the variance on dataset $\mathcal{D}_T$ is:

$$\mathbb{E}_{x_T \in X_{\mathcal{D}_T}}[\mathrm{var}(x_T)] = \frac{n_S}{2\lambda_S}\mathrm{MMD}^2(X_{\mathcal{D}_S}, X_{\mathcal{D}_T}) + \lambda_T - \frac{n_S}{2\lambda_S}\beta_T + O(\epsilon), \qquad (5.3)$$

with MMD the empirical Maximum Mean Discrepancy (MMD) in the RKHS of $K^2(x,y) = (K(x,y))^2$;$\lambda_T \triangleq \mathbb{E}_{x_T \in X_{\mathcal{D}_T}}K(x_T, x_T)$ and $\beta_T \triangleq \mathbb{E}_{(x_T, x'_T) \in X^2_{\mathcal{D}_T}, x_T \neq x'_T}K^2(x_T, x'_T)$ the empirical mean similarities resp. measured between identical (w.r.t. K) and different (w.r.t. K^2) samples averaged over $X_{\mathcal{D}_T}$.

Proof. Our proof is original and is based on the posterior form of GPs in Lemma C.1. Given $\mathcal{D}_S$, we recall Eq. (C.5) that states $\forall x \in \mathcal{X}$:

$$\mathrm{var}(x) = K(x,x) - K(x, X_{\mathcal{D}_S})K(X_{\mathcal{D}_S}, X_{\mathcal{D}_S})^{-1}K(x, X_{\mathcal{D}_S})^{\mathsf{T}}.$$

Denoting $B = K(X_{\mathcal{D}_S}, X_{\mathcal{D}_S})^{-1}$ with symmetric coefficients $b_{i,j} = b_{j,i}$, then

$$\mathrm{var}(x) = K(x,x) - \sum_{\substack{1 \leq i \leq n_S \\ 1 \leq j \leq n_S}} b_{i,j}K(x, x_S^i)K(x, x_S^j). \qquad (C.6)$$

Assumption 11 states that $K(X_{\mathcal{D}_S}, X_{\mathcal{D}_S}) = A + H$ where $A = \lambda_S \mathbb{I}_{n_S}$ and $H = (h_{ij})_{\substack{1 \leq i \leq n_S \\ 1 \leq j \leq n_S}}$ with $h_{i,i} = 0$ and $\max_{i,j}|h_{i,j}| \leq \epsilon$.
We fix $x_T \in X_{\mathcal{D}_T}$ and determine the form of B^{-1} in two cases: $\epsilon = 0$ and $\epsilon \neq 0$.

Case when $\epsilon = 0$ We first derive a simplified result, when $\epsilon = 0$.
Then, $b_{i,i} = \frac{1}{\lambda_S}$ and $b_{i,j} = 0$ s.t.

$$\mathrm{var}(x_T) = K(x_T, x_T) - \sum_{x_S \in X_{\mathcal{D}_S}} \frac{K(x_T, x_S)^2}{\lambda_S} = K(x,x) - \frac{n_S}{\lambda_S}\mathbb{E}_{x_S \in X_{\mathcal{D}_S}}[K^2(x, x_S)]$$

We can then write:

$$\mathbb{E}_{x_T \in X_{\mathcal{D}_T}}[\mathrm{var}(x_T)] = \mathbb{E}_{x_T \in X_{\mathcal{D}_T}}[K(x_T, x_T)] - \frac{n_S}{\lambda_S}\mathbb{E}_{x_T \in X_{\mathcal{D}_T}}[\mathbb{E}_{x_S \in X_{\mathcal{D}_S}}[K^2(x_T, x_S)]]$$

$$\mathbb{E}_{x_T \in X_{\mathcal{D}_T}}[\mathrm{var}(x_T)] = \lambda_T - \frac{n_S}{\lambda_S}\mathbb{E}_{x_S \in X_{\mathcal{D}_S}, x_T \in X_{\mathcal{D}_T}}[K^2(x_T, x_S)].$$

We now relate the second term on the r.h.s. to a MMD distance. As K is a kernel, K^2 is a kernel and its MMD between $X_{\mathcal{D}_S}$ and $X_{\mathcal{D}_T}$ is per Gretton et al. 2012:

$$\text{MMD}^2(X_{\mathcal{D}_S}, X_{\mathcal{D}_T}) = \mathbb{E}_{x_S \neq x'_S \in X^2_{\mathcal{D}_S}}[K^2(x_S, x'_S)] + \mathbb{E}_{x_T \neq x'_T \in X^2_{\mathcal{D}_T}}[K^2(x_T, x'_T)]$$
$$- 2\mathbb{E}_{x_S \in X_{\mathcal{D}_S}, x_T \in X_{\mathcal{D}_T}}[K^2(x_T, x_S)].$$

Finally, because $\epsilon = 0$, $\mathbb{E}_{x_S \neq x'_S \in X^2_{\mathcal{D}_S}} K^2(x_S, x'_S) = 0$ s.t.

$$\mathbb{E}_{x_T \in X_{\mathcal{D}_T}}[\text{var}(x_T)] = \frac{n_S}{2\lambda_S} \text{MMD}^2(X_{\mathcal{D}_S}, X_{\mathcal{D}_T}) + \lambda_T$$
$$- \frac{n_S}{2\lambda_S}\left(\mathbb{E}_{x_T \neq x'_T \in X^2_{\mathcal{D}_T}} K^2(x_T, x'_T) + \mathbb{E}_{x_S \neq x'_S \in X^2_{\mathcal{D}_S}} K^2(x_S, x'_S)\right)$$
$$= \frac{n_S}{2\lambda_S} \text{MMD}^2(X_{\mathcal{D}_S}, X_{\mathcal{D}_T}) + \lambda_T - \frac{n_S}{2\lambda_S}\mathbb{E}_{x_T \neq x'_T \in X^2_{\mathcal{D}_T}} K^2(x_T, x'_T)$$
$$= \frac{n_S}{2\lambda_S} \text{MMD}^2(X_{\mathcal{D}_S}, X_{\mathcal{D}_T}) + \lambda_T - \frac{n_S}{2\lambda_S}\beta_T.$$

We recover the same expression with a $O(\epsilon)$ in the general setting where $\epsilon \neq 0$.

Case when $\epsilon \neq 0$ We denote $I : \begin{cases} \text{GL}_{n_S}(\mathbb{R}) & \to \text{GL}_{n_S}(\mathbb{R}) \\ A & \mapsto A^{-1} \end{cases}$ the inversion function defined on $\text{GL}_{n_S}(\mathbb{R})$, the set of invertible matrices of $\mathcal{M}_{n_S}(\mathbb{R})$.
The function I is differentiable Magnus et al. 2019 in all $A \in \text{GL}_{n_S}(\mathbb{R})$ with its differentiate given by the linear application $dI_A : \begin{cases} \mathcal{M}_{n_S}(\mathbb{R}) & \to \mathcal{M}_{n_S}(\mathbb{R}) \\ H & \mapsto -A^{-1}HA^{-1} \end{cases}$.
Therefore, we can perform a Taylor expansion of I at the first order at A:

$$I(A + H) = I(A) + dI_A(H) + o(\|H\|),$$
$$(A + H)^{-1} = A^{-1} - A^{-1}HA^{-1} + o(\|H\|).$$

where $\|H\| \leq n_S\epsilon = O(\epsilon)$. Thus,

$$(\lambda_S \mathbb{I}_{n_S} + H)^{-1} = (\lambda_S \mathbb{I}_{n_S})^{-1} - (\lambda_S \mathbb{I}_{n_S})^{-1} H (\lambda_S \mathbb{I}_{n_S})^{-1} + O(\epsilon) = \frac{1}{\lambda_S}\mathbb{I}_{n_S} - \frac{1}{\lambda_S^2}H + O(\epsilon),$$

$$\forall i \in [\![1, n_S]\!], b_{ii} = \frac{1}{\lambda_S} - \frac{1}{\lambda_S^2}h_{i,i} + o(\epsilon) = \frac{1}{\lambda_S} + O(\epsilon),$$

$$\forall i \neq j \in [\![1, n_S]\!], b_{ij} = -\frac{1}{\lambda_S^2}h_{i,j} + o(\epsilon) = O(\epsilon).$$

Therefore, when ϵ is small, Eq. (C.6) can be developed into:

$$\text{var}(x_T) = K(x_T, x_T) - \sum_{x_S \in X_{\mathcal{D}_S}} (\frac{1}{\lambda_S} + O(\epsilon)) K(x_T, x_S)^2 + O(\epsilon)$$

$$= K(x_T, x_T) - \frac{n_S}{\lambda_S} \mathbb{E}_{x_S \in X_{\mathcal{D}_S}}[K(x_T, x_S)^2] + O(\epsilon)$$

Following the derivation for the case $\epsilon = 0$, and remarking that under Assumption 11 we have $\mathbb{E}_{x_S \neq x'_S \in X^2_{\mathcal{D}_S}} K^2(x_S, x'_S) = O(\epsilon^2)$, yields:

$$\mathbb{E}_{x_T \in X_{\mathcal{D}_T}}[\text{var}(x_T)] = \frac{n_S}{2\lambda_S} \text{MMD}^2(X_{\mathcal{D}_S}, X_{\mathcal{D}_T}) + \lambda_T - \frac{n_S}{2\lambda_S}\beta_T + O(\epsilon).$$

$\square$

C.2.4.4 Variance and initialization

The MMD depends on the kernel K, *i.e.* only on the initialization of f in the kernel regime per Jacot et al. 2018. Thus, to reduce variance, we could act on the initialization to match $p_S(X)$ and $p_T(X)$ in the RKHS of K^2. This is consistent with Section 5.2.3 that motivated matching the train and test in features. In our paper, we used the standard pretraining from ImageNet Krizhevsky et al. 2012, as commonly done on DomainBed Gulrajani et al. 2021b. The Linear Probing (Kumar et al. 2022) initialization of the classifier was shown in Kumar et al. 2022 to prevent the distortion of the features along the training. This could be improved by pretraining the encoder on a task with fewer domain-specific information, *e.g.* CLIP Radford et al. 2021 image-to-text translation as in Ruan et al. 2022.

C.2.5 WA vs. its members

We validate that WA's expected error is smaller than its members' error under the locality constraint.

Lemma C.2 (WA vs. its members.)**.**

$$\mathbb{E}_{L_S^M}\mathcal{E}_T(\theta_{WA}(L_S^M)) - \mathbb{E}_{l_S}\mathcal{E}_T(\theta(l_S)) = \frac{M-1}{M}\mathbb{E}_{x \sim p_T}[\text{cov}(x) - \text{var}(x)] + O(\bar{\Delta}^2) \leq O(\bar{\Delta}^2).$$

(C.7)

Proof. The proof builds upon Eq. (BVCL):

$$\mathbb{E}_{L_S^M}\mathcal{E}_T(\theta_{WA}) = \mathbb{E}_{(x,y) \sim p_T}\left[\text{bias}(x,y)^2 + \frac{1}{M}\text{var}(x) + \frac{M-1}{M}\text{cov}(x)\right] + O(\bar{\Delta}^2),$$
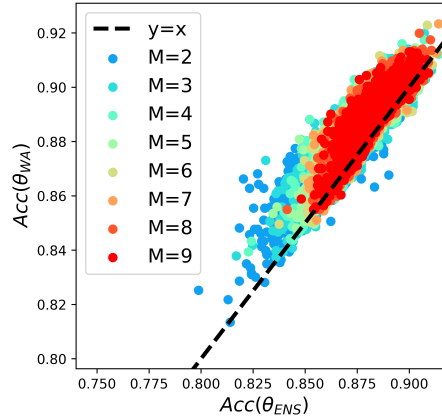
Figure C.4. – f_{WA} performs similarly or better than f_{ENS} on domain "Art" on PACS.

and the expression of the standard bias-variance decomposition in Eq. (BV) from Kohavi et al. 1996,

$$\mathbb{E}_{l_S}\mathcal{E}_T(\theta) = \mathbb{E}_{(x,y)\sim p_T}\Big[\mathrm{bias}(x,y)^2 + \mathrm{var}(x)\Big].$$

The difference between the two provides:

$$\mathbb{E}_{L_S^M}\mathcal{E}_T(\theta_{\mathrm{WA}}) - \mathbb{E}_{l_S}\mathcal{E}_T(\theta) = \frac{M-1}{M}\mathbb{E}_{(x,y)\sim p_T}\Big[\mathrm{cov}(x) - \mathrm{var}(x)\Big] + O(\bar{\Delta}^2).$$

Cauchy Schwartz inequality states $|\mathrm{cov}(Y,Y')| \le \sqrt{\mathrm{var}(Y)\mathrm{var}(Y')}$, thus $\mathrm{cov}(x) \le \mathrm{var}(x)$. Then:

$$\mathbb{E}_{L_S^M}\mathcal{E}_T(\theta_{\mathrm{WA}}) - \mathbb{E}_{l_S}\mathcal{E}_T(\theta) \le O(\bar{\Delta}^2).$$

$\square$

C.3   WA versus functional ensembling

We further compare the following two methods to combine M weights $\{\theta(l_S^{(m)})\}_{m=1}^M$: f_{WA} that averages the weights and f_{ENS} Lakshminarayanan et al. 2017 that averages the predictions. We showed in Lemma 5.1 that $f_{\mathrm{WA}} \approx f_{\mathrm{ENS}}$ when $\max_{m=1}^M \|\theta(l_S^{(m)}) - \theta_{\mathrm{WA}}\|_2$ is small.

In particular, when $\{l_S^{(m)}\}_{m=1}^M$ share the same initialization and the hyperparameters are sampled from mild ranges, we empirically validate our approximation on OfficeHome in Figure 5.1. This is confirmed on PACS dataset in Figure C.4. For both datasets, we even observe that f_{WA} performs slightly but consistently

better than f_{ENS}. The observed improvement is non-trivial; we refer to Equation 1 in Wortsman et al. 2022a for some initial explanations based on the value of OOD Hessian and the confidence of f_{WA}. The complete analysis of this second-order difference is left for future work.

Yet, we do not claim that f_{WA} is systematically better than f_{ENS}. In Table C.3, we show that this is no longer the case when we relax our two constraints, consistently with Figure 5.5. *First*, when the classifiers' initializations vary, ENS improves thanks to this additional diversity; in contrast, DiWA degrades because weights are no longer averageable. *Second*, when the hyperparameters are sampled from extreme ranges (defined in Table C.5), performance drops significantly for DiWA, but much less for ENS. As a side note, the downward trend in this second setup (even for ENS) is due to inadequate hyperparameters that degrade the expected individual performances.

This highlights a limitation of DiWA, which requires weights that satisfy the locality requirement or are at least linearly connectable. In contrast, Deep Ensembles Lakshminarayanan et al. 2017 are computationally expensive (and even impractical for large M), but can leverage additional sources of diversity. An interesting extension of DiWA for future work would be to consider the functional ensembling of several DiWAs trained from different initializations or even with different network architectures Singh et al. 2016. Thus the Ensemble of Averages (EoA) strategy introduced in Arpit et al. 2021 is complementary to DiWA and could be extended into an Ensemble of Diverse Weight Averages.

Table C.3. – **DiWA's vs. ENS's accuracy ($\%, \uparrow$)** on domain "Art" from OfficeHome when varying initialization and hyperparameter ranges. Best on each setting is in **bold**.

Configuration		$M = 20$		$M = 60$	
Shared classifier init	Mild hyperparameter ranges	DiWA	ENS	DiWA	ENS
✓	✓	**67.3** $\pm$ 0.2	66.1 $\pm$ 0.1	**67.7**	66.5
✗	✓	65.0 $\pm$ 0.5	**67.5** $\pm$ 0.3	65.9	**68.5**
✓	✗	56.6 $\pm$ 0.9	**64.3** $\pm$ 0.4	59.5	**64.7**

C.4   Additional diversity analysis

C.4.1   Feature diversity on OfficeHome

In Section 5.4, our diversity-based theoretical findings were empirically validated using the ratio-error Aksela 2003, a common diversity measure notably used in

Rame et al. 2021a; Rame et al. 2021b. In Figure C.5, we recover similar conclusions with another diversity measure: the Centered Kernel Alignment Complement (CKAC) Kornblith et al. 2019, also used in Neyshabur et al. 2020; Wortsman et al. 2022b. CKAC operates in the feature space and measures to what extent the pairwise similarity matrices (computed on domain T) are aligned — where similarity is the dot product between penultimate representations extracted from two different networks.

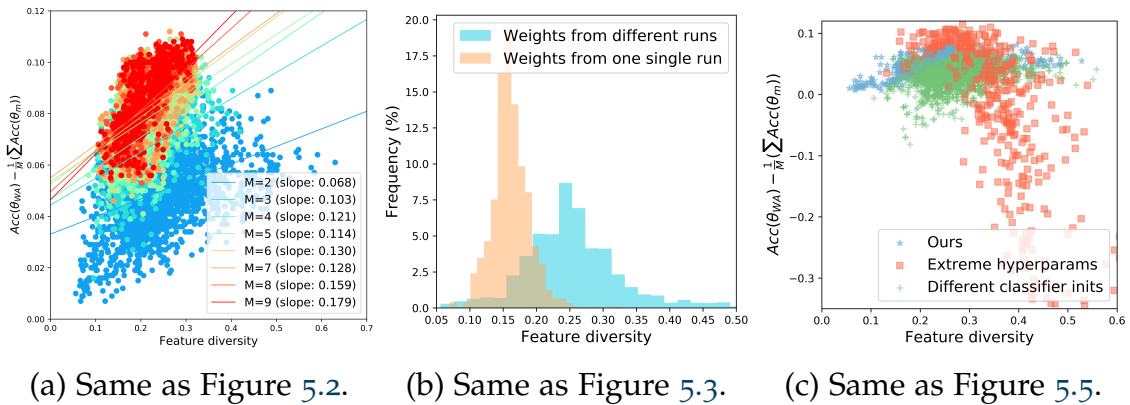(a) Same as Figure 5.2.   (b) Same as Figure 5.3.   (c) Same as Figure 5.5.

Figure C.5. – Same analysis as Section 5.4, where diversity is measured with CKAC (Kornblith et al. 2019) in features rather than with ratio-error (Aksela 2003) in predictions.

C.4.2   Accuracy gain per unit of diversity on OfficeHome

In Figures 5.2 and C.5a, we indicated the slope of the linear regressions relating diversity to accuracy gain at fixed M (between 2 and 9). For example, when $M = 9$ weights are averaged, the accuracy gain increases by 0.297 per unit of additional diversity in prediction Aksela 2003 (see Figure 5.2) and by 0.179 per unit of additional diversity in features Kornblith et al. 2019 (see Figure C.5a). Most importantly, we note that the slope increases with M. To make this more visible, we plot slopes w.r.t. M in Figure C.6. Our observations are consistent with the $(M-1)/M$ factor in front of $\mathrm{cov}(x)$ in Eq. (BVCL). This shows that diversity becomes more important for large M. Yet, large M is computationally impractical in standard functional ensembling, as one forward step is required per model. In contrast, WA has a fixed inference time which allows it to consider larger M. Increasing M from 20 to 60 is the main reason why DiWA† improves DiWA.
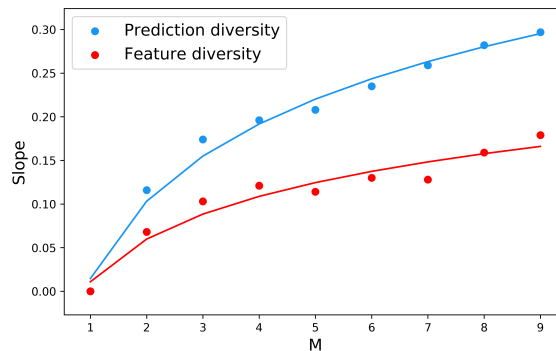
Figure C.6. – The slopes of linear regression — relating diversity to accuracy gain in Figure 5.2 and Figure C.5a — increases with M.

C.4.3   Diversity comparison across a wide range of methods on OfficeHome

Inspired by Gontijo-Lopes et al. 2022, we further analyze in Figure C.7 the diversity between two weights obtained from different (more or less correlated) learning procedures.

- In the upper part, weights are obtained from a single run. They share the same initialization/hyperparameters/data/noise in the optimization procedure and only differ by the number of training steps (which we choose to be a multiple of 50). They are less diverse than the weights in the middle part of Figure C.7, that are sampled from two ERM runs.

- When sampled from different runs, the weights become even more diverse when they have more extreme hyperparameter ranges, they do not share the same classifier initialization or they are trained on different data. The first two are impractical for WA, as it breaks the locality requirement (see Figures 5.5 and C.5c). Luckily, the third setting "data diversity" is more convenient and is another reason for the success of DiWA†; its 60 weights were trained on 3 different data splits. Data diversity has provable benefits Efron 1992, *e.g.* in bagging Breiman 1996.

- Finally, we observe that diversity is increased (notably in features) when two runs have different objectives, for example, Interdomain Mixup (Yan et al. 2020) and Coral Sun et al. 2016. Thus incorporating weights trained with different invariance-based objectives have two benefits that explain the strong results in Table 5.2: (1) they learn invariant features by leveraging the domain information

and (2) they enrich the diversity of solutions by extracting different features. These solutions can bring their own particularity to WA.

In conclusion, our analysis confirms that "model pairs that diverge more in training methodology display categorically different generalization behavior, producing increasingly uncorrelated errors", as stated in Gontijo-Lopes et al. 2022.
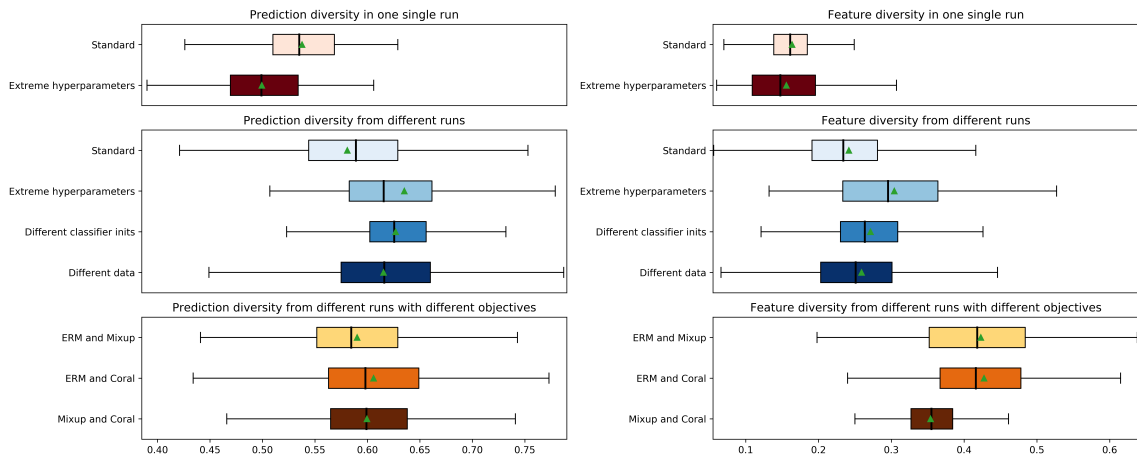
(a) Prediction diversity (Aksela 2003). (b) Feature diversity (Kornblith et al. 2019).

Figure C.7. – **Diversity analysis** across weights, which are per default trained with ERM, with a mild hyperparameter range (see Table C.5), with a shared random classifier initialization, on a given data split. *First*, it confirms Figures 5.3 and C.5b: weights obtained from two different runs are more different than those sampled from a single run (even with extreme hyperparameters). *Second*, this shows that weights from two runs are more diverse when the two runs have different hyperparameters/data/classifier initializations/training objectives. Domain "Art" on OfficeHome.

C.4.4 Trade-off between diversity and averageability on Office-Home

We argue in Section 5.2.3 that our weights should ideally be diverse functionally while being averageable (despite the nonlinearities in the network). We know from Neyshabur et al. 2020 that models fine-tuned from a shared initialization with shared hyperparameters can be connected along a linear path where error remains low; thus, they are averageable as their WA also has a low loss. In Figure 5.5, we confirmed that averaging models from different initializations performs poorly. Regarding the hyperparameters, Figure 5.5 shows that hyperparameters can be

selected slightly different but not too distant. That is why we chose mild hyper-parameter ranges (defined in Table C.5) in our main experiments.

A complete analysis of when the averageability holds when varying the different hyperparameters is a promising lead for future work. Still, Figure C.8 is a preliminary investigation of the impact of different learning rates (between learning procedures of each weight). First, we validate that more distant learning rates lead to more functional diversity in Figure C.8a. Yet, we observe in Figure C.8b that if learning rates are too different, weight averaging no longer approximates functional ensembling because the $O(\Delta^2_{L^M_S})$ term in Lemma 5.1 can be large.
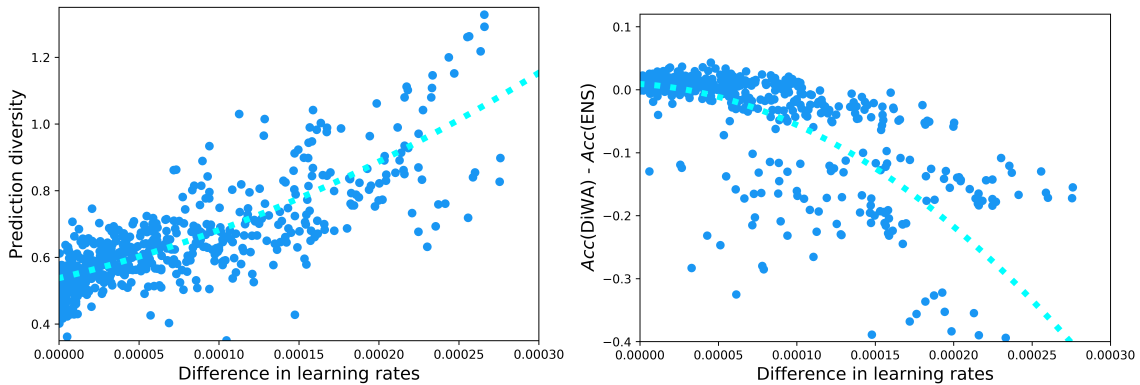
(a) Prediction diversity (↑) (Aksela 2003) between models.

(b) Accuracy (↑) difference between DiWA and ENS.

Figure C.8. – **Trade-off between diversity and averageability for various differences in learning rates**. Considering $M = 2$ weights obtained from two learning procedures with learning rates lr_1 and lr_2 (sampled from the extreme distribution in Table C.5), we plot in Figure C.8a the prediction diversity for these $M = 2$ models vs. $|\mathrm{lr}_1 - \mathrm{lr}_2|$. Then, in Figure C.8b, we plot the accuracy differences $\mathrm{Acc}(\mathrm{DiWA}) - \mathrm{Acc}(\mathrm{ENS})$ vs. $|\mathrm{lr}_1 - \mathrm{lr}_2|$.

C.4.5   On PACS

We perform in Figure C.9 on domain "Art" from PACS the same core diversity-based experiments than on OfficeHome in Section 5.4. We recover the same conclusions.
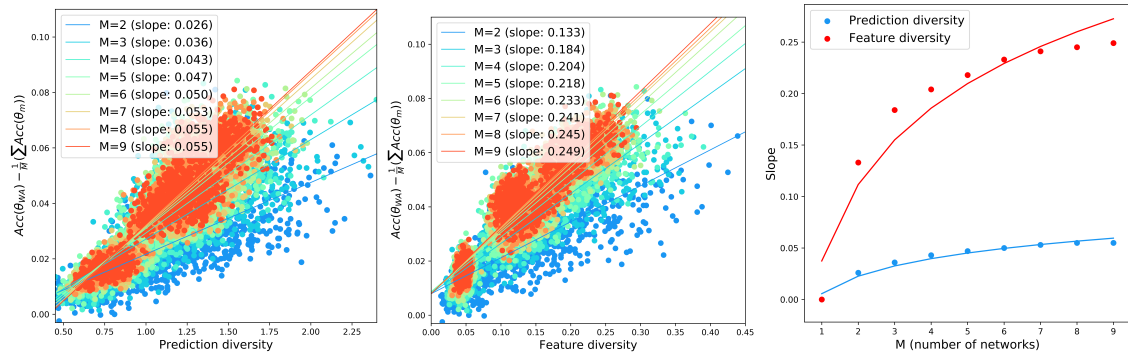
(a) Same as Figure 5.2.   (b) Same as Figure C.5a.   (c) Same as Figure C.6.

Figure C.9. – Same analysis on PACS as previously done on OfficeHome.
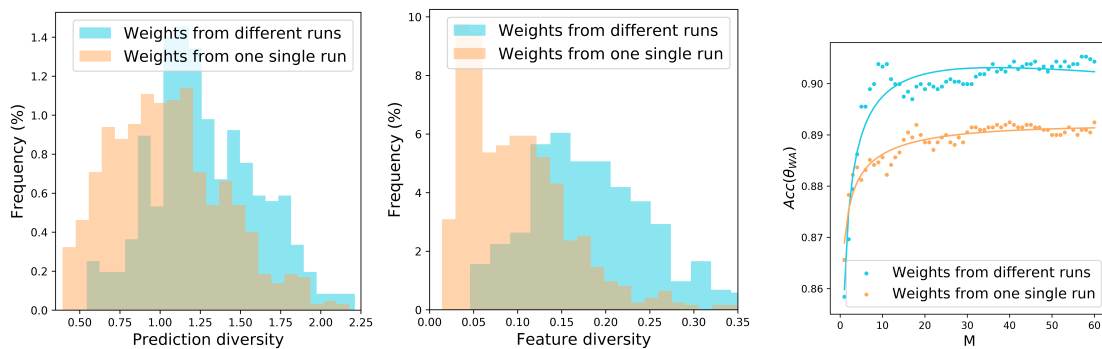
(a) Same as Figure 5.3.   (b) Same as Figure C.5b.   (c) Same as Figure 5.4.

Figure C.10. – Same analysis on PACS as previously done on OfficeHome.

C.5 Number of training runs

In our experiments, we train 20 independent training runs per data split. We selected this value as 20 is the standard number of hyperparameter trials in DomainBed Gulrajani et al. 2021b. In Figure C.11 we ablate this choice on the OOD domain "Art" of OfficeHome. We observe that a larger number of runs leads to improved performance and reduced standard deviation. These results are consistent with our theoretical analysis, as the variance is divided per M in Proposition 5.1. If reducing the training time is critical, one could benefit from significant gains over ERM even with a smaller number of runs: for example, 10 runs seem sufficient in this case. This analysis complements Figure 5.4 — where 60 runs were launched then sorted in increasing validation accuracy.
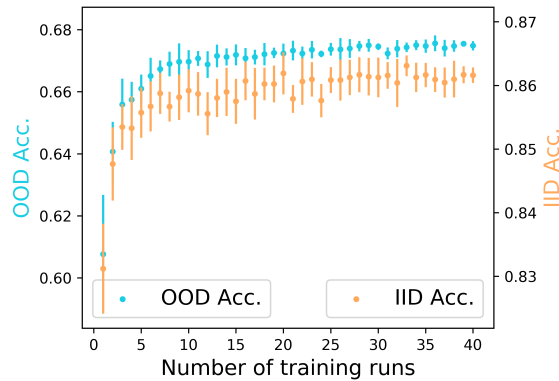
Figure C.11. – **Mean and standard deviation of DiWA-uniform's accuracy (↑)**
on OfficeHome when increasing the number of training runs and
uniformly averaging all weights. OOD accuracy is computed on
domain "Art", while IID accuracy is computed on validation data
from the "Clipart"+"Product"+"Photo" domains.

Moreover, in Table C.4 we report DiWA's results when considering only 5 runs,
with uniform weight selection. Interestingly, it shows that $M = 5$ is enough to be
competitive against SWAD (Cha et al. 2021), the previous state of the art.

Table C.4. – **Accuracy ($\%$, ↑) on DomainBed**. DiWA-uniform and Linear Probing
(LP) initialization Kumar et al. 2022.

Algorithm	PACS	VLCS	OfficeHome	TerraInc	DomainNet	Avg
SWAD (Cha et al. 2021)	88.1 ± 0.1	**79.1 ± 0.1**	70.6 ± 0.2	50.0 ± 0.3	46.5 ± 0.1	66.9
DiWA: $M = 5$	87.9 ± 0.2	78.3 ± 0.3	71.5 ± 0.2	51.0 ± 0.7	46.9 ± 0.3	67.1
DiWA: $M = 20$	88.7 ± 0.2	78.4 ± 0.2	72.1 ± 0.2	51.4 ± 0.6	47.4 ± 0.2	67.6
DiWA†: $M = 60$	**89.0**	78.6	**72.8**	**51.9**	**47.7**	**68.0**

C.6   DomainBed

C.6.1   Description of the DomainBed benchmark

We now further detail our experiments on the DomainBed benchmark Gulrajani
et al. 2021b.

Data.   DomainBed includes several computer vision classification datasets di-
vided into multiple domains. Each domain is successively considered as the test
domain while other domains are used in training. In practice, the data from each

domain is split into 80% (used as training and testing) and 20% (used as validation for hyperparameter selection) splits. This random process is repeated with 3 different seeds: the reported numbers are the means and the standard errors over these 3 seeds.

Training protocol.   We follow the training protocol from https://github.com/facebookresearch/DomainBed. For each dataset, domain and seed, we perform a random search of 20 trials on the hyperparameter distributions described in Table C.5. Our mild distribution is taken directly from Cha et al. 2021, yet could be adapted by dataset for better results. Even though these distributions are more restricted than the extreme distributions introduced Gulrajani et al. 2021b, our ERM runs perform better. It leads to a total amount of 2640 runs only for Table 5.1. In Appendix C.1, the ρ hyperparameter for SAM is sampled from $[0.001, 0.002, 0.005, 0.01, 0.02, 0.05]$. In Table 5.2, hyperparameters specific to Interdomain Mixup (Yan et al. 2020) ("mixup_alpha") and Coral Sun et al. 2016 ("mmd_gamma") are sampled from the distributions defined in Gulrajani et al. 2021b. We use a ResNet50 (He et al. 2016b) pretrained on ImageNet, with a dropout layer before the newly added dense layer and fine-tuned with frozen batch normalization layers. The optimizer is Adam (Kingma et al. 2015). Our classifier is either initialized randomly or with Linear Probing (Kumar et al. 2022); in the latter case, we first learn only the classifier (with the encoder frozen) with the default hyperparameters defined in Table C.5; the classifier's weights are then used to initialize all subsequent runs. All runs are trained for 5k steps, except on DomainNet with 15k steps as done in concurrent works (Cha et al. 2021; Arpit et al. 2021). As in Cha et al. 2021, validation accuracy is calculated every 50 steps for VLCS, 500 steps for DomainNet and 100 steps for others.

Table C.5. – Hyperparameters, their default values and distributions for random search.

Hyperparameter	Default value	Random distribution	
		Extreme (DomainBed Gulrajani et al. 2021b)	Mild (DiWA as Cha et al. 2021)
Learning rate	$5 \cdot 10^{-5}$	$10^{\mathcal{U}(-5,-3.5)}$	$[1, 3, 5] \cdot 10^{-5}$
Batch size	32	$2^{\mathcal{U}(3,5.5)}$	32
ResNet dropout	0	$[0, 0.1, 0.5]$	$[0, 0.1, 0.5]$
Weight decay	0	$10^{\mathcal{U}(-6,-2)}$	$[10^{-6}, 10^{-4}]$

Model selection and scores.   We consider the training-domain validation set protocol. From each run, we thus take the weights of the epoch with maximum accuracy on the validation dataset — which follows the training distribution. Our restricted weight selection is also based on this training-domain validation set.

This strategy is not possible for DiWA† as it averages $M = 20 \times 3$ weights trained with different data splits: they do not share a common validation dataset. The scores for ERM and Coral are taken from DomainBed (Gulrajani et al. 2021b). Scores for SWAD (Cha et al. 2021) and MA (Arpit et al. 2021) are taken from their respective papers. Note that MA and SWAD perform similarly even though SWAD introduced three additional hyperparameters tuned per dataset: "an optimum patient parameter, an overfitting patient parameter, and the tolerance rate for searching the start iteration and the end iteration". Thus we reproduced MA (Arpit et al. 2021) which was much easier to implement, and closer to our uniform weight selection.

C.6.2 DomainBed results detailed per domain for each real-world dataset

Tables below detail results per domain for the 5 multi-domain real-world datasets from DomainBed: PACS Li et al. 2017a, VLCS Fang et al. 2013, OfficeHome Venkateswara et al. 2017b, TerraIncognita Beery et al. 2018b and DomainNet Peng et al. 2019. Critically, Ye et al. 2022 showed that diversity shift dominates in these datasets.

Table C.6. – **Accuracy ($\%, \uparrow$) on PACS with ResNet50** (best in **bold** and second best <u>underlined</u>).

	Algorithm	Weight selection	Init	A	C	P	S	Avg
	ERM	N/A		84.7 ± 0.4	80.8 ± 0.6	97.2 ± 0.3	79.3 ± 1.0	85.5 ± 0.2
	Coral	N/A	Random	88.3 ± 0.2	80.0 ± 0.5	97.5 ± 0.3	78.8 ± 1.3	86.2 ± 0.3
	SWAD	Overfit-aware		89.3 ± 0.5	83.4 ± 0.6	97.3 ± 0.3	82.5 ± 0.8	88.1 ± 0.1
	MA	Uniform		89.1 ± 0.1	82.6 ± 0.2	97.6 ± 0.0	80.5 ± 0.9	87.5 ± 0.2
	DENS	Uniform: $M = 6$		88.3	<u>83.6</u>	96.5	81.9	87.6
	ERM	N/A		87.6 ± 0.4	80.1 ± 1.5	97.7 ± 0.3	76.7 ± 1.2	85.5 ± 0.5
	MA	Uniform		89.9 ± 0.1	83.3 ± 0.4	97.8 ± 0.2	80.6 ± 0.3	87.9 ± 0.1
	ENS	Uniform: $M = 20$	Random	88.9 ± 0.4	82.3 ± 0.5	97.4 ± 0.3	83.2 ± 0.3	88.0 ± 0.1
	DiWA	Restricted: $M \leq 20$		90.0 ± 0.3	82.0 ± 0.5	97.5 ± 0.1	82.0 ± 0.6	87.9 ± 0.2
	DiWA	Uniform: $M = 20$		90.1 ± 0.6	83.3 ± 0.6	<u>98.2 ± 0.1</u>	83.4 ± 0.4	88.8 ± 0.4
Our runs	DiWA†	Uniform: $M = 60$		<u>90.5</u>	**83.7**	<u>98.2</u>	**83.8**	**89.0**
	ERM	N/A		86.8 ± 0.8	80.6 ± 1.0	97.4 ± 0.4	78.7 ± 2.0	85.9 ± 0.6
	MA	Uniform		89.5 ± 0.1	82.8 ± 0.2	97.8 ± 0.1	80.9 ± 1.3	87.8 ± 0.3
	ENS	Uniform: $M = 20$		89.6 ± 0.2	81.6 ± 0.3	97.8 ± 0.2	83.5 ± 0.5	88.1 ± 0.3
	DiWA	Restricted: $M \leq 20$	LP	89.3 ± 0.2	82.8 ± 0.2	98.0 ± 0.1	82.0 ± 0.9	88.0 ± 0.3
	DiWA	Uniform: $M = 5$		89.9 ± 0.5	82.3 ± 0.3	97.7 ± 0.4	81.7 ± 0.8	87.9 ± 0.2
	DiWA	Uniform: $M = 20$		90.1 ± 0.2	82.8 ± 0.6	**98.3 ± 0.1**	83.3 ± 0.4	88.7 ± 0.2
	DiWA†	Uniform: $M = 60$		**90.6**	<u>83.4</u>	<u>98.2</u>	**83.8**	**89.0**

Table C.7. – **Accuracy (%, ↑) on VLCS with ResNet50** (best in **bold** and second best underlined).

	Algorithm	Weight selection	Init	C	L	S	V	Avg
	ERM	N/A		97.7 ± 0.4	64.3 ± 0.9	73.4 ± 0.5	74.6 ± 1.3	77.5 ± 0.4
	Coral	N/A		98.3 ± 0.1	66.1 ± 1.2	73.4 ± 0.3	77.5 ± 1.2	78.8 ± 0.6
	SWAD	Overfit-aware	Random	98.8 ± 0.1	63.3 ± 0.3	75.3 ± 0.5	$\underline{79.2} \pm 0.6$	79.1 ± 0.1
	MA	Uniform		$\mathbf{99.0} \pm 0.2$	63.0 ± 0.2	74.5 ± 0.3	76.4 ± 1.1	78.2 ± 0.2
	DENS	Uniform: $M = 6$		98.7	64.5	72.1	78.9	78.5
	ERM	N/A		97.9 ± 0.5	64.2 ± 0.3	73.5 ± 0.5	74.9 ± 1.2	77.6 ± 0.2
	MA	Uniform		98.5 ± 0.2	63.5 ± 0.2	74.4 ± 0.8	77.3 ± 0.3	78.4 ± 0.1
	ENS	Uniform: $M = 20$	Random	98.6 ± 0.1	$\mathbf{64.9} \pm 0.2$	73.5 ± 0.3	77.7 ± 0.3	78.7 ± 0.1
	DiWA	Restricted: $M \leq 20$		98.3 ± 0.1	63.9 ± 0.2	$\underline{75.6} \pm 0.2$	79.1 ± 0.3	$\underline{79.2} \pm 0.1$
	DiWA	Uniform: $M = 20$		98.4 ± 0.1	63.4 ± 0.1	75.5 ± 0.3	78.9 ± 0.6	79.1 ± 0.2
	DiWA†	Uniform: $M = 60$		98.4	63.3	$\mathbf{76.1}$	$\mathbf{79.6}$	$\mathbf{79.4}$
	ERM	N/A		98.1 ± 0.3	64.4 ± 0.3	72.5 ± 0.5	77.7 ± 1.3	78.1 ± 0.5
	MA	Uniform		$\underline{98.9} \pm 0.0$	62.9 ± 0.5	73.7 ± 0.3	78.7 ± 0.6	78.5 ± 0.4
	ENS	Uniform: $M = 20$		98.5 ± 0.1	$\mathbf{64.9} \pm 0.1$	73.4 ± 0.4	77.2 ± 0.4	78.5 ± 0.1
	DiWA	Restricted: $M \leq 20$	LP	98.4 ± 0.0	64.1 ± 0.2	73.3 ± 0.4	78.1 ± 0.8	78.5 ± 0.1
	DiWA	Uniform: $M = 5$		98.8 ± 0.0	63.8 ± 0.5	72.9 ± 0.2	77.6 ± 0.5	78.3 ± 0.3
	DiWA	Uniform: $M = 20$		98.8 ± 0.1	62.8 ± 0.2	73.9 ± 0.3	78.3 ± 0.1	78.4 ± 0.2
	DiWA†	Uniform: $M = 60$		$\underline{98.9}$	62.4	73.9	78.9	78.6

(rows 6–11 marked "Our runs")

C.7   Failure of WA under correlation shift on ColoredMNIST

Based on Eq. (BVCL), we explained that WA is efficient when variance dominates; we showed in Section 5.2.3 that this occurs under diversity shift. This is confirmed by our state-of-the-art results in Table 5.1 and Appendix E.1 on PACS, OfficeHome, VLCS, TerraIncognita and DomainNet. In contrast, we argue that WA is inefficient when bias dominates, *i.e.* in the presence of correlation shift (see Section 5.2.3). We verify this failure on the ColoredMNIST Arjovsky et al. 2019a dataset, which is dominated by correlation shift Peng et al. 2019.

Colored MNIST is a colored variant of the MNIST handwritten digit classification dataset LeCun et al. 2010 where the correlation strengths between color and label vary across domains. We follow the protocol described in Appendix C.6.1 except that (1) we used the convolutional neural network architecture introduced in DomainBed Gulrajani et al. 2021b for MNIST experiments and (2) we used the test-domain model selection in addition to the train-domain model selection. Indeed, as stated in Ye et al. 2022, "it may be improper to apply training-domain validation to datasets dominated by correlation shift since under the influence of spurious correlations, achieving excessively high accuracy in the training environments often leads to low accuracy in novel test environments".

Table C.8. – **Accuracy (%, ↑) on OfficeHome with ResNet50** (best in **bold** and second best <u>underlined</u>).

	Algorithm	Weight selection	Init	A	C	P	R	Avg
	ERM	N/A		61.3 ± 0.7	52.4 ± 0.3	75.8 ± 0.1	76.6 ± 0.3	66.5 ± 0.3
	Coral	N/A		65.3 ± 0.4	54.4 ± 0.5	76.5 ± 0.1	78.4 ± 0.5	68.7 ± 0.3
	SWAD	Overfit-aware	Random	66.1 ± 0.4	57.7 ± 0.4	78.4 ± 0.1	80.2 ± 0.2	70.6 ± 0.2
	MA	Uniform		66.7 ± 0.5	57.1 ± 0.1	78.6 ± 0.1	80.0 ± 0.0	70.6 ± 0.1
	DENS	Uniform: $M = 6$		65.6	58.5	78.7	80.5	70.8
	ERM	N/A		62.9 ± 1.3	54.0 ± 0.2	75.7 ± 0.9	77.0 ± 0.8	67.4 ± 0.6
	MA	Uniform		65.0 ± 0.2	57.9 ± 0.3	78.5 ± 0.1	79.7 ± 0.1	70.3 ± 0.1
	ENS	Uniform: $M = 20$	Random	66.1 ± 0.1	57.0 ± 0.3	79.0 ± 0.2	80.0 ± 0.1	70.5 ± 0.1
Our runs	DiWA	Restricted: $M \leq 20$		66.7 ± 0.1	57.0 ± 0.3	78.5 ± 0.3	79.9 ± 0.3	70.5 ± 0.1
	DiWA	Uniform: $M = 20$		67.3 ± 0.2	57.9 ± 0.2	79.0 ± 0.2	79.9 ± 0.1	71.0 ± 0.1
	DiWA†	Uniform: $M = 60$		67.7	<u>58.8</u>	79.4	80.5	71.6
	ERM	N/A		63.9 ± 1.2	54.8 ± 0.6	78.7 ± 0.1	80.4 ± 0.2	69.4 ± 0.2
	MA	Uniform		67.4 ± 0.4	57.3 ± 0.9	79.7 ± 0.1	<u>81.7 ± 0.6</u>	71.5 ± 0.3
	ENS	Uniform: $M = 20$		67.0 ± 0.1	57.9 ± 0.4	<u>80.0 ± 0.2</u>	<u>81.7 ± 0.3</u>	71.7 ± 0.1
	DiWA	Restricted: $M \leq 20$	LP	67.8 ± 0.5	57.2 ± 0.5	79.6 ± 0.1	81.4 ± 0.4	71.5 ± 0.2
	DiWA	Uniform: $M = 5$		68.4 ± 0.4	57.4 ± 0.5	79.2 ± 0.2	80.9 ± 0.4	71.5 ± 0.3
	DiWA	Uniform: $M = 20$		68.4 ± 0.2	58.2 ± 0.5	<u>80.0 ± 0.1</u>	<u>81.7 ± 0.3</u>	<u>72.1 ± 0.2</u>
	DiWA†	Uniform: $M = 60$		**69.2**	**59.0**	**80.6**	**82.2**	**72.8**

In Tables C.11 and C.12, we observe that DiWA-uniform and MA both perform poorly compared to ERM. Note that DiWA-restricted does not degrade ERM as it selects only a few models for averaging (low M). This confirms that our approach is useful to tackle diversity shift but not correlation shift, for which invariance-based approaches as IRM Arjovsky et al. 2019a or Fishr Rame et al. 2022 remain state-of-the-art.

C.8   Last layer retraining when some target data is available

The traditional OOD generalization setup does not provide access to target samples (labelled or unlabelled). The goal is to learn a model able to generalize to any kind of distributions. This is arguably the most challenging generalization setup: under these strict conditions, we showed that DiWA outperforms other approaches on DomainBed. Yet, in real-world applications, some target data is often available for training; moreover, last layer retraining on these target samples was shown highly efficient in Kirichenko et al. 2022; Rosenfeld et al. 2022. The complete analysis of DiWA for this new scenario should be properly addressed in future work; yet, we now hint that a DiWA strategy could be helpful.

Specifically, in Table C.13, we consider that after a first training phase on the "Clipart", "Product" and "Photo" domains, we eventually have access to some

Table C.9. – **Accuracy (%, ↑) on TerraIncognita with ResNet50** (best in **bold** and second best <u>underlined</u>).

	Algorithm	Weight selection	Init	L100	L38	L43	L46	Avg
	ERM	N/A		49.8 ± 4.4	42.1 ± 1.4	56.9 ± 1.8	35.7 ± 3.9	46.1 ± 1.8
	Coral	N/A		51.6 ± 2.4	42.2 ± 1.0	57.0 ± 1.0	39.8 ± 2.9	47.6 ± 1.0
	SWAD	Overfit-aware	Random	55.4 ± 0.0	44.9 ± 1.1	59.7 ± 0.4	39.9 ± 0.2	50.0 ± 0.3
	MA	Uniform		54.9 ± 0.4	45.5 ± 0.6	60.1 ± 1.5	40.5 ± 0.4	50.3 ± 0.5
	DENS	Uniform: $M = 6$		53.0	42.6	60.5	40.8	49.2
	ERM	N/A		56.3 ± 2.9	43.1 ± 1.6	57.1 ± 1.0	36.7 ± 0.7	48.3 ± 0.8
	MA	Uniform		53.2 ± 0.4	46.3 ± 1.0	60.1 ± 0.6	40.2 ± 0.8	49.9 ± 0.2
	ENS	Uniform: $M = 20$	Random	56.4 ± 1.5	45.3 ± 0.4	$\mathbf{61.0} \pm 0.3$	$\underline{41.4} \pm 0.5$	51.0 ± 0.5
	DiWA	Restricted: $M \leq 20$		55.6 ± 1.5	47.5 ± 0.5	59.5 ± 0.5	39.4 ± 0.2	50.5 ± 0.5
	DiWA	Uniform: $M = 20$		52.2 ± 1.8	46.2 ± 0.4	59.2 ± 0.2	37.8 ± 0.6	48.9 ± 0.5
	DiWA†	Uniform: $M = 60$		52.7	46.3	59.0	37.7	49.0
	ERM	N/A		$\mathbf{59.9} \pm 4.2$	46.9 ± 0.9	54.6 ± 0.3	40.1 ± 2.2	50.4 ± 1.8
	MAML	Uniform		54.6 ± 1.4	48.6 ± 0.4	59.9 ± 0.7	$\mathbf{42.7} \pm 0.8$	51.4 ± 0.6
	ENS	Uniform: $M = 20$		55.6 ± 1.4	45.4 ± 0.4	$\mathbf{61.0} \pm 0.4$	41.3 ± 0.3	50.8 ± 0.5
	DiWA	Restricted: $M \leq 20$	LP	$\underline{58.5} \pm 2.2$	48.2 ± 0.3	58.5 ± 0.3	41.1 ± 1.2	$\underline{51.6} \pm 0.9$
	DiWA	Uniform: $M = 5$		56.0 ± 2.5	48.9 ± 0.8	58.4 ± 0.2	40.6 ± 0.8	51.0 ± 0.7
	DiWA	Uniform: $M = 20$		56.3 ± 1.9	$\underline{49.4} \pm 0.7$	59.9 ± 0.4	39.8 ± 0.5	51.4 ± 0.6
	DiWA†	Uniform: $M = 60$		57.2	$\mathbf{50.1}$	60.3	39.8	$\mathbf{51.9}$

(Our runs)

samples from the target "Art" domain (20% or 80% of the whole domain). Following Kirichenko et al. 2022, we re-train only the last layer of the network on these samples before testing. We observe improved performance when the (frozen) feature extractor was obtained via DiWA (from the first stage) rather than from ERM. It suggests that features extracted by DiWA are more adapted to last layer retraining/generalization than those of ERM. In conclusion, we believe our DiWA strategy has great potential for many real-world applications, whether some target data is available for training or not.

Table C.10. – **Accuracy (%, ↑) on DomainNet with ResNet50** (best in **bold** and second best underlined).

Algorithm	Weight selection	Init	clip	info	paint	quick	real	sketch	Avg
ERM	N/A		58.1 ± 0.3	18.8 ± 0.3	46.7 ± 0.3	12.2 ± 0.4	59.6 ± 0.1	49.8 ± 0.4	40.9 ± 0.1
Coral	N/A	Random	59.2 ± 0.1	19.7 ± 0.2	46.6 ± 0.3	13.4 ± 0.4	59.8 ± 0.2	50.1 ± 0.6	41.5 ± 0.1
SWAD	Overfit-aware		66.0 ± 0.1	22.4 ± 0.3	53.5 ± 0.1	16.1 ± 0.2	65.8 ± 0.4	55.5 ± 0.3	46.5 ± 0.1
MA	Uniform		64.4 ± 0.3	22.4 ± 0.2	53.4 ± 0.3	15.4 ± 0.1	64.7 ± 0.2	55.5 ± 0.1	46.0 ± 0.1
DENS	Uniform: $M = 6$		**68.3**	23.1	54.5	16.3	66.9	**57.0**	**47.7**
ERM	N/A		62.6 ± 0.4	21.6 ± 0.3	50.4 ± 0.1	13.8 ± 0.2	63.6 ± 0.4	52.5 ± 0.4	44.1 ± 0.1
MA	Uniform		64.5 ± 0.2	22.7 ± 0.1	53.8 ± 0.1	15.6 ± 0.1	66.0 ± 0.1	55.7 ± 0.1	46.4 ± 0.1
ENS	Uniform: $M = 20$	Random	67.3 ± 0.4	22.9 ± 0.1	54.2 ± 0.2	15.5 ± 0.2	67.7 ± 0.2	56.7 ± 0.2	47.4 ± 0.2
DiWA	Restricted: $M \leq 20$		65.2 ± 0.3	23.0 ± 0.3	54.0 ± 0.1	15.9 ± 0.1	66.2 ± 0.1	55.5 ± 0.1	46.7 ± 0.1
DiWA	Uniform: $M = 20$		63.4 ± 0.2	23.1 ± 0.1	53.9 ± 0.2	15.4 ± 0.2	65.5 ± 0.2	55.1 ± 0.2	46.1 ± 0.1
DiWA†	Uniform: $M = 60$		63.5	23.3	54.3	15.6	65.7	55.3	46.3
ERM	N/A		63.4 ± 0.2	21.1 ± 0.4	50.7 ± 0.3	13.5 ± 0.4	64.8 ± 0.4	52.4 ± 0.1	44.3 ± 0.2
MA	Uniform		64.8 ± 0.1	22.3 ± 0.0	54.2 ± 0.1	16.0 ± 0.1	67.4 ± 0.0	55.2 ± 0.1	46.6 ± 0.0
ENS	Uniform: $M = 20$		66.7 ± 0.4	22.2 ± 0.1	54.1 ± 0.2	15.1 ± 0.2	68.4 ± 0.1	55.7 ± 0.2	47.0 ± 0.2
DiWA	Restricted: $M \leq 20$	LP	66.7 ± 0.2	23.3 ± 0.2	55.3 ± 0.1	16.3 ± 0.2	68.2 ± 0.0	56.2 ± 0.1	**47.7** ± 0.1
DiWA	Uniform: $M = 5$		65.7 ± 0.5	22.6 ± 0.2	54.4 ± 0.4	15.5 ± 0.5	67.7 ± 0.0	55.5 ± 0.4	46.9 ± 0.3
DiWA	Uniform: $M = 20$		65.9 ± 0.4	23.0 ± 0.2	55.0 ± 0.3	16.1 ± 0.2	68.4 ± 0.1	55.7 ± 0.4	47.4 ± 0.2
DiWA†	Uniform: $M = 60$		66.2	23.3	55.4	16.5	68.7	56.0	**47.7**

(Rows 6–18 grouped under "Our runs".)

Table C.11. – **Accuracy (%, ↑) on ColoredMNIST**. WA does not improve performance under correlation shift. Random initialization of the classifier. Training-domain model selection.

Algorithm	Weight selection	+90%	+80%	-90%	Avg
ERM	N/A	71.7 ± 0.1	72.9 ± 0.2	10.0 ± 0.1	51.5 ± 0.1
Coral	N/A	71.6 ± 0.3	73.1 ± 0.1	9.9 ± 0.1	51.5 ± 0.1
IRM	N/A	**72.5** ± 0.1	73.3 ± 0.5	10.2 ± 0.3	**52.0** ± 0.1
Fishr	N/A	72.3 ± 0.9	**73.5** ± 0.2	10.1 ± 0.2	**52.0** ± 0.2
ERM	N/A	71.5 ± 0.4	73.3 ± 0.2	10.3 ± 0.2	51.7 ± 0.2
MA	Uniform	68.9 ± 0.0	71.8 ± 0.1	10.0 ± 0.1	50.3 ± 0.0
ENS	Uniform: $M = 20$	71.0 ± 0.2	72.9 ± 0.2	9.9 ± 0.0	51.3 ± 0.1
DiWA	Restricted: $M \leq 20$	71.3 ± 0.2	72.9 ± 0.1	10.0 ± 0.1	51.4 ± 0.1
DiWA	Uniform: $M = 20$	69.1 ± 0.8	72.6 ± 0.4	**10.6** ± 0.1	50.8 ± 0.4
DiWA†	Uniform: $M = 60$	69.3	72.3	10.3	50.6

(Rows 5–10 grouped under "Our runs".)

Table C.12. – **Accuracy (%, ↑) on ColoredMNIST**. WA does not improve performance under correlation shift. Random initialization of the classifier. Test-domain model selection.

Algorithm	Weight selection	+90%	+80%	-90%	Avg
ERM	N/A	71.8 ± 0.4	72.9 ± 0.1	28.7 ± 0.5	57.8 ± 0.2
Coral	N/A	71.1 ± 0.2	73.4 ± 0.2	31.1 ± 1.6	58.6 ± 0.5
IRM	N/A	72.0 ± 0.1	72.5 ± 0.3	58.5 ± 3.3	67.7 ± 1.2
Fishr	N/A	**74.1** ± 0.6	73.3 ± 0.1	**58.9** ± 3.7	**68.8** ± 1.4
ERM	N/A	71.5 ± 0.3	**74.1** ± 0.4	21.5 ± 1.9	55.7 ± 0.4
MA	Uniform	68.8 ± 0.2	72.1 ± 0.2	10.2 ± 0.0	50.4 ± 0.1
ENS	Uniform: $M = 20$	71.0 ± 0.2	72.9 ± 0.2	9.9 ± 0.0	51.3 ± 0.1
DiWA	Restricted: $M \leq 20$	71.9 ± 0.4	73.6 ± 0.2	21.5 ± 1.9	55.7 ± 0.8
DiWA	Uniform: $M = 20$	69.1 ± 0.8	72.6 ± 0.4	10.6 ± 0.1	50.8 ± 0.4
DiWA†	Uniform: $M = 60$	69.3	72.3	10.3	50.6

(Rows 5–10 grouped under "Our runs".)

Table C.13. – **Accuracy (↑)** on domain "Art" from OfficeHome when some target samples are available for last layer retraining (LLR) (Kirichenko et al. 2022). The feature extractor is either pre-trained only on ImageNet (✗), fine-tuned on the source domains "Clipart", "Product" and "Photo" (ERM), or obtained by averaging multiple runs on these source domains (DiWA-uniform $M = 20$).

Training on source domains	LLR on target domain (% domain in training)		
	✗ (0%)	✓ (20%)	✓ (80%)
✗	-	61.2 ± 0.6	74.4 ± 1.2
ERM	62.9 ± 1.3	68.0 ± 0.7	74.7 ± 0.6
DiWA	$\mathbf{67.3} \pm 0.3$	$\mathbf{70.4} \pm 0.1$	$\mathbf{78.1} \pm 0.6$

Appendix D

Supplementary Material of Chapter 6

D.1   Discussion

We discuss in more details the originality and differences of CoDA w.r.t. several Multi-Task Learning (MTL) and gradient-based or contextual meta-learning methods illustrated in Figure D.1. We consider CAVIA (Zintgraf et al. 2019), MAML (Finn et al. 2017), ANIL (Raghu et al. 2020), hard-parameter sharing MTL (Caruana 1997; Ruder 2017), LEADS (Yin et al. 2021a).

D.1.1   Adaptation Rule

We compare the adaptation rule in Eq. (6.4) w.r.t. these work.

GBML   Given k gradient steps, MAML defines

$$\theta^e = \theta^c + (-\eta \sum_{i=0}^{k} \nabla_\theta \mathcal{L}(\theta_i^e, \mathcal{D}^e)) \text{ where } \begin{cases} \theta_{i+1}^e = \theta_i^e - \eta \nabla_\theta \mathcal{L}(\theta_i^e, \mathcal{D}^e) & i > 0 \\ \theta_0^e = \theta^c & i = 0 \end{cases} \quad \text{(D.1)}$$

With $\delta\theta^e \triangleq -\eta \sum_{i=0}^{k} \nabla_\theta \mathcal{L}(\theta_i^e, \mathcal{D}^e)$, Eq. (6.4) thus includes MAML. ANIL and related Gradient-Based Meta Learning (GBML) methods (Lee et al. 2019; Bertinetto et al. 2019) restrict Eq. (D.1) to parameters of the final layer, while remaining parameters are shared.

MTL   MTL models can be identified to Eq. (D.1). They fix $\theta^c \triangleq 0$, removing the ability of performing fast adaptation as parameters are retrained from scratch instead of being initialized to θ^c. Hard-parameter sharing MTL restricts the sum in Eq. (D.1) to the final layer, as ANIL. LEADS sums the outputs of a shared and an environment specific network, thus splits parameters into two independent blocks that do not share connections.
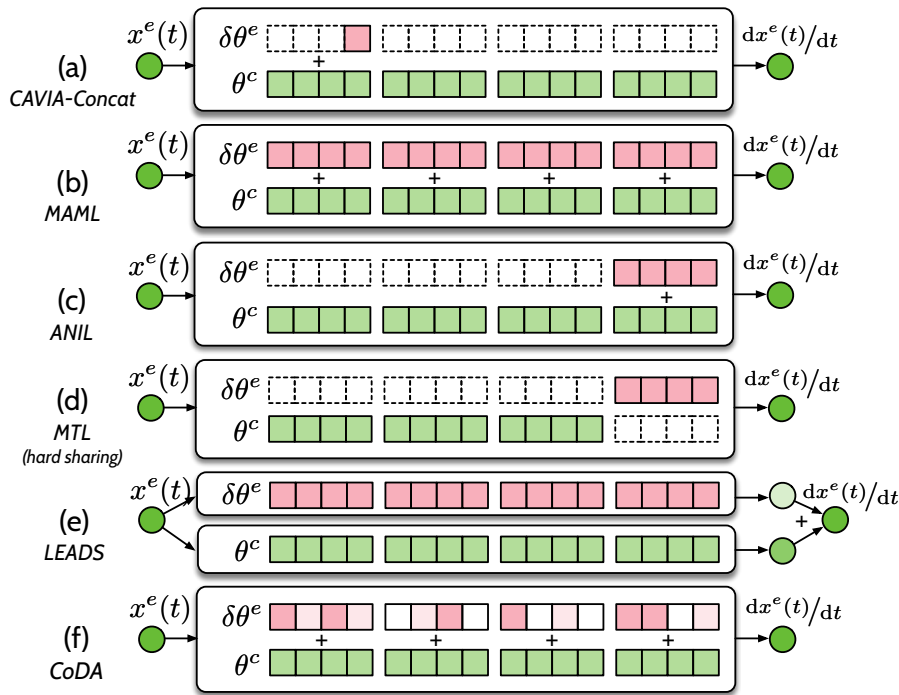
Figure D.1. – Illustration of representative baselines for multi-environment learning. Shared parameters are blue, environment-specific parameters are red. (a) CAVIA-Concat acts upon the bias of the first layer with conditioning via concatenation. (b) MAML acts upon all parameters without penalization nor prior structure information. (c) ANIL restricts meta-learning to the final layer. (d) Hard-sharing MTL train the final layer from scratch, while the remaining is hard-shared. (e) LEADS sums the output of a common and a environment-specific network. (f) CoDA acts upon a subspace of the parameter space with a locality constraint.

D.1.2 Decoding for Context-Informed Adaptation

We show that conditioning strategies in contextual meta-learning for decoding context vectors ξ^e into $\delta\theta^e$ are a special case of hypernetwork-decoding. The two main approaches are conditioning via concatenation and conditioning via feature modulation a.k.a. FiLM (Perez et al. 2018).

Conditioning via Concatenation   We show that conditioning via concatenation is equivalent to a linear hypernetwork $A_\phi : \xi^e \mapsto W\xi^e + \theta^c$ with $\phi = \{\theta^c, W\}$ that only predicts the bias of the first layer of g_θ.

We assume that g_θ has N layers and analyze the output of the first layer of g_θ, omitting the nonlinearity, when the input $x \in \mathbb{R}^{d_x}$ in an environment $e \in \mathcal{E}$ is concatenated to a context vector $\xi^e \in \mathbb{R}^{d_\xi}$. We denote $x\|\xi^e$ the concatenated vector, n_h the number of hidden units of the first layer, $W^1 \in \mathbb{R}^{n_h \times (d_x+d_\xi)}$ and $b^1 \in \mathbb{R}^{n_h}$ the weight matrix and bias term of the first layer, $W^2, \cdots, W^N$ and $b^2, \cdots, b^N$ those of the following layers. The output of the first layer is

$$y^1 = W^1 \cdot x\|\xi^e + b^1$$

We split W^1 along rows into two weight matrices, $W_x^1 \in \mathbb{R}^{n_h \times d_x}; W_\xi^1 \in \mathbb{R}^{n_h \times d_\xi}$ s.t.

$$y^1 = W_x^1 \cdot x + W_\xi^1 \cdot \xi^e + b^1$$

$b_\xi^1 \triangleq W_\xi^1 \cdot \xi^e + b^1$ does not depend on x and corresponds to an environment-specific bias. Thus, concatenation is included in Eq. (6.4) when

$$\theta^c \triangleq \{W_x^1, b^1, W^2, b^2, \cdots, W^N, b^N\}$$
$$\delta\theta^e \triangleq \{\ 0\ , b_\xi^1,\ 0\ , 0, \cdots,\ \ 0\ \ , 0\}$$

$\delta\theta^e$ is decoded via a hypernetwork with parameters $\{\theta^c, W \triangleq (0, W_\xi^1, 0, \cdots, 0)\}$.

Conditioning via Feature Modulation   We show that conditioning via FiLM is equivalent to a linear hypernetwork $A_\phi : \xi^e \mapsto W\xi^e + \theta^c$ with $\phi = \{\theta^c, W\}$ that only predicts the batch norm (BN) statistics of g_θ.

For simplicity, we focus on a single BN layer and denote $\{h_i\}_{i=1}^M$, M feature maps output by preceding convolutional layers. These feature maps are first normalized then rescaled with an affine transformation. Rescaling is similar to a FiLM layer that transforms linearly $\{h_i\}_{i=1}^M$ with:

$$\forall i \in \{1, \cdots, M\}, \mathrm{FiLM}(h_i) = \gamma_i \odot h_i + \beta$$

where $\gamma, \beta \in \mathbb{R}^M$ are output by a NN f_ψ conditioned on the context vectors ξ^e *i.e.* $[\gamma, \beta] = f_\psi(\xi^e)$. In general, f_ψ is linear s.t. $f_\psi(\xi^e) \triangleq W_\xi \xi^e + b_\xi$, with $\psi = \{W_\xi, b_\xi\}$. Then $\gamma = W_\xi^\gamma \xi^e + b_\xi^\gamma$, $\beta = W_\xi^\beta \xi^e + b_\xi^\beta$.

Thus, for this layer, modulation is included in Eq. (6.4) when

$$\delta\theta^e \triangleq W\xi^e = \{W_\xi^\gamma \xi^e, W_\xi^\beta \xi^e\}$$
$$\theta^c \triangleq b_\xi = \{b_\xi^\gamma, b_\xi^\beta\}$$

$\delta\theta^e$ is decoded via hypernetwork $f_\psi \triangleq A_\phi$ with parameters $\phi = \{\theta^c \triangleq b_\xi, W \triangleq W_\xi\}$.

D.2   Proofs

Proposition 6.3. Given a class of linearly parametrized dynamics $\mathcal{F}$ with d_p varying parameters, $\forall \theta^c \in \mathbb{R}^{d_\theta}$, subspace $\mathcal{G}_{\theta^c}$ in Definition 6.2 is low-dimensional and satisfies $\dim(\mathcal{G}_{\theta^c}) \leq d_p \ll d_\theta$.

Proof. We define the linear mapping $\psi : p \in \mathbb{R}^{d_p} \to f \in \mathcal{F}$ from parameters to dynamics s.t. $\psi(\mathbb{R}^{d_p}) = \mathcal{F}$. Given this linear mapping, we first prove the following lemma: $\dim(\mathcal{F}) \leq d_p$. The proof is based on surjectivity of ψ onto $\mathcal{F}$, given by definition. We define $\{b_i\}_{i=1}^{d_p}$ a basis of $\mathbb{R}^{d_p}$. Given $f \in \mathcal{F}$, $\exists p \in \mathbb{R}^{d_p}, \psi(p) = f$. We note $p = \sum_{i=1}^{d_p} \lambda_i b_i$ where $\forall i, \lambda_i \in \mathbb{R}$. Then $\psi(p) = \sum_{i=1}^{d_p} \lambda_i \psi(b_i)$. We extract a basis from $\{\psi(b_i)\}_{i=1}^{d_p}$ and denote $d_f \leq d_p$ the number of elements in this basis. This basis forms a basis of $\mathcal{F}$ *i.e.* $d_f = \dim(\mathcal{F}) \leq d_p$.

Now, given $\theta \in \mathbb{R}^{d_\theta}$ and $f^e \in \mathcal{F}$. We precise that given a (probability) measure $\rho_\mathcal{V}$ on $\mathcal{V} \subset \mathbb{R}^d$, the function space $\mathcal{F} \subset L^2(\rho_x, \mathbb{R}^d)$, then

$$\mathcal{L}(\theta, \mathcal{D}^e) \triangleq \int_\mathcal{V} \|(f^e - g_\theta)(v)\|_2^2 \mathrm{d}\rho_\mathcal{V}(v) = \|f^e - g_\theta\|_2^2$$

The gradient of $\mathcal{L}(\theta, \mathcal{D}^e)$ is then

$$\nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e) = \nabla_\theta \int_\mathcal{V} \|f^e(v) - g_{\theta^c}(v)\|_2^2 \mathrm{d}\rho_\mathcal{V}(v)$$
$$= \int_\mathcal{V} \nabla_\theta \|f^e(v) - g_{\theta^c}(v)\|_2^2 \mathrm{d}\rho_\mathcal{V}(v)$$
$$= -2 \int_\mathcal{V} \mathbf{J}_\theta g_{\theta^c}(v)^\top (f^e(v) - g_{\theta^c}(v)) \mathrm{d}\rho_\mathcal{V}(v)$$
$$= -2 D_\theta g_{\theta^c}^\top (f^e - g_{\theta^c})$$

where $\mathbf{J}_\theta g_\theta(v)$ is the Jacobian matrix of g_{θ^c} w.r.t. θ at point x. $\theta \mapsto D_\theta g_{\theta^c}$ is the differential of g_θ. Note that $D_\theta g_{\theta^c} : \mathbb{R}^{d_\theta} \to \mathcal{F}$ is a linear map (analogue of Jacobian

matrix). $D_\theta g_{\theta^c}^\top : \mathcal{F}^\star \to \mathbb{R}^{d_\theta}$ denotes its adjoint (analogue of transposed matrix), which is also a linear map.

As $\mathcal{G}_{\theta^c} \subseteq \mathrm{Im}(D_\theta g_\theta^\top)$, then according to Rank-nullity theorem,

$$\dim(\mathcal{G}_{\theta^c}) \leq \dim(\mathrm{Im}(D_\theta g_{\theta^c}^\top)) = \dim(\mathcal{F}) - \dim(Ker(D_\theta g_{\theta^c}^\top)) \leq \dim(\mathcal{F}) \leq d_p$$

$\square$

Proposition 6.1. Given $\{\theta^c, W\}$ fixed, if $\|\cdot\| = \ell_2$, then Eq. (6.8) is quadratic. If $\lambda' W^\top W$ or $\bar{H}^e(\theta^c) = W^\top \nabla_\theta^2 \mathcal{L}(\theta^c, \mathcal{D}^e) W$ are invertible then $\bar{H}^e(\theta^c) + \lambda' W^\top W$ is invertible except for a finite number of λ' values. The problem in Eq. (6.8) is then also convex and admits an unique solution, $\{\xi^{e\star}\}_{e \in \mathcal{E}_{\mathrm{ad}}}$. With $\lambda' \triangleq 2\lambda$,

$$\xi^{e*} = -\left(\bar{H}^e(\theta^c) + \lambda' W^\top W\right)^{-1} W^\top \nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e)$$

$\bar{H}^e(\theta^c) + \lambda' W^\top W$ is invertible $\forall \lambda'$ except a finite number of values if $\bar{H}^e(\theta^c)$ or $\lambda' W^\top W$ is invertible.

Proof. When $\|\cdot\| = \ell_2$, we consider the following second order Taylor expansion of $\mathcal{L}_\mathrm{r}(\theta, \mathcal{D}^e) \triangleq \mathcal{L}(\theta, \mathcal{D}^e) + \lambda \|\theta - \theta^c\|_2^2$ at θ^c, where $\delta\theta^e = \theta - \theta^c = W\xi^e$.

$$\mathcal{L}_\mathrm{r}(\theta^c + \delta\theta^e, \mathcal{D}^e) = \mathcal{L}(\theta^c, \mathcal{D}^e) + \nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e)^\top \delta\theta^e +$$
$$\frac{1}{2}\delta\theta^{e\top}\left(\nabla_\theta^2 \mathcal{L}(\theta^c, \mathcal{D}^e) + 2\lambda\mathrm{Id}\right)\delta\theta^e + o(\|\delta\theta^e\|_2^3) \quad \text{(D.2)}$$

With $\delta\theta^e = W\xi^e$, we expand Eq. (D.2) into

$$\mathcal{L}_\mathrm{r}(\theta^c + W\xi^e, \mathcal{D}^e) = \mathcal{L}(\theta^c, \mathcal{D}^e) + (W^\top \nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e))^\top \xi^e$$
$$+ \frac{1}{2}\xi^{e\top}(W^\top \nabla_\theta^2 \mathcal{L}(\theta^c, \mathcal{D}^e)W + 2\lambda W^\top W)\xi^e + o(\|\delta\theta^e\|_2^3)$$

i.e. with $\bar{H}^e(\theta^c) = W^\top \nabla_\theta^2 \mathcal{L}(\theta^c, \mathcal{D}^e)W$ and $\lambda' = 2\lambda$

$$\mathcal{L}_\mathrm{r}(\theta^c + W\xi^e, \mathcal{D}^e) = \mathcal{L}(\theta^c, \mathcal{D}^e) + (W^\top \nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e))^\top \xi^e$$
$$+ \frac{1}{2}\xi^{e\top}\left(\bar{H}^e(\theta^c) + \lambda' W^\top W\right)\xi^e + o(\|\delta\theta^e\|_2^3) \quad \text{(D.3)}$$

Eq. (D.3) is quadratic. If $\bar{H}^e(\theta^c) + \lambda' W^\top W$ is invertible, then the problem is also convex with unique solution

$$\xi^{e*} = -\left(\bar{H}^e(\theta^c) + \lambda' W^\top W\right)^{-1} W^\top \nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e)$$

$\bar{H}^e(\theta^c)$ and $\lambda' W^\top W$ are two square matrices. The application $p : \lambda' \mapsto \det(\bar{H}^e(\theta^c) + \lambda' W^\top W)$ is well-defined and forms a continuous polynomial. Thus either it equals zero or it has a finite number of roots. If $\bar{H}^e(\theta^c)$ or $\lambda' W^\top W$ is invertible, then $p(0) = \det(\bar{H}^e(\theta^c)) \neq 0$ or $p(\infty) \sim \det(\lambda' W^\top W) \neq 0$. Thus $p \neq 0$ has a finite number of roots *i.e.* $\bar{H}^e(\theta^c) + \lambda' W^\top W$ is invertible $\forall \lambda'$ except a finite number of values corresponding to the roots of p. $\qquad\square$

D.3 System Parameter Estimation

Proposition 6.4. Under Assumptions 12 to 16, system parameters are perfectly identified on new environments if the dynamics model g and hypernetwork A satisfy $\forall f \in \mathcal{B}$ with system parameter p, $g_{A(p)} = f$.

Proof. We define the linear mapping $\psi : p \in \mathbb{R}^{d_p} \to f \in \mathcal{F}$ from parameters to dynamics s.t. $\psi(\mathbb{R}^{d_p}) = \mathcal{F}$ (Assumption 12). Unicity of parameters (Assumption 14) implies that ψ is bijective with inverse ψ^{-1}, thus $\dim(\mathcal{F}) = \dim(\mathbb{R}^{d_p}) = d_p$. Given a basis $\mathcal{B} = \{f_i\}_{i=1}^{d_p}$ of $\mathcal{F}$, we denote $p_i = \psi^{-1}(f_i)$. We fix g, A s.t. $\forall i \in \{1, ..., d_p\}, g_{A(p_i)} = f_i = \psi(p_i)$. This is possible as f_i and g are linear w.r.t. inputs (Assumptions 12 and 13) and p_i are known (Assumption 16).

g, A are linear (Assumption 13), thus $g_{A(\cdot)}$ is linear with inputs in $\mathbb{R}^{d_\xi}$. Then, $\dim(\mathrm{Im}(g_{A(\cdot)})) \leq d_\xi$. Moreover, $\forall i \in \{1, ..., d_p\}, f_i \in \mathrm{Im}(g_{A(\cdot)})$, thus $\mathcal{F} \subset \mathrm{Im}(g_{A(\cdot)})$ *i.e.* $d_p \leq \dim(\mathrm{Im}(g_{A(\cdot)}))$. Thus, $d_p \leq \dim(\mathrm{Im}(g_{A(\cdot)})) \leq d_\xi$. Assumption 15 states that $d_\xi = d_p$, s.t. $\dim(\mathrm{Im}(g_{A(\cdot)})) = d_p$. As $\mathcal{F} \subset \mathrm{Im}(g_{A(\cdot)})$ and $\dim(\mathcal{F}) = \dim(\mathrm{Im}(g_{A(\cdot)}))$, $\mathcal{F} = \mathrm{Im}(g_{A(\cdot)})$ *i.e.* $g_{A(\cdot)}$ is surjective onto $\mathcal{F}$. As $\dim(\mathcal{F}) = d_\xi$, the dimension of the input space, $g_{A(\cdot)}$ is bijective.

By bijectivity of ψ, $\{p_i\}_{i=1}^{d_p}$ forms a basis of $\mathbb{R}^{d_p}$. $g_{A(\cdot)}$ and ψ map this basis to the same basis $\{f_i\}_{i=1}^{d_p}$ of $\mathcal{F}$. As both mappings are bijective, this implies that $g_{A(\cdot)} = \psi(\cdot)$. This means that $\forall e \in \mathcal{E}, g_A^{-1}(f^e) = \psi^{-1}(f^e)$ *i.e.* system parameters p^e are recovered. $\qquad\square$

Proposition 6.5. For linearly parametrized systems, nonlinear w.r.t. inputs and nonlinear dynamics model g_θ with parameters output by a linear hypernetwork A, $\exists \alpha > 0$ s.t. system parameters are perfectly identified $\forall e \in \mathcal{E}$ where $\|\xi^e\| \leq \alpha$ if $\forall f \in \mathcal{B}$ with parameter p, $g_{A(\alpha \frac{p}{\|p\|})} = f$.

Proof. On environment $e \in \mathcal{E}$, g_{θ^e} is differentiable w.r.t. $\theta^e = A(\xi^e) = \theta^c + W\xi^e \in \mathbb{R}^{d_\theta}$. We perform a first order Taylor expansion of $g_{A(\cdot)}$ around $\mathbf{0}$. We note $\alpha > 0$, s.t. $\forall \xi^e \in \mathbb{R}^{d_\xi}$ that satisfy $\|\xi^e\| < \alpha$, we have $g_{\theta^e} = g_{\theta^c} + \nabla_\theta g_{\theta^c} W\xi^e$. $g_{A(\cdot)}$ is then linear in the neighborhood of $\mathbf{0}$ defined by α. $\forall i \in [\![1, d_p]\!], \alpha \frac{p_i}{\|p_i\|}$ belongs to this

neighborhood s.t. the proof of Proposition 6.4 applies to this neighborhood if $\forall i \in [\![1, d_p]\!], g_{A(\alpha \frac{p_i}{\|p_i\|})} = f_i$, where $\mathcal{B} = \{f_i\}_{i=1}^{d_p}$ is a basis of $\mathcal{F}$. □

We now show the validity of the unicity condition (Assumption 14) for two linearly parametrized systems.

Lemma D.1. *There is an unique set of parameters in $\mathbb{R}^4$ for a Lotka-Volterra (LV) system.*

Proof. With $\psi : c \triangleq (\alpha, \beta, \delta, \gamma) \mapsto \left[\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} \alpha x - \beta xy \\ \delta xy - \gamma y \end{pmatrix}\right]$ a surjective linear mapping from $\mathbb{R}^4$ to $\mathcal{F}$ (all LV systems are parametrized). Injectivity of ψ *i.e.* $\psi(c_1) = \psi(c_2) \iff c_1 = c_2$ will imply bijectivity *i.e.* unicity of parameters for a LV system. As ψ is linear, injectivity is equivalent to $\psi(c) = 0 \iff c = 0$, shown below:

$$\psi(c) = 0 \iff \forall \begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} x(\alpha - \beta y) \\ (\delta x - \gamma)y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\iff \forall \begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} \alpha - \beta y \\ \delta x - \gamma \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\iff c = (\alpha, \beta, \delta, \gamma) = (0, 0, 0, 0)$$

□

Lemma D.2. *There is an unique set of parameters in $\mathbb{R}^{d+1}$, where d is the grid size, for a Navier-Stokes (NS) system.*

Proof. With $\psi : c \triangleq (\nu, f) \mapsto \left[w \mapsto -v\nabla w + \nu \Delta w + f\right]$, a surjective linear mapping from $\mathbb{R}^{d+1}$ to $\mathcal{F}$ (all NS systems are parametrized), bijectivity of ψ is induced by injectivity *i.e.* $\psi(c_1) = \psi(c_2) \iff c_1 = c_2$, shown below:

$$\psi(c_1) = \psi(c_2)$$

$$\iff \forall w, -v\nabla w + \nu_1 \Delta w + f_1 = -v\nabla w + \nu_2 \Delta w + f_2$$

$$\iff \forall w, (\nu_1 - \nu_2)\Delta w = -(f_1 - f_2)$$

$$\iff (\nu_1, f_1) = (\nu_2, f_2) \iff c_1 = c_2$$

□

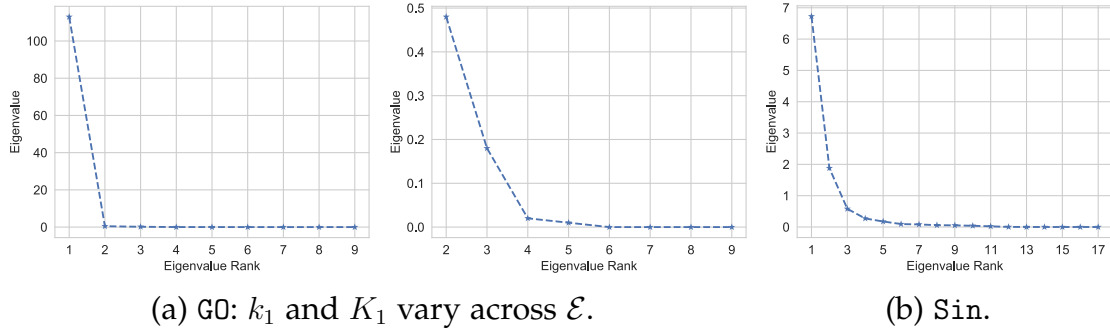(a) GO: k_1 and K_1 vary across $\mathcal{E}$.   (b) Sin.

Figure D.2. – Ranked singular values of the gradients across environments $\mathcal{E}_{\text{tr}}$, $\mathcal{G}_{\theta^c}$ for CoDA-ℓ_1.

D.4   Low-Rank Assumption

When the systems are nonlinearly parametrized, we show empirically with Figure D.2 that the low-rank assumption is still reasonable for two different systems.

Glycolitic-Oscillator (GO)   We consider the Glycolitic-Oscillator system (GO), described in Appendix D.6.1, which is nonlinear w.r.t. K_1. We vary parameters k_1, K_1 in Eq. (D.5) across environments. We observe in Figure D.2a that there are three main gradient directions with SVD. The first is the most significant one while the second and third ones are orders of magnitude smaller.

Sinusoidal (Sin)   We consider a sinusoidal family of functions $S(n) = \{f : \mathbb{R} \to \mathbb{R} | f(x) = \sum_{i=1}^{N} \lambda_i \sin(\omega_i x + \phi_i)\}$ (Sin). We sample 20 environments that correspond each to different amplitudes (uniformly sampled in $[0, 1]$), frequencies (uniformly sampled in $[0, 10]$)) and phases (uniformly sampled in $[0, 3.14]$). We depict in Figure D.2b the evaluation of the singular values at initialization. Figure D.2b shows that the number of directions to consider for convergence is small and that a single direction accounts for a significant amount of the variance in the gradients. This corroborates the low-rank assumption.

D.5   Locality Constraint

We derive the upper-bounds to $\| \cdot \|$ for two variations.

$\| \cdot \| = \ell_2$: we apply triangle inequality to obtain $\Omega = \ell_2^2$

$$\|W\xi^e\|_2^2 \leq \|W\|_2^2 \|\xi^e\|_2^2$$

$\| \cdot \| = \ell_1$: we apply Cauchy-Schwartz inequality to obtain $\Omega(W) = \ell_{1,2}(W) \triangleq \sum_{i=1}^{d_\theta} \|W_{i,:}\|_2$

$$\|W\xi^e\|_1 = \sum_{i=1}^{d_\theta} |W_{i,:}\xi^e| \leq \|\xi^e\|_2 \sum_{i=1}^{d_\theta} \|W_{i,:}\|_2$$

Eq. (6.11) minimizes the $\log$ of the above upper-bounds.

D.6 Experimental Settings

We present in Appendix D.6.1 the equations and the data generation specificities for all considered dynamical systems.

D.6.1 Dynamical Systems

Lotka-Volterra (LV, (Lotka 1925))   The system describes the interaction between a prey-predator pair in an ecosystem, formalized into the following Ordinary Differential Equation (ODE):

$$\frac{dx}{dt} = \alpha x - \beta xy$$
$$\frac{dy}{dt} = \delta xy - \gamma y \tag{D.4}$$

where x, y are respectively the quantity of the prey and the predator, $\alpha, \beta, \delta, \gamma$ define how two species interact.

We generate trajectories on a temporal grid with $\Delta t = 0.5$ and temporal horizon $T = 10$. We sample on each training environment $N_{\text{tr}} = 4$ initial conditions for training from a uniform distribution $p(V_0) = \text{Unif}([1,3]^2)$. We sample for evaluation 32 initial conditions from $p(V_0)$. Across environments, $\alpha = 0.5, \gamma = 0.5$. For training, we consider $\#\mathcal{E}_{\text{tr}} = 9$ environments with parameters $\beta, \delta \in \{0.5, 0.75, 1.0\}^2$. For adaptation, we consider $\#\mathcal{E}_{\text{ad}} = 4$ environments with parameters $\beta, \delta \in \{0.625, 1.125\}^2$.

Glycolytic-Oscillator (GO, (Daniels et al. 2015))   GO describes yeast glycolysis dynamics with the ODE:

$$
\begin{aligned}
\frac{\mathrm{d}S_1}{\mathrm{d}t} &= J_0 - \frac{k_1 S_1 S_6}{1 + (1/K_1^q)S_6^q} \\
\frac{\mathrm{d}S_2}{\mathrm{d}t} &= 2\frac{k_1 S_1 S_6}{1 + (1/K_1^q)S_6^q} - k_2 S_2(N - S_5) - k_6 S_2 S_5 \\
\frac{\mathrm{d}S_3}{\mathrm{d}t} &= k_2 S_2(N - S_5) - k_3 S_3(A - S_6) \\
\frac{\mathrm{d}S_4}{\mathrm{d}t} &= k_3 S_3(A - S_6) - k_4 S_4 S_5 - \kappa(S_4 - S_7) \\
\frac{\mathrm{d}S_5}{\mathrm{d}t} &= k_2 S_2(N - S_5) - k_4 S_4 S_5 - k_6 S_2 S_5 \\
\frac{\mathrm{d}S_6}{\mathrm{d}t} &= -2\frac{k_1 S_1 S_6}{1 + (1/K_1^q)S_6^q} + 2k_3 S_3(A - S_6) - k_5 S_6 \\
\frac{\mathrm{d}S_7}{\mathrm{d}t} &= \psi\kappa(S_4 - S_7) - kS_7
\end{aligned}
\tag{D.5}
$$

where $S_1, S_2, S_3, S_4, S_5, S_6, S_7$ represent the concentrations of 7 biochemical species. We generate trajectories on a temporal grid with $\Delta t = 0.05$ and temporal horizon $T = 1$. We sample on each training environment $N_{\text{tr}} = 32$ initial conditions for training from a uniform distribution $p(V_0)$ defined in Table 2 in (Daniels et al. 2015). Across environments, $J_0 = 2.5, k_2 = 6, k_3 = 16, k_4 = 100, k_5 = 1.28, k_6 = 12, q = 4, N = 1, A = 4, \kappa = 13, \psi = 0.1, k = 1.8$. For training, we consider $\#\mathcal{E}_{\text{tr}} = 9$ environments with parameters $k_1 \in \{100, 90, 80\}, K_1 \in \{1, 0.75, 0.5\}$. For adaptation, we consider $\#\mathcal{E}_{\text{ad}} = 4$ environments with parameters $k_1 \in \{85, 95\}, K_1 \in \{0.625, 0.875\}$.

Gray-Scott (GS, (Pearson 1993))   The Partial Differential Equation (PDE) descibes a reaction-diffusion system with complex spatiotemporal patterns through the following 2D PDE:

$$
\begin{aligned}
\frac{\partial u}{\partial t} &= D_u \Delta u - uv^2 + F(1 - u) \\
\frac{\partial v}{\partial t} &= D_v \Delta v + uv^2 - (F + k)v
\end{aligned}
\tag{D.6}
$$

where u, v represent the concentrations of two chemical components in the spatial domain S with periodic boundary conditions. D_u, D_v denote the diffusion coefficients respectively for u, v and F, k are the reaction parameters.

We generate trajectories on a temporal grid with $\Delta t = 40$ and temporal horizon $T = 400$. S is a 2D space of dimension 32×32 with spatial resolution of $\Delta s = 2$. We define initial conditions $(u_0, v_0) \sim p(V_0)$ by uniformly sampling three two-by-two squares in S. These squares trigger the reactions. $(u_0, v_0) = (1 - \epsilon, \epsilon)$ with

$\epsilon = 0.05$ inside the squares and $(u_0, v_0) = (0, 1)$ outside the squares. We sample on each training environment $N_{\text{tr}} = 1$ initial conditions for training. Across environments, $D_u = 0.2097, D_v = 0.105$. For training, we consider $\#\mathcal{E}_{\text{tr}} = 4$ environments with parameters $F \in \{0.30, 0.39\}, k \in \{0.058, 0.062\}$. For adaptation, we consider $\#\mathcal{E}_{\text{ad}} = 4$ environments with parameters $F \in \{0.33, 0.36\}, k \in \{0.59, 0.61\}$.

Navier-Stokes (NS, (Stokes 1851))   NS describes the dynamics of incompressible flows with the 2D PDE:

$$\frac{\partial w}{\partial t} = -v\nabla w + \nu\Delta w + f \text{ where } w = \nabla \times v$$
$$\nabla v = 0$$
(D.7)

where v is the velocity field, $w = \nabla \times v$ is the vorticity. Both v, w lie in a spatial domain S with periodic boundary conditions, ν is the viscosity and f is the constant forcing term in the domain S. We generate trajectories on a temporal grid with $\Delta t = 1$ and temporal horizon $T = 10$. S is a 2D space of dimension 32×32 with spatial resolution of $\Delta s = 1$. We sample on each training environment $N_{\text{tr}} = 16$ initial conditions for training from $p(V_0)$ as in Li et al. (2021c). Across environments, $f(X, Y) = 0.1(\sin(2\pi(X + Y)) + \cos(2\pi(X + Y)))$. For training, we consider $\#\mathcal{E}_{\text{tr}} = 5$ environments with parameters $\nu \in \{8 \cdot 10^{-4}, 9 \cdot 10^{-4}, 1.0 \cdot 10^{-3}, 1.1 \cdot 10^{-3}, 1.2 \cdot 10^{-3}\}$. For adaptation, we consider $\#\mathcal{E}_{\text{ad}} = 4$ environments with parameters $\nu \in \{8.5 \cdot 10^{-4}, 9.5 \cdot 10^{-4}, 1.05 \cdot 10^{-3}, 1.15 \cdot 10^{-3}\}$.

D.6.2   Implementation and Hyperparameters

Architecture   We implement the dynamics model g_θ with the following architectures:

- LV, GO: 4-layer MLPs with hidden layers of width 64.

- GS: 4-layer ConvNet with 64-channel hidden layers, and 3×3 convolution kernels

- NS: Fourier Neural Operator (Li et al. 2021c) with 4 spectral convolution layers. 12 frequency modes and hidden layers with width 10.

We apply Swish activation (Ramachandran et al. 2018). The hypernet A is a single affine layer NN.

Optimizer   We use the Adam optimizer (Kingma et al. 2015) with learning rate 10^{-3} and $(\beta_1, \beta_2) = (0.9, 0.999)$. We apply early stopping. All experiments

are performed with a single NVIDIA Titan Xp GPU on an internal cluster. We distribute training by batching together predictions across trajectories to reduce running time. States across batch elements are concatenated.

Hyperparameters   We define hyperparameters for the following models: CoDA:

- LV: $\lambda_\xi = 10^{-4}$, $\lambda_{\ell_1} = 10^{-6}$, $\lambda_{\ell_2} = 10^{-5}$

- GO: $\lambda_\xi = 10^{-3}$, $\lambda_{\ell_1} = 10^{-7}$, $\lambda_{\ell_2} = 10^{-7}$

- GS: $\lambda_\xi = 10^{-2}$, $\lambda_{\ell_1} = 10^{-5}$, $\lambda_{\ell_2} = 10^{-5}$

- NS: $\lambda_\xi = 10^{-3}$, $\lambda_{\ell_1} = 2 \cdot 10^{-3}$, $\lambda_{\ell_2} = 2 \cdot 10^{-3}$

LEADS: we use the same parameters as Yin et al. (2021a).

GBML: the outer-loop learning rate is 10^{-3}, we apply 1-step inner-loop update for training and adaptation to maintain low running times. The inner-loop learning rate for each system is:

- LV: 0.1

- GO: 0.01

- GS: 10^{-3}

- NS: 10^{-3}

. These values are also used to initialize the per-parameter inner-loop learning rate in Meta-SGD.

D.7   Trajectory Prediction Visualization

We visualize in Figures D.3 and D.4 the prediction Mean-Squared Error (MSE) by MAML, LEADS, CAVIA-Concat and CoDA-ℓ_1 along ground truth trajectories on the PDE systems NS and GS. We consider a new test trajectory on an *Adaptation* environment $e \in \mathcal{E}_{\mathrm{ad}}$ with parameters defined in the caption.
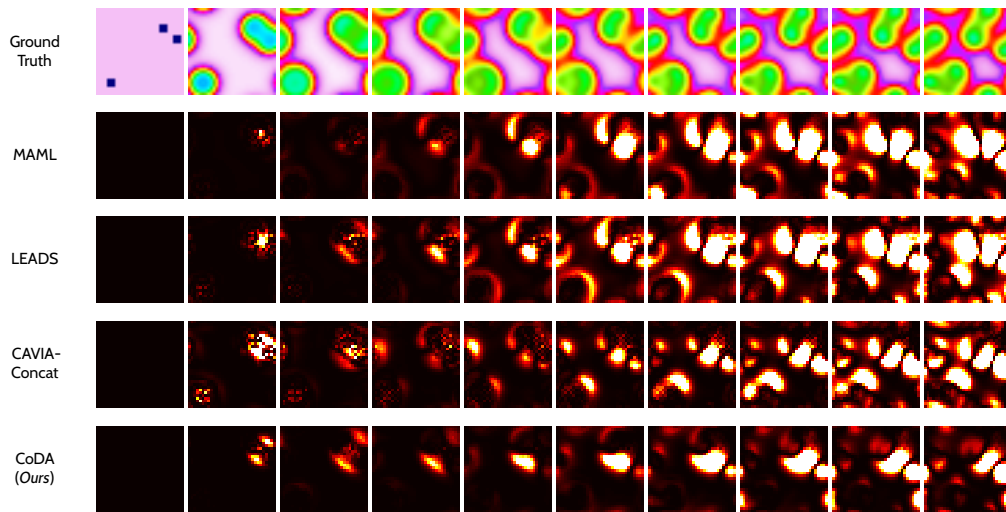
Figure D.3. – Adaptation to new GS system - $F = 0.033$, $k = 0.061$, $D_u = 0.2097$, $D_v = 0.105$. Ground-truth trajectory and prediction MSE per frame for MAML, LEADS, CAVIA-Concat and CoDA.
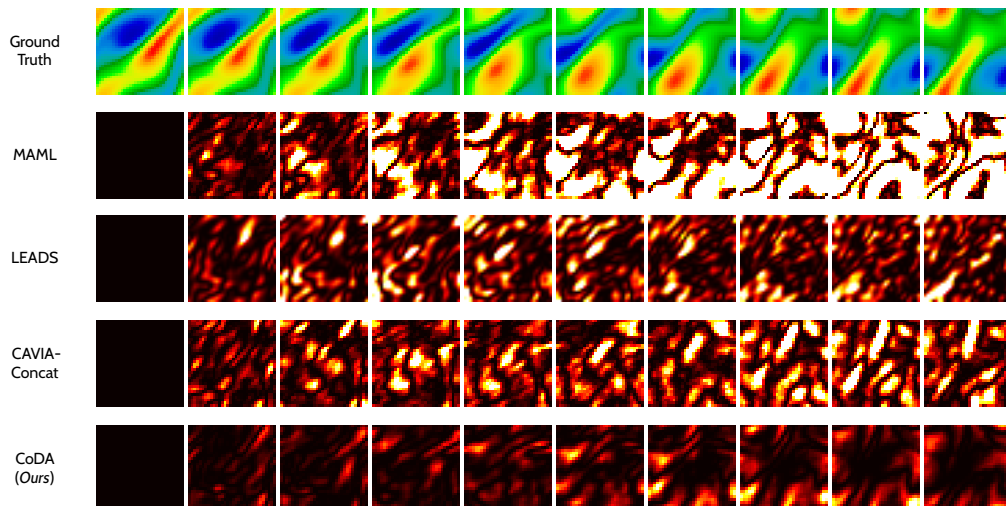

Figure D.4. – Adaptation to new NS system - $\nu = 1.15 \cdot 10^{-3}$. Ground-truth trajectory and prediction MSE per frame for MAML, LEADS, CAVIA-Concat and CoDA.

Appendix E

Supplementary Material of Chapter 7

E.1 Full results

We provide in Table E.1 a more detailed version of Table 7.2 for the space-time extrapolation problem where we report the performance *In-s* (on the observation grid) and *Out-s* (outside). We add $s = 50\%$.

Then, we report in Table E.2, a more detailed version of Table 7.3a, which includes the results of $\mathcal{X}_{\text{ts}} = \mathcal{X}_{\text{tr}}$. This corresponds to our generalization across grids problem.

E.2 Prediction

We display the test prediction of DINo (Figure E.1) and I-MP-PDE (Figure E.2) on *Navier-Stokes* for various subsampling levels when $\mathcal{X} = \mathcal{X}_{\text{tr}} = \mathcal{X}_{\text{ts}}$. We plot the prediction of DINo on *Wave* in the same setting in Figure E.3. Predictions are performed on a 64×64 uniform grid which defines the observation grid $\mathcal{X}$ via different subsampling rates. Yellow points correspond to the observation grid $\mathcal{X}$ (*In-s*) while purple points indicate off-grid points (*Out-s*). The prediction for I-MP-PDE at $t = 0$ is the interpolated initial condition.

E.3 Detailed description of datasets

We choose $\mathcal{T}$ (resp. $\mathcal{T}'$) on a regular grid in $[0, T]$ (resp. $[0, T']$) with a given temporal resolution and fix $T' = 2T$. We provide further details on the choice of these parameters and other experimental parameters, such as the number of observed trajectories.

Table E.1. – **Space and time extrapolation.** The train and test observation grids are equal; they are subsampled with a ratio s from an uniform $64{\times}64$ grid fixed here to be the inference grid $\mathcal{X}'$. We report MSE ($\downarrow$) on $\mathcal{X}'$ (on the observation grid *In-s*, outside *Out-s* or on both *Full*) and the inference time interval $\mathcal{T}'$, divided within training horizon (*In-t*, $\mathcal{T}$) and beyond (*Out-t*, outside $\mathcal{T}$) across subsampling ratios $s \in \{5\%, 25\%, 50\%, 100\%\}$. Best in **bold** and second best underlined.

	Model	Navier-Stokes Train In-t	Train Out-t	Test In-t	Test Out-t	Wave Train In-t	Train Out-t	Test In-t	Test Out-t
					$s = 5\%$ subsampling				
In-s	I-MP-PDE	**3.525e−5**	1.295E−3	4.554E−4	1.414E−3	**1.824e−6**	8.672E−5	1.113E−5	1.987E−4
	DeepONet	4.778E−4	4.517E−3	1.060E−2	1.059E−2	2.546E−4	8.831E−3	1.501E−2	3.196E−2
	SIREN	5.966E−3	1.769E−1	4.082E−2	2.150E−1	1.690E−3	1.707E−2	2.951E−2	6.955E−2
	DINo	1.016E−4	**6.945e−4**	**3.623e−4**	**8.306e−4**	2.250E−6	**5.283e−6**	**7.530e−6**	**2.146e−5**
Out-s	I-MP-PDE	8.550E−3	8.515E−3	8.306E−3	8.571E−3	7.412E−4	7.414E−4	1.195E−3	1.163E−3
	DeepONet	3.475E−3	7.515E−3	1.361E−2	1.426E−2	8.624E−4	9.318E−3	1.702E−2	3.259E−2
	SIREN	8.882E−3	1.767E−1	4.314E−2	2.124E−1	2.791E−3	1.823E−2	3.359E−2	6.965E−2
	DINo	**1.076e−3**	**1.704e−3**	**1.375e−3**	**1.863e−3**	**4.285e−5**	**4.304e−5**	**6.703e−5**	**7.659e−5**
Full	I-MP-PDE	8.154E−3	8.166E−3	7.926E−3	8.225E−3	7.055E−4	7.097E−4	1.138E−3	1.116E−3
	DeepONet	3.330E−3	7.370E−3	1.346E−2	1.408E−2	8.331E−4	9.295E−3	1.692E−2	3.256E−2
	SIREN	8.741E−3	1.767E−1	4.303E−2	2.126E−1	2.738E−3	1.818E−2	3.339E−2	6.964E−2
	DINo	**1.029e−3**	**1.655e−3**	**1.326e−3**	**1.813e−3**	**4.088e−5**	**4.121e−5**	**6.415e−5**	**7.392e−5**
					$s = 25\%$ subsampling				
In-s	I-MP-PDE	1.447E−4	5.677E−4	**1.763e−4**	6.147E−4	**6.754e−7**	8.251E−5	**9.253e−7**	1.227E−4
	DeepONet	7.500E−4	5.779E−3	9.227E−3	1.300E−2	5.196E−4	1.058E−2	1.743E−2	3.246E−2
	SIREN	4.786E−3	2.178E−1	2.461E−1	3.884E−1	8.478E−4	1.282E−2	1.733E−2	5.104E−2
	DINo	**8.295e−5**	**4.273e−4**	2.444E−4	**5.735e−4**	3.194E−6	**3.747e−6**	8.907E−6	**1.029e−5**
Out-s	I-MP-PDE	3.678E−4	7.748E−4	4.026E−4	8.143E−4	4.330E−5	1.200E−4	6.764E−5	1.648E−4
	DeepONet	9.503E−4	5.987E−3	9.423E−3	1.337E−2	5.891E−4	1.062E−2	1.762E−2	3.213E−2
	SIREN	5.305E−3	2.173E−1	2.428E−1	3.853E−1	9.159E−4	1.295E−2	1.798E−2	5.156E−2
	DINo	**1.081e−4**	**4.578e−4**	**2.711e−4**	**6.021e−4**	**4.192e−6**	**4.657e−6**	**1.153e−5**	**1.220e−5**
Full	I-MP-PDE	3.135E−4	7.245E−4	3.476E−4	7.658E−4	3.293E−5	1.108E−4	5.142E−5	1.545E−4
	DeepONet	9.016E−4	5.936E−3	9.376E−3	1.328E−2	5.722E−4	1.061E−2	1.757E−2	3.221E−2
	SIREN	5.180E−3	2.175E−1	2.436E−1	3.861E−1	8.995E−4	1.292E−2	1.783E−2	5.143E−2
	DINo	**1.020e−4**	**4.504e−4**	**2.646e−4**	**5.951e−4**	**3.949e−6**	**4.436e−6**	**1.089e−5**	**1.174e−5**
					$s = 50\%$ subsampling				
In-s	I-MP-PDE	1.153E−4	5.016E−4	**1.594e−4**	6.043E−4	**2.200e−7**	3.179E−5	**8.843e−7**	5.854E−5
	DeepONet	6.214E−4	4.277E−3	5.699E−3	1.082E−2	7.581E−4	1.187E−2	1.649E−2	3.378E−2
	SIREN	4.911E−3	6.815E−1	1.607E−1	6.889E−1	5.134E−4	1.481E−2	3.086E−2	8.196E−2
	DINo	**8.151e−5**	**2.920e−4**	2.004E−4	**4.283e−4**	3.277E−6	**3.659e−6**	8.978E−6	**9.572e−6**
Out-s	I-MP-PDE	1.186E−4	5.010E−4	**1.626e−4**	6.132E−4	**9.638e−7**	3.153E−5	**2.367e−6**	5.574E−5
	DeepONet	6.851E−4	4.343E−3	5.740E−3	1.099E−2	7.842E−4	1.185E−2	1.679E−2	3.391E−2
	SIREN	5.067E−3	6.867E−1	1.599E−1	6.845E−1	5.354E−4	1.492E−2	3.113E−2	8.333E−2
	DINo	**9.175e−5**	**3.041e−4**	2.116E−4	**4.409e−4**	3.277E−6	**3.659e−6**	8.978E−6	**9.572e−6**
Full	I-MP-PDE	1.170E−4	5.013E−4	**1.611e−4**	6.088E−4	**6.021e−7**	3.166E−5	**1.646e−6**	5.710E−5
	DeepONet	6.541E−4	4.311E−3	5.720E−3	1.091E−2	7.715E−4	1.186E−2	1.665E−2	3.385E−2
	SIREN	4.995E−3	6.841E−1	1.603E−1	6.867E−1	5.246E−4	1.486E−2	3.100E−2	8.265E−2
	DINo	**8.677e−5**	**2.982e−4**	2.062E−4	**4.348e−4**	3.380E−6	**3.751e−6**	9.251E−6	**9.710e−6**
					$s = 100\%$ subsampling				
Full	CNODE	2.319E−2	9.652E−2	2.305E−2	1.143E−1	2.337E−5	5.280E−4	3.057E−5	7.288E−4
	MP-PDE	1.140E−4	5.500E−4	**1.785e−4**	5.856E−4	**1.718e−7**	1.993E−5	**9.256e−7**	4.261E−5
	MNO	**3.190e−5**	8.678E−4	2.763E−4	8.946E−4	9.381E−6	4.890E−3	1.993E−4	6.128E−3
	DeepONet	1.375E−3	6.573E−3	9.704E−3	1.244E−2	6.431E−4	1.293E−2	1.847E−2	3.317E−2
	SIREN	1.066E−3	4.336E−1	3.874E−1	1.037E0	3.674E−4	9.956E−3	3.013E−2	7.842E−2
	MFN	1.651E−3	1.037E0	2.106E−1	1.059E0	1.408E−4	1.763E−1	4.735E−3	2.274E−1
	DINo (no sep.)	3.235E−4	1.593E−3	7.850E−4	1.889E−3	2.641E−6	4.081E−5	5.977E−5	2.979E−4
	DINo	8.339E−5	**3.115e−4**	2.092E−4	**4.311e−4**	3.309E−6	**3.506e−6**	9.495E−6	**9.946e−6**

Table E.2. – **Generalization across grids.** $\mathcal{X}_{tr}, \mathcal{X}_{ts}$ are subsampled with different ratios $s_{tr} \neq s_{ts} \in \{5, 50, 100\}\%$ from the same uniform $64{\times}64$ grid. We report *test* MSE within $\mathcal{X}_{ts}$ (*In-s*). **Best** in bold.

| Subsampling | Test → | $\mathcal{X}_{ts} = \mathcal{X}_{tr}$ | | $\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$ | | | | | |
| | | $s_{ts} = s_{tr}$ | | $s_{ts} = 5\%$ | | $s_{ts} = 50\%$ | | $s_{ts} = 100\%$ | |
Train ↓		In-t	Out-t	In-t	Out-t	In-t	Out-t	In-t	Out-t
$s_{tr} = 5\%$	MP-PDE	**1.967e−4**	**6.631e−4**	1.330E−1	3.852E−1	1.859E−1	6.680E−1	2.105E−1	7.120E−1
	DINo	3.623E−4	8.306E−4	**1.494e−3**	**2.291e−3**	**1.257e−3**	**1.883e−3**	**1.287e−3**	**1.947e−3**
$s_{tr} = 50\%$	MP-PDE	**1.346e−4**	5.110E−4	4.494e−2	9.403e−2	4.793E−3	1.997E−2	6.330E−3	3.712E−2
	DINo	2.004E−4	**4.283E−4**	**2.470e−4**	**4.697e−4**	**2.073e−4**	**4.284e−4**	**2.058e−4**	**4.361e−4**
$s_{tr} = 100\%$	MP-PDE	**1.785e−4**	5.856E−4	1.358E−1	3.355E−1	1.182E−2	2.664E−2	**1.785E−4**	**5.856E−4**
	DINo	2.092E−4	**4.311E−4**	**2.495e−4**	**4.805e−4**	**2.109e−4**	**4.325e−4**	2.092E−4	4.311e−4
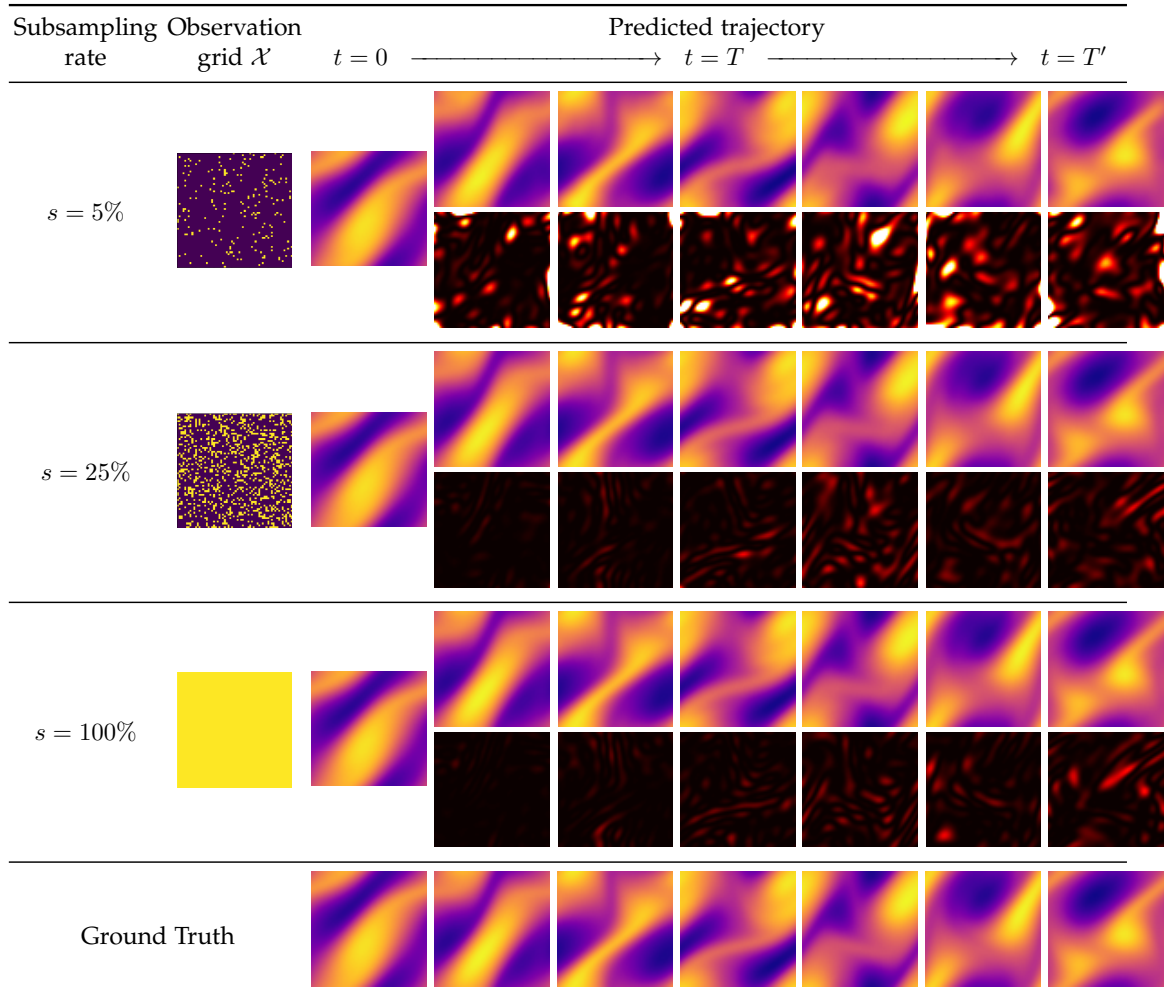
Figure E.1. – Prediction MSE per frame for **DINo** on *Navier-Stokes* with its corresponding observed train and test grid $\mathcal{X}$. For each model, the first row contains the predicted trajectory from 0 to T', the second row is the corresponding error maps w.r.t. the reference data (the darker the pixel, the lower the error).
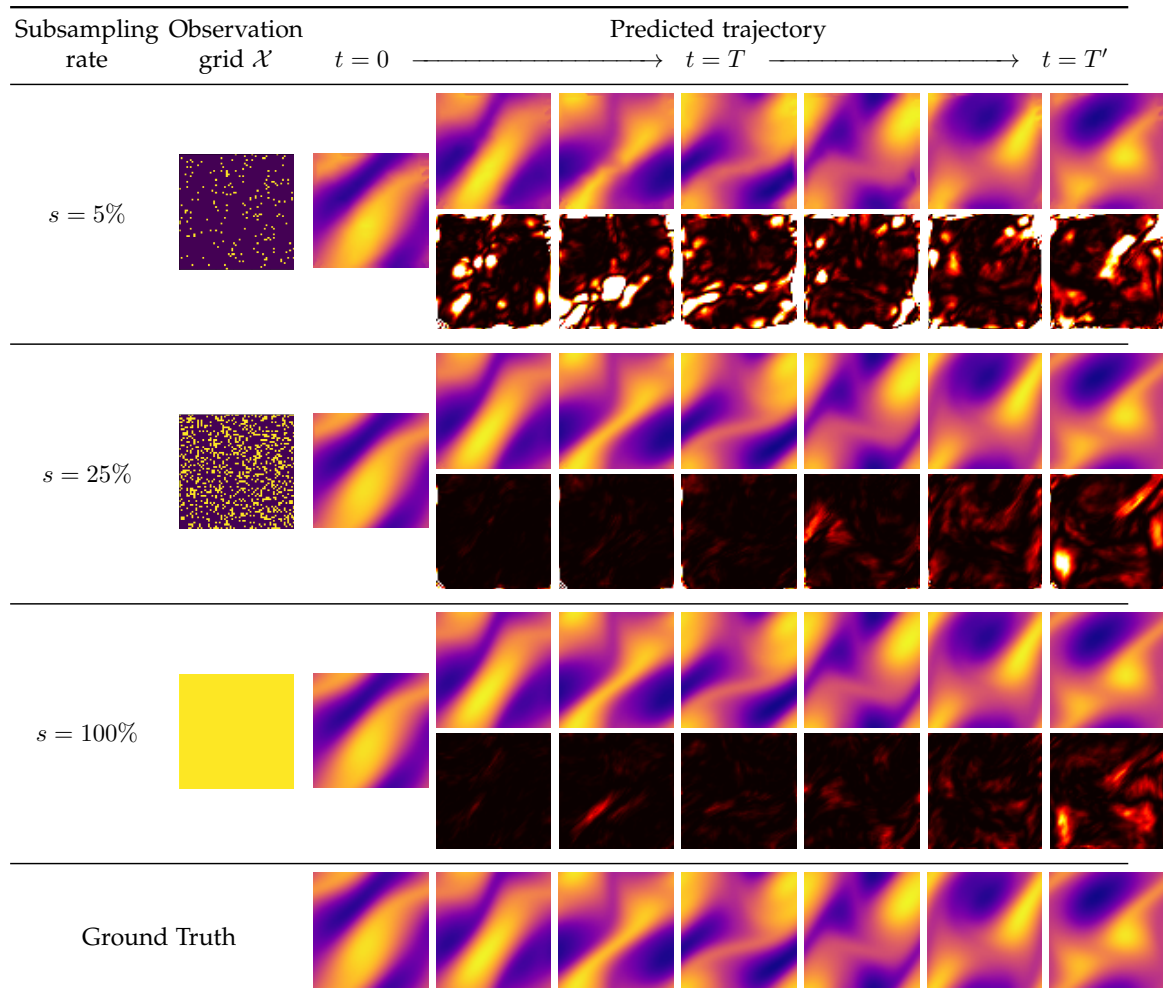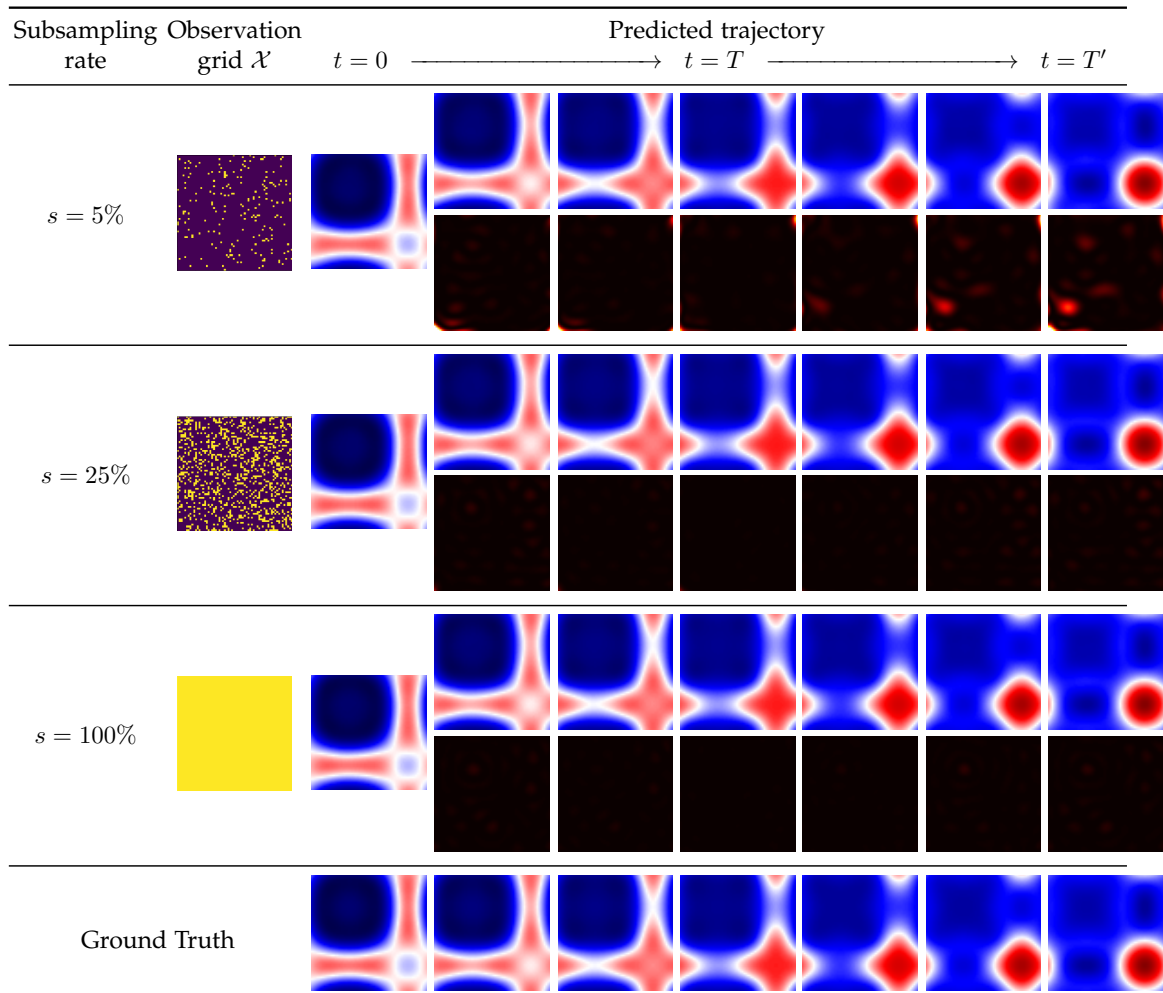
Figure E.2. – Prediction MSE per frame for **I-MP-PDE** on *Navier-Stokes* with its corresponding observed train and test grid $\mathcal{X}$. For each model, the first row contains the predicted trajectory from 0 to T', the second row is the corresponding error maps w.r.t. the reference data (the darker the pixel, the lower the error).

Subsampling rate	Observation grid $\mathcal{X}$	$t=0$			Predicted trajectory $t=T$			$t=T'$

Figure E.3. – Prediction MSE per frame for **DINo** on *Wave* with its corresponding observed train and test grid $\mathcal{X}$. For each model, the first row contains the predicted trajectory from 0 to T', the second row is the corresponding error maps w.r.t. the reference data (the darker the pixel, the lower the error).

2D Wave equation (*Wave*).   It is a second-order PDE:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \Delta u, \tag{E.1}$$

where u is a function of the displacement at each point in space w.r.t. the rest position, $c \in \mathbb{R}_+^*$ is the speed of wave traveling. We transform the equation to a first-order form, considering the input $v_t = (u_t, \frac{\mathrm{d}u_t}{\mathrm{d}t})$, so that the dimension of $v_t(x)$ at each point $x \in \Omega$ is $n = 2$.

We generate our dataset for speed $c = 2$ with periodic boundary condition. The domain is $\Omega = [-1, 1]^2$. For initial conditions $v_0 = (u_0, \frac{\mathrm{d}u_t}{\mathrm{d}t}_{|t=0})$, the initial displacement u_0 is a Gaussian function:

$$u_0(x; a, b, r) = a \exp\left(-\frac{(x - b)^2}{2r^2}\right), \tag{E.2}$$

where the height of the peak displacement is $a \sim \mathcal{U}(2, 4)$, the location of the peak displacement is $(b_1, b_2) \sim \mathcal{U}(-1, 1)$, and the standard deviation is $r \sim \mathcal{U}(0.25, 0.3)$. The initial time derivative is $\frac{\partial u_t}{\partial t}_{|t=0} = 0$. Each snapshot is generated on a uniform grid of 64×64. Each sequence is generated with fixed interval $\delta t = 0.25$. We set the training horizon $T = 2.25$ and the inference horizon $T = 4.75$. We generated 512 training trajectories and 32 test trajectories.

2D Navier Stokes (*Navier-Stokes*, Stokes 1851).   This dataset corresponds to an incompressible fluid dynamics described by:

$$\frac{\partial w}{\partial t} = -u\nabla w + \nu\Delta w + f, \quad w = \nabla \times u, \quad \nabla u = 0, \tag{E.3}$$

where u is the velocity field and w the vorticity. u, w lie on a spatial domain with periodic boundary conditions, ν is the viscosity and f is a constant forcing term. The input v_t is w_t ($n = 1$). ν is the viscosity and f is the constant forcing term in the domain Ω.

The spatial domain is $\Omega = [-1, 1]^2$, the viscosity is $\nu = 1 \times 10^{-3}$, the forcing term is set as:

$$\forall x \in \Omega, f(x_1, x_2) = 0.1(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2))). \tag{E.4}$$

The full spatial grid is of dimension 64×64 or 256×256 according to experiments in Section 7.4. We sample initial conditions as in Li et al. (2021c) to create different trajectories. The first 20 steps of the trajectories are cut off as they are too noisy and not informative in terms of dynamics. Trajectories are collected with $\delta t = 1$. We

set the training horizon $T = 19$ and the inference horizon $T' = 39$. We generated 512 training trajectories and 32 test trajectories.

3D spherical shallow water (*Shallow-Water*, Galewsky et al. 2004). The following problem is originally presented for testing numerical models of global shallow-water equations. The shallow water equations is written as:

$$
\begin{aligned}
\frac{\mathrm{d}u}{\mathrm{d}t} &= -fk \times u - g\nabla h_+ \nu\Delta u, \\
\frac{\mathrm{d}h}{\mathrm{d}t} &= -h_t\nabla \cdot u + \nu\Delta h.
\end{aligned}
\tag{E.5}
$$

where $\frac{\mathrm{d}}{\mathrm{d}t}$ is the material derivative, k is the unit vector orthogonal to the spherical surface, u is the velocity field tangent to the surface of the sphere, which can be transformed into the vorticity $w = \nabla \times u$, h is the thickness of the sphere. Note that the data we observe at each time t is $v_t = (w_t, h_t)$. f, g, ν, Ω are parameters of the Earth (*cf.* Galewsky et al. 2004 for details).

The initial conditions are slightly modified from Galewsky et al. 2004, detailed below, to create symmetric phenomena on the northern and southern hemisphere. The initial zonal velocity u_0 contains two non-null symmetric bands in the both hemispheres, which are parallel to the circles of latitude. At each latitude and longitude $\phi, \theta \in [-\pi/2, \pi/2] \times [-\pi, \pi]$:

$$
u_0(\phi, \theta) = \begin{cases}
(\frac{u_{\max}}{e_\mathrm{n}} \exp(\frac{1}{(\phi - \phi_0)(\phi - \phi_1)}), 0) & \text{if } \phi \in (\phi_0, \phi_1), \\
(\frac{u_{\max}}{e_\mathrm{n}} \exp(\frac{1}{(\phi + \phi_0)(\phi + \phi_1)}), 0) & \text{if } \phi \in (-\phi_1, -\phi_0), \\
(0, 0) & \text{otherwise.}
\end{cases}
\tag{E.6}
$$

where $u_{\max}$ is the maximum velocity, $\phi_0 = \pi/7$, $\phi_1 = \pi/2 - \phi_0$, and $e_\mathrm{n} = \exp(-4/(\phi_1 - \phi_0)^2)$. The water height h_0 is initialized by solving a boundary value condition problem as in Galewsky et al. (2004). It is then perturbed by adding the following h_0' to h_0:

$$
h_0'(\phi, \theta) = \hat{h} \cos(\phi) \exp(-(\frac{\theta}{\alpha})^2) \left[\exp(-(\frac{\phi_2 - \phi}{\beta})^2) + \exp(-(\frac{\phi_2 + \phi}{\beta})^2) \right].
\tag{E.7}
$$

where $\phi_2 = \pi/4$, $\hat{h} = 120\,\mathrm{m}$, $\alpha = 1/3$, $\beta = 1/15$ are constants defined in Galewsky et al. 2004.

We simulate this phenomenon with Dedalus (Burns et al. 2020) on a latitude-longitude (lat-lon) grid. The size of the grid is 128 (lat)×256 (lon). We take different initial conditions by sampling $u_{\max} \sim \mathcal{U}(60, 80)$ to generate long trajectories. These long trajectories are then sliced into shorter ones. For simulation, we take one snapshot per hour (of internal simulation time), *i.e.* $\delta t = 1\,\mathrm{h}$. We stop the

simulation at the 320th hour. To construct a dataset rich of dynamical phenomena, we take the snapshots within the last $160\,\text{h}$ in a long trajectory and slice them into 8 shorter trajectoires. Also note that the data is scaled into a reasonable range: the height h is scaled by a factor of 3×10^3, and the vorticity w by a factor 2. In each short trajectory, $T = 9\,\text{h}$ and $T' = 19\,\text{h}$. In total, we generated 16 long trajectories (*i.e.* 128 short trajectories) for train, 2 for test (*i.e.* 16 short trajectories).

E.4   Implementation

E.4.1   Algorithm

We detail the algorithm of DINo for training and test via pseudo-code in Algorithm E.1.

Algorithm E.1 DINo pseudo-code

1: *Training:* $\mathcal{D} = \{v_\mathcal{T}\}$, $\{\alpha_\mathcal{T}^v\}_{v \in \mathcal{D}} \leftarrow \{0\}$, $\phi \leftarrow \phi_0$, $\psi \leftarrow \psi_0$
2: **loop**
3:     **for** $v \in \mathcal{D}$ **do**
4:         $\alpha_\mathcal{T}^v \leftarrow \alpha_\mathcal{T}^v - \eta_\alpha \nabla_{\alpha_\mathcal{T}^v} \ell_{\text{dec}}(\phi, \alpha_\mathcal{T}^v)$ ▷ Modulation
5:     **end for**
6:     $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \big(\sum_{v \in \mathcal{D}} \ell_{\text{dec}}(\phi, \alpha_\mathcal{T}^v) \big)$ ▷ Hypernetwork
7:     $\psi \leftarrow \psi - \eta_\psi \nabla_\psi \big(\sum_{v \in \mathcal{D}} \ell_{\text{dyn}}(\psi, \alpha_\mathcal{T}^v) \big)$ ▷ Dynamics
8: **end loop**
9: *Test:* $\mathcal{D}_0' = \{v_0\}$, $\{\alpha_0^v\}_{v \in \mathcal{D}'} \leftarrow \{0\}$, $\phi^\star$, $\psi^\star$, $\mathcal{T}' \neq \mathcal{T}$
10: **loop**
11:     **for** $v \in \mathcal{D}'$ **do**
12:         $\alpha_0^v \leftarrow \alpha_0^v - \eta \nabla_{\alpha_0^v} \ell_{\text{dec}}(\phi^\star, \alpha_0^v)$ ▷ Modulation
13:     **end for**
14: **end loop**
15: **for** $v \in \mathcal{D}', t \in \mathcal{T}'$ **do**
16:     $\alpha_t^v \leftarrow \alpha_0^v + \int_0^t f_{\psi^\star}(\alpha_\tau^v) \mathrm{d}\tau$ ▷ Unroll dynamics
17: **end for**

E.4.2   Convergence

Convergence analysis.   In practice, we observe no training instability induced by the two-stage learning process of Eq. (7.6) and Algorithm E.1: the objectives are non-conflicting. To assess this, we track the evolution of the auto-decoding loss ℓ_{dec} and the dynamics loss ℓ_{dyn} throughout training on *Navier-Stokes* ($s = 100\%$)
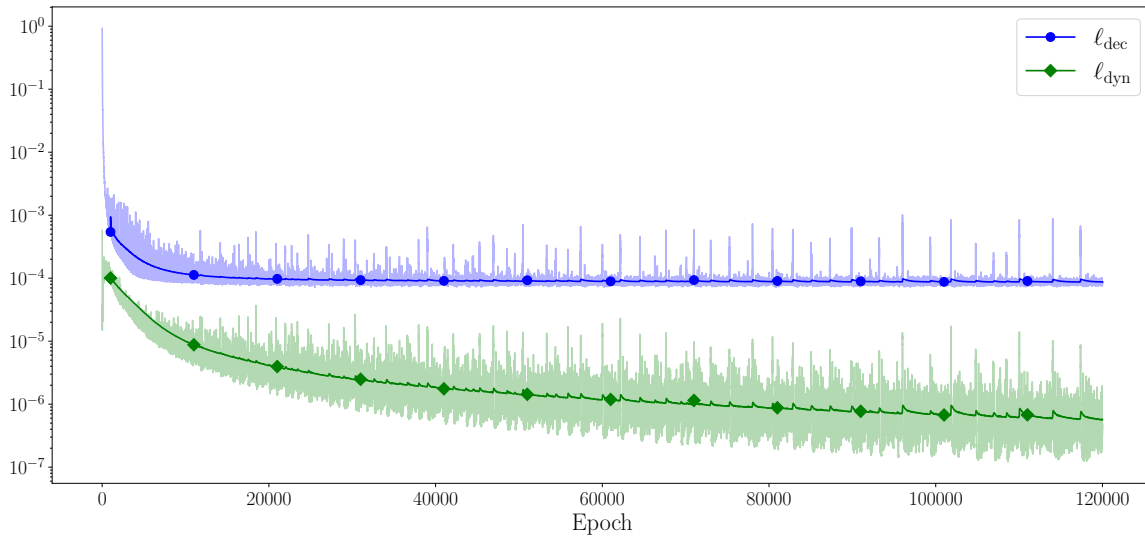
Figure E.4. – Learning curves on *Navier-Stokes* for ℓ_{dec} and ℓ_{dyn} throughout training (pale lines) and corresponding exponential moving averages from epoch 500 with half-life 1000 (opaque lines).

in Figure E.4. We observe that both losses smoothly converge until the end of training.

E.4.3 Time efficiency

Our auto-decoding strategy coupled with a latent neural ODE makes DINo computationally efficient compared to our best competitor MP-PDE.

Inferring α_0 via auto-decoding.   Given a decoder and an observation frame v_0, finding α_0 corresponds to solving an inverse problem, *cf.* Eq. (7.3). At inference, we use 300 steps to infer α_0; using less steps is possible but results in slight underfitting. This represents $2.76\,\text{s}$ for 64 trajectories on a single Tesla V100 Nvidia GPU. Note that, as we unroll dynamics in the latent space, there is no need to relearn α_t when $t > 0$. Moreover, this differs from training, where α_t is continuously optimized for all $t \in [0, T]$ within the train horizon, alternatively with our INR decoder. Overall, we trained MP-PDE and DINo for approximately 7 days such that there is no major additional temporal training cost for DINo.

Latent Neural ODE.   Unrolling the dynamics with a neural ODE is efficient ($0.35\,\text{s}$ for 19 time predictions for 64 trajectories on a single Tesla V100 Nvidia GPU). Indeed, the latent space is small (at most 100 dimension) and the dynamics models uses a simple four-layer MLP for f_ψ. With the same latent dynamics

model, using an RK4 numerical scheme only incurs four additional function evaluations over a discretized alternative *e.g.* standard ResNet. This incurs a minor computational cost but enables DINO to operate at different temporal resolutions, unlike *e.g.* MP-PDE.

In comparison, the official code of MP-PDE takes $312\,\mathrm{s}$ for inference on the same hardware for the same number of trajectories (vs $3\,\mathrm{s}$ for DINO). MP-PDE requires building an adjacency matrix and incurs for this reason a high memory cost, especially as the number of nodes increases. Interpolation also significantly increases inference time. This is not the case for DINO, which is faster.

E.4.4 Additional implementation details

We use PyTorch (Paszke et al. 2019) to implement DINO and our baselines. Hyperparameters are further defined in Appendix E.4.5. The dynamics model f_ψ is a multilayer perceptron. Its input and output size are same as the size of latent space d_α. All hidden layers share the same size. DINO's parameters are initialized with the default initialization in PyTorch, defining ϕ_0, ψ_0, ω in Algorithm E.1. We recall that ω is fixed throughout training to reduce the number of optimized parameters without loss of performance. As in related work (Sitzmann et al. 2020; Fathony et al. 2021), the frequency parameters ω are scaled by a factor, ω_s, considered as a hyperparameter. For dynamics learning, we use an RK4 integrator via TorchDiffEq (Chen et al. 2018) and apply exponential Scheduled Sampling (Bengio et al. 2015) to stabilize training. In practice, modulations α_t are learned channel-wise such that $I_\theta \colon \Omega \to \mathbb{R}^{d_c}$ has separate parameters per output dimension to make predictions less correlated across channels. We optimize all parameters ϕ, α, ψ using Adam (Kingma et al. 2015) with decay parameters $(\beta_1, \beta_2) = (0.9, 0.999)$.

E.4.5 Hyperparameters

We list the hyperparameters of DINO for each dataset in Table E.3. In practice, we observe it is beneficial to decay the learning rates η_ϕ, η_α when the loss reaches a plateau.

E.4.6 Baselines implementation

We detail in the following the hyperparameters and architectures used in our experiments for the considered baselines, which we reimplemented for our paper.

Table E.3. – DINo's hyperparameters.

Hyperparameter	Navier-Stokes	Wave	Shallow-Water
Decoder $g_\phi = I_{h_\phi}$			
Number of layers	3	3	6
Number hidden channels	64	64	256
Frequency scale factor ω_s	64	64	64
Size of latent space d_α	100	50	300
Dynamics model f_ψ			
Number of layers	4	4	4
Hidden layer size	512	512	800
Activation function	Swish	Swish	Swish
Optimization			
Learning rate η_ϕ	10^{-2}	10^{-2}	10^{-2}
Learning rate η_α	10^{-3}	10^{-3}	10^{-3}
Learning rate η_ψ	10^{-3}	10^{-3}	10^{-3}
Number of epochs	12 000	12 000	12 000
Batch size *i.e.* sequences per batch	64	64	16

- **CNODE** is implemented with four 2D convolutional layers with 64 hidden features, ReLU activations, 3×3 kernel and zero padding. Learning rate is fixed to 10^{-3}. We use an adjoint method for integration like (Chen et al. 2018).

- **MNO.** We use the FNO architecture in Li et al. 2021c with three FNO blocks, GeLU activations, 12 modes and a width of 32. Learning rate is fixed to 10^{-3}.

- **DeepONet.** We consider an autoregressive formulation of DeepONet. We choose a width of 1000 for hidden features with a depth of 4 for both trunk and branch nets with ReLU activations. Learning rate is fixed to 10^{-5}.

- **MP-PDE.** We adapt the implementation in Brandstetter et al. 2022 to handle 2D and 3D PDEs. We use a time window of 1 with pushforward trick. Batch size and number of neighbors are fixed to 8. Learning rate is fixed to 10^{-3}. We use ReLU activations.

- **SIREN.** To represent data in space and time, SIREN takes space and time coordinates (x, t) as input. To handle multiple trajectories, we concatenate an optimizable per-trajectory context code α to the coordinates like in DINo. We fix the hidden layer size of SIREN to 256. We initialize the parameters and use the default input scale as in Sitzmann et al. 2020. The size of the context code is $d_\alpha = 800$. The learning rate is 10^{-3}.

Table E.4. – Long term extrapolation performance of DINo and (I-)MP-PDE in the space and time generalization experiment for test trajectories on Out-t ($]T, T' = T + \Delta T]$); *cf.* Table 7.2 and Section 7.4.1.

Subsampling ratio	Model	$\Delta T = T$	$\Delta T = 5T$	$\Delta T = 10T$	$\Delta T = 50T$
$s = 5\%$	DINo	**2.017e−3**	**4.895e−3**	**1.209e−2**	**1.440e−1**
	I-MP-PDE	8.387E−3	3.580E−2	3.356E−1	4.031E1
$s = 100\%$	DINo	**4.617e−4**	**2.082e−3**	**6.901e−3**	**1.215e−1**
	MP-PDE	5.251E−4	3.524E−2	3.339E−1	9.755E1

- **MFN.** Similarly to the previous SIREN baseline, we concatenate the per-trajectory context code to space and time coordinates at the first layer. The hidden layer size is fixed to 256 and we use the default parameter initialization with a frequency scale ω_s of 64 higher than DINo. The size of the context code is $d_\alpha = 800$. The learning rate is 10^{-3}.

E.5   Complementary analyses

We detail in this section additional experiments, allowing us to further analyze and assess the performance of DINo.

E.5.1   Long-term temporal extrapolation

We provide in Table E.4 an analysis of error accumulation over time for long-term extrapolation. More precisely, we generate a Navier-Stokes dataset with longer trajectories and report MSE for $T' = T + \Delta T$ where $\Delta T \in \{T, 5T, 10T, 50T\}$. Note that $\Delta T = T$ is the setting in our initial submission ($T' = 2T$).

We observe that DINo's MSE in long-term forecasting is more than an order of magnitude smaller than for (I-)MP-PDE. This demonstrates the extrapolation abilities of our model.

E.5.2   INRs' advantage over interpolation

We report in Table E.5 the MSE of bicubic interpolation, our FourierNet's MSE (auto-decoding with amplitude modulation but without dynamics model) and DINo's MSE (with dynamics model) on train In-t for both *Navier-Stokes* and *Wave*.

Table E.5. – MSE reconstruction error (In-s and Out-s) of train sequences within the train horizon (In-t) for three different methods: interpolation of observed points in $\mathcal{X}_{\mathrm{tr}}$, FourierNet learned over individual frames in $\mathcal{X}_{\mathrm{tr}}$, and DINo (FourierNet with a dynamics model).

MSE train In-t	Interpolation	FourierNet	DINo
Navier-Stokes, $s = 5\%$	8.277E−3	**9.673e−4**	1.029E−3
Wave, $s = 5\%$	7.075E−4	**4.085e−5**	4.088E−5

This corresponds to MSE averaged over all training frames within the train horizon and not only the initial condition v_0.

We observe that FourierNet is better than interpolation. Indeed, interpolation is poorly adapted to sparse observation grids: the interpolation errors are clearly visible in Figure E.2, first row (5% setting). Interestingly, DINo's MSE is only slightly worse than the FourierNet's MSE, showing that we correctly learned the dynamics of latent modulations α_t. I-MP-PDE, which combines bicubic interpolation with MP-PDE, is then expectedly outperformed by DINo on this challenging 5% setting. This shows the advantage of using INRs instead of standard bicubic interpolation to interpolate between observed spatial locations.

E.5.3   Modeling real-world data

SST.   We evaluate DINo on real-world data to further assess its applicability. Following Bézenac et al. 2018a and Donà et al. 2021, we model the Sea Surface Temperature (SST) of the Atlantic ocean, derived from the data-assimilation engine NEMO (Nucleus for European Modeling of the Ocean, Madec et al. n.d.) using E.U. Copernicus Marine Service Information[1]. Accurately modeling SST dynamics is critical in weather forecasting or planning of coastal activities. This problem is particularly challenging as SST dynamics are only partially observed: several unobserved variables affecting the dynamics (*e.g.* the sea water flow) need to be estimated from data.

For this experiment, we consider trajectories collected from three geographical zones (17 to 20) following the initial train / test split of Bézenac et al. 2018a. Notably, $T = 9\,\mathrm{d}$, which includes $\tau = 4\,\mathrm{d}$ of conditioning frames, *i.e.* models are tested to predict $v_{t\in[\![\tau,T]\!]}$ from $v_{t\in[\![0,\tau]\!]}$.

---

1. https://data.marine.copernicus.eu/product/GLOBAL_ANALYSIS_FORECAST_PHY_001_024/description

Table E.6. – SST test prediction performance for DINo and VarSep.

Method	MSE
VarSep (Donà et al. 2021)	1.43
DINo	**1.27**

Incorporating consecutive time steps.   To model SST which includes non-Markovian data and thus does not correspond to an Initial Value Problem as in Section 7.2, we modify our dynamics model in a similar fashion to Yıldız et al. 2019 to integrate a history of several consecutive observations $v_{t \in [\![0, \tau]\!]}$ instead of only the initial observation v_0. In more details, we define a neural ODE over an augmented state $[\alpha_t, \alpha_t']$ where α_t is our auto-decoded state and α_t' is an encoding of $\tau = 4$ past auto-decoded observations via a neural network c_ξ. We adjust our inference and training settings as follows:

- inference: we compute $\alpha_{\tau-1}' = c_\xi(\alpha_0, \cdots, \alpha_{\tau-1})$ and then unroll our neural ODE from the initial condition $[\alpha_{\tau-1}, \alpha_{\tau-1}']$ to obtain $[\alpha_t, \alpha_t']$ for all $t > \tau - 1$:

$$\forall t \in [\![0, \tau - 1]\!], \alpha_t = e_\varphi(v_t), \quad \alpha_{\tau-1}' = c_\xi(\alpha_0, \cdots, \alpha_{\tau-1}), \quad \frac{\mathrm{d}[\alpha_t, \alpha_t']}{\mathrm{d}t} = f_\psi([\alpha_t, \alpha_t']);$$

- training: for all t, we infer $\alpha_{t+\tau-1}' = c_\xi(\alpha_t, \cdots, \alpha_{t+\tau-1})$ and fit the above neural ODE on the $[\alpha_t, \alpha_t']$ obtained for all $t \in [\![0, T - \tau + 1]\!]$.

This experiment confirms that our space- and time-continuous framework can easily be extended to incorporate refined temporal models.

Results.   We report in Table E.6 test MSE for DINo and VarSep (Donà et al. 2021), the current state-of-the-art on SST, retrained on the same training data. DINo notably outperforms VarSep in prediction performance. This demonstrates DINo's potential to handle complex real-world spatiotemporal dynamics. We also provide some visualizations of DINo's train and test predictions in Figure E.5. We make two observations. First, DINo fits very accurately the train data. Second, on the test, we observe that the dynamics on low frequencies seem to be correctly modeled while the prediction of high frequencies dynamics are less accurate. Larger scale experiments would be required to effectively evaluate the model performance on this challenging dataset. Given the complexity of the data, this is out of the scope of the paper. Yet, these experiments already demonstrate that DINo behaves competitively w.r.t the previous state-of-the-art.
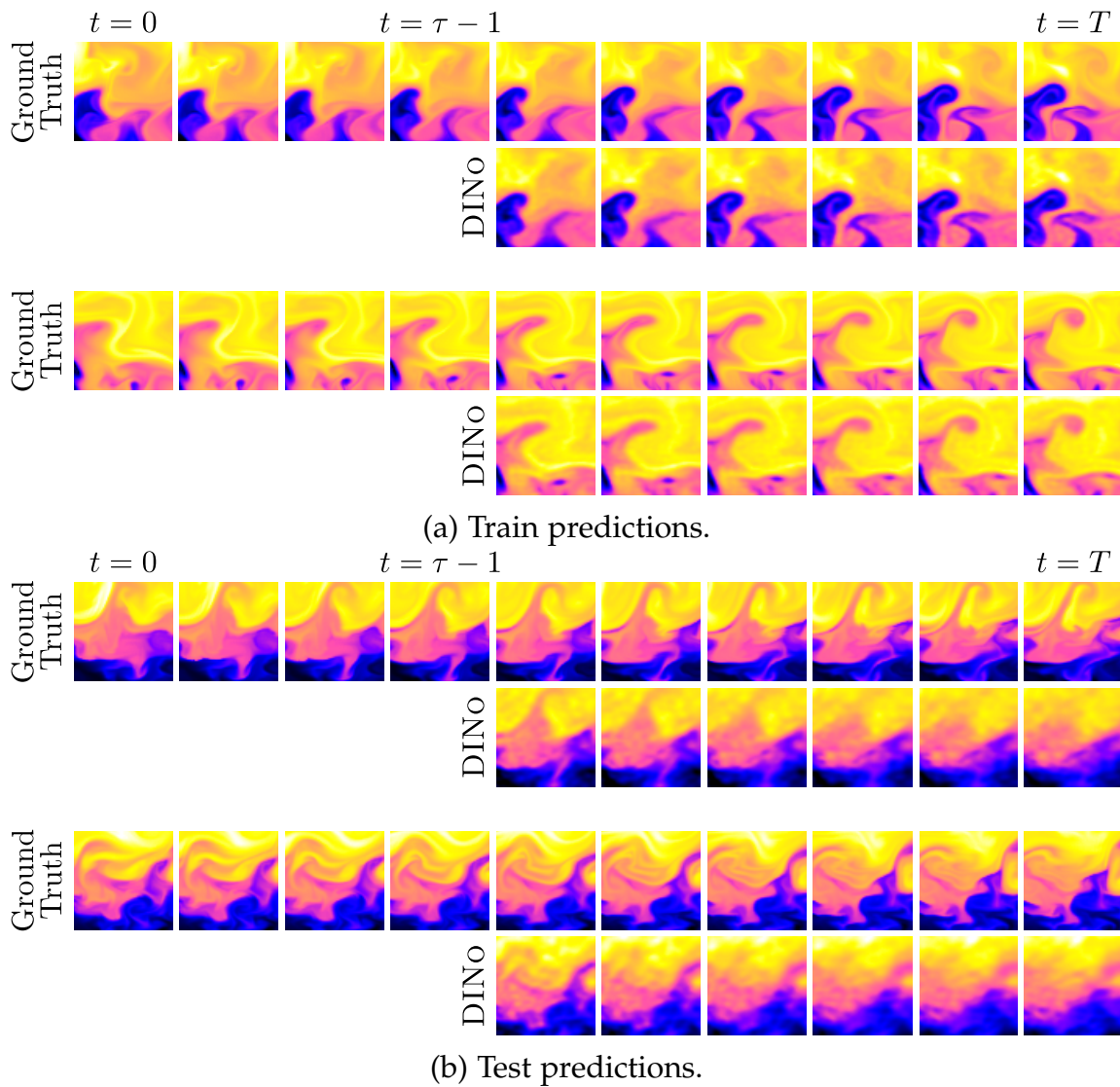
(a) Train predictions.

(b) Test predictions.

Figure E.5. – DINo's prediction examples on SST.

Implementation choices.   We choose a similar INR and dynamics architecture than for our Shallow-water experiment. We use for c_ξ, which takes as input four consecutive α_ts, individual encodings of the α_ts through a four-layer fully connected network which are then fed to a single linear layer.

Bibliography

Agarap, Abien Fred (2018). "Deep learning using rectified linear units (relu)". In: *arXiv preprint* (cit. on p. 78).

Aggarwal, Karan, Matthieu Kirchmeyer, Pranjul Yadav, S. Sathiya Keerthi, and Patrick Gallinari (2019). "Regression with Conditional GAN". In: *CoRR* abs/1905.12868. arXiv: 1905.12868. URL: http://arxiv.org/abs/1905.12868 (cit. on pp. 35, 137).

Aksela, Matti (2003). "Comparison of classifier selection methods for improving committee performance". In: *MCS* (cit. on pp. 84, 87, 177, 190, 191, 193, 194).

Amini, M-R and P Gallinari (Nov. 2005). "Semi-supervised Learning with an Imperfect Supervisor". In: *Knowl. Inf. Syst.* 8, pp. 385–413 (cit. on pp. 12, 47).

Antoniou, Antreas, Harrison Edwards, and Amos J. Storkey (2019). "How to train your MAML". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=HJGven05Y7 (cit. on p. 106).

Argyriou, Andreas, Theodoros Evgeniou, and Massimiliano Pontil (2006). "Multi-Task Feature Learning". In: *Advances in Neural Information Processing Systems*. Ed. by B. Schölkopf, J. Platt, and T. Hoffman. Vol. 19. MIT Press. URL: https://proceedings.neurips.cc/paper/2006/file/0afa92fc0f8a9cf051bf2961b06ac56b-Paper.pdf (cit. on p. 13).

Arjovsky, Martin, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz (2019a). "Invariant Risk Minimization". In: *arXiv preprint* (cit. on pp. 78, 82, 199, 200).

Arjovsky, Martin, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz (2019b). "Invariant Risk Minimization". In: *ArXiv* arxiv:1907.02893 (cit. on pp. 17, 115).

Arpit, Devansh, Huan Wang, Yingbo Zhou, and Caiming Xiong (2021). "Ensemble of Averages: Improving Model Selection and Boosting Performance in Domain Generalization". In: *arXiv preprint* (cit. on pp. 18, 80, 84, 88, 89, 137, 175, 178, 182, 190, 197, 198).

Ashukha, Arsenii, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov (2020). "Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning". In: *ICLR* (cit. on pp. 18, 78).

Avila Belbute-Peres, Filipe de, Thomas D. Economon, and J. Zico Kolter (2020). "Combining Differentiable PDE Solvers and Graph Neural Networks for Fluid Flow Prediction". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Ma-

chine Learning Research. PMLR, pp. 2402–2411. URL: http://proceedings.mlr.press/v119/de-avila-belbute-peres20a.html (cit. on p. 3).

Ayed, Ibrahim, Emmanuel de Bézenac, Arthur Pajot, and P. Gallinari (May 2020). "Learning the Spatio-Temporal Dynamics of Physical Processes from Partial Observations". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3232–3236 (cit. on pp. 118, 126).

Barjasteh, Iman, R. Forsati, Farzan Masrour, A. Esfahanian, and H. Radha (2015). "Cold-Start Item and User Recommendation with Decoupled Completion and Transduction". In: *Proceedings of the 9th ACM Conference on Recommender Systems*, pp. 91–98 (cit. on p. 37).

Beery, Sara, Grant van Horn, and Pietro Perona (2018a). *Recognition in Terra Incognita*. URL: https://arxiv.org/abs/1807.04975 (cit. on p. 4).

Beery, Sara, Grant Van Horn, and Pietro Perona (2018b). "Recognition in terra incognita". In: *ECCV* (cit. on pp. 88, 198).

Ben-David, Shai, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan (2010). "A theory of learning from different domains". In: *Machine Learning* 79.1, pp. 151–175 (cit. on p. 142).

Bengio, Samy, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer (2015). "Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett. Vol. 28. Curran Associates, Inc., pp. 1171–1179 (cit. on pp. 107, 228).

Benton, Gregory, Wesley Maddox, Sanae Lotfi, and Andrew Gordon Wilson (2021). "Loss Surface Simplexes for Mode Connecting Volumes and Fast Ensembling". In: *ICML* (cit. on p. 18).

Berger, Marsha J. and Joseph Oliger (Mar. 1984). "Adaptive mesh refinement for hyperbolic partial differential equations". In: *Journal of Computational Physics* 53.3, pp. 484–512 (cit. on p. 118).

Bertinetto, Luca, Joao F. Henriques, Philip Torr, and Andrea Vedaldi (2019). "Meta-learning with differentiable closed-form solvers". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=HyxnZh0ct7 (cit. on p. 205).

Bézenac, Emmanuel de, Ibrahim Ayed, and P. Gallinari (2021). "CycleGAN through the lens of (Dynamical) Optimal Transport". In: *ECML-PKDD* (cit. on pp. 62, 66).

Bézenac, Emmanuel de, Arthur Pajot, and P. Gallinari (2018a). "Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge". In: *International Conference on Learning Representations* (cit. on pp. 24, 231).

Blitzer, John, Ryan McDonald, and Fernando Pereira (2006). "Domain Adaptation with Structural Correspondence Learning". In: *Proceedings of the 2006 Confer-*

ence on Empirical Methods in Natural Language Processing, pp. 120–128 (cit. on p. 52).

Bora, Ashish, Eric Price, and Alexandros G. Dimakis (2018). "AmbientGAN: Generative models from lossy measurements". In: *International Conference on Learning Representations* (cit. on p. 36).

Brain, Damien and Geoffrey I Webb (1999). "On the effect of data set size on bias and variance in classification learning". In: *AKAW* (cit. on p. 185).

Brandstetter, Johannes, Daniel E. Worrall, and Max Welling (2022). "Message Passing Neural PDE Solvers". In: *International Conference on Learning Representations* (cit. on pp. 24, 26, 118, 119, 126, 229).

Breiman, Leo (1996). "Bagging predictors". In: *Machine learning* (cit. on pp. 18, 192).

Brenier, Y. (1991). "Polar factorization and monotone rearrangement of vector-valued functions". In: (cit. on p. 164).

Brodersen, K. H., C. S. Ong, K. E. Stephan, and J. M. Buhmann (2010). "The Balanced Accuracy and Its Posterior Distribution". In: *20th ICPR*, pp. 3121–3124 (cit. on p. 68).

Brown, Gavin, Jeremy Wyatt, and Ping Sun (2005). "Between two extremes: Examining decompositions of the ensemble objective function". In: *MCS* (cit. on pp. 81, 179).

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020). "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf (cit. on p. 3).

Brunton, Steven L. and J. Nathan Kutz (2022). *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press (cit. on p. 118).

Burns, Keaton J., Geoffrey M. Vasil, Jeffrey S. Oishi, Daniel Lecoanet, and Benjamin P. Brown (Apr. 2020). "Dedalus: A flexible framework for numerical simulations with spectral methods". In: *Physical Review Research* 2.2 (cit. on p. 225).

Cai, L., Z. Wang, H. Gao, D. Shen, and S. Ji (2018). "Deep adversarial learning for multi-modality missing data completion". In: *Proceedings of the 24th*

ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1158–1166 (cit. on pp. 34, 36).

Carreira-Perpiñán, Miguel Á. and Weiran Wang (2014). "LASS: A Simple Assignment Model with Laplacian Smoothing". In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. Ed. by Carla E. Brodley and Peter Stone. AAAI Press, pp. 1715–1721. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8226 (cit. on p. 164).

Caruana, Rich (1997). "Multitask Learning". In: *Machine Learning* 28.1, pp. 41–75 (cit. on pp. 13, 205).

Cha, Junbum, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park (2021). "SWAD: Domain Generalization by Seeking Flat Minima". In: *NeurIPS* (cit. on pp. 18, 78, 80, 84, 85, 88, 173, 175, 176, 178, 182, 196–198).

Chae, Sangwon, Joonhyeok Shin, Sungjun Kwon, Sangmok Lee, Sungwon Kang, and Donghyun Lee (2021). "PM10 and PM2.5 real-time prediction models using an interpolated convolutional neural network". In: *Scientific Reports* 11 (cit. on p. 24).

Chapelle, Olivier, Bernhard Schlkopf, and Alexander Zien (2010). *Semi-Supervised Learning*. 1st. The MIT Press (cit. on pp. 13, 62).

Chen, C., Q. Dou, H. Chen, J. Qin, and P-A Heng (2019a). "Synergistic Image and Feature Adaptation: Towards Cross-Modality Domain Adaptation for Medical Image Segmentation". In: *Proceedings of the 33rd Conference on Artificial Intelligence (AAAI)*, pp. 865–872 (cit. on p. 35).

Chen, Hao, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava (2021). "NeRV: Neural Representations for Videos". In: *Advances in Neural Information Processing Systems*. Ed. by Marc'Aurelio Ranzato, Alina Beygelzimer, Yann Dauphin, Percy Liang, and Jenn Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 21557–21568 (cit. on p. 25).

Chen, Minmin, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha (2012). "Marginalized Denoising Autoencoders for Domain Adaptation". In: *Proceedings of the 29th International Conference on International Conference on Machine Learning*. Edinburgh, Scotland, pp. 1627–1634 (cit. on p. 145).

Chen, Ricky T. Q. (2021). *torchdiffeq*. Version 0.2.2. URL: https://github.com/rtqichen/torchdiffeq (cit. on p. 107).

Chen, Ricky T. Q., Yulia Rubanova, Jesse Bettencourt, and David Duvenaud (2018). "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems*. Ed. by Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Curran Associates, Inc., pp. 6572–6583 (cit. on pp. 21, 22, 119, 228, 229).

Chen, Xinyang, Sinan Wang, Mingsheng Long, and Jianmin Wang (Sept. 2019b). "Transferability vs. Discriminability: Batch Spectral Penalization for Adversarial Domain Adaptation". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 1081–1090. URL: https://proceedings.mlr.press/v97/chen19i.html (cit. on pp. 15, 58, 71).

Chen, Yutian, Abram L Friesen, Feryal Behbahani, Arnaud Doucet, David Budden, Matthew Hoffman, and Nando de Freitas (2020a). "Modular Meta-Learning with Shrinkage". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 2858–2869 (cit. on p. 112).

Chen, Zhengdao, Jianyu Zhang, Martin Arjovsky, and Léon Bottou (2020b). "Symplectic Recurrent Neural Networks". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=BkgYPREtPr (cit. on p. 98).

Choshen, Leshem, Elad Venezian, Noam Slonim, and Yoav Katz (2022). "Fusing finetuned models for better pretraining". In: *arXiv preprint* (cit. on p. 18).

Clavera, Ignasi, Anusha Nagabandi, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn (2019). "Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=HyztsoC5Y7 (cit. on p. 99).

Combes, Remi Tachet des, Han Zhao, Yu-Xiang Wang, and Geoff Gordon (2020). "Domain Adaptation with Conditional Distribution Matching and Generalized Label Shift". In: *Advances in Neural Information Processing Systems* (cit. on pp. 58, 62–65, 68, 70, 71, 171, 172).

Courtier, Philippe, J-N Thépaut, and Anthony Hollingsworth (1994). "A strategy for operational implementation of 4D-Var, using an incremental approach". In: *Quarterly Journal of the Royal Meteorological Society* 120.519, pp. 1367–1387 (cit. on p. 98).

Courty, N., Rémi Flamary, Amaury Habrard, and A. Rakotomamonjy (2017a). "Joint distribution optimal transportation for domain adaptation". In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 3730–3739 (cit. on pp. 15, 141).

Courty, N., Rémi Flamary, Devis Tuia, and A. Rakotomamonjy (2017b). "Optimal Transport for Domain Adaptation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.9, pp. 1853–1865 (cit. on pp. 15, 62, 71).

Covington, Paul, Jay Adams, and Emre Sargin (2016). "Deep Neural Networks for YouTube Recommendations". In: *Proceedings of the 10th ACM Conference on Recommender Systems*. New York, NY, USA (cit. on p. 3).

Crammer, Koby, Michael Kearns, and Jennifer Wortman (2008). "Learning from Multiple Sources". In: *Journal of Machine Learning Research*. Vol. 9, pp. 1757–1774 (cit. on p. 144).

D'Amour, Alexander, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. (2020). "Underspecification presents challenges for credibility in modern machine learning". In: *JMLR* (cit. on pp. 78, 83).

Damodaran, Bharath Bhushan, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and N. Courty (2018). "DeepJDOT : Deep Joint Distribution Optimal Transport for Unsupervised Domain Adaptation". In: *European Conference in Computer Visions*, pp. 467–483 (cit. on pp. 15, 40, 141, 144, 146, 148, 150).

Daniels, Bryan C. and Ilya Nemenman (Mar. 2015). "Efficient Inference of Parsimonious Phenomenological Models of Cellular Dynamics Using S-Systems and Alternating Regression". In: *PLOS ONE* 10.3, pp. 1–14 (cit. on pp. 108, 214).

De Avila Belbute-Peres, Filipe, Thomas Economon, and Zico Kolter (July 2020). "Combining Differentiable PDE Solvers and Graph Neural Networks for Fluid Flow Prediction". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 2402–2411 (cit. on p. 118).

de Bézenac, Emmanuel, Arthur Pajot, and P. Gallinari (2018b). "Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=By4HsfWAZ (cit. on p. 98).

DeGrave, Alex J, Joseph D Janizek, and Su-In Lee (2021). "AI for radiographic COVID-19 detection selects shortcuts over signal". In: *Nature Machine Intelligence* (cit. on pp. 4, 78).

Denton, Emily and Rob Fergus (Oct. 2018). "Stochastic Video Generation with a Learned Prior". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1174–1183. URL: https://proceedings.mlr.press/v80/denton18a.html (cit. on p. 3).

Dietterich, Thomas G (2000). "Ensemble methods in machine learning". In: *MCS* (cit. on p. 81).

Ding, Zhengming, Ming Shao, and Yun Fu (2014). "Latent low-rank transfer subspace learning for missing modality recognition". In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pp. 1192–1198 (cit. on p. 37).

Dinh, Laurent, Razvan Pascanu, Samy Bengio, and Yoshua Bengio (2017). "Sharp Minima Can Generalize For Deep Nets". In: *ICML* (cit. on p. 174).

Doinychko, A and M-R Amini (2020). "Biconditional GANs for Multiview Learning with Missing Views". In: *Advances in Information Retrieval*, pp. 807–820 (cit. on pp. 34, 36).

Domingos, Pedro (2000). "A unified bias-variance decomposition". In: *ICML* (cit. on p. 80).

Donà, Jérémie, Jean-Yves Franceschi, Sylvain Lamprier, and P. Gallinari (2021). "PDE-Driven Spatiotemporal Disentanglement". In: *International Conference on Learning Representations* (cit. on pp. 231, 232).

Draxler, Felix, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht (2018). "Essentially No Barriers in Neural Network Energy Landscape". In: *ICML* (cit. on p. 18).

Dupont, Emilien, Hyunjik Kim, S. M. Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum (July 2022). "From data to functa: Your data point is a function and you can treat it like one". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 5694–5725 (cit. on p. 125).

Duraisamy, Karthik, Gianluca Iaccarino, and Heng Xiao (2019). "Turbulence modeling in the age of data". In: *Annual Review of Fluid Mechanics* 51, pp. 357–377 (cit. on p. 98).

Efron, Bradley (1992). "Bootstrap methods: another look at the jackknife". In: *Breakthroughs in statistics* (cit. on p. 192).

Fallah, Alireza, Aryan Mokhtari, and Asuman Ozdaglar (2020). "Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 3557–3568. URL: https://proceedings.neurips.cc/paper/2020/file/24389bfe4fe2eba8bf9aa9203a44cdad-Paper.pdf (cit. on p. 4).

Fang, Chen, Ye Xu, and Daniel N Rockmore (2013). "Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias". In: *ICCV* (cit. on pp. 88, 198).

Fathony, Rizal, Anit Kumar Sahu, Devin Willmott, and J. Zico Kolter (2021). "Multiplicative Filter Networks". In: *International Conference on Learning Representations* (cit. on pp. 25, 124–126, 228).

Ferradans, Sira, N. Papadakis, Julien Rabin, Gabriel Peyré, and Jean-François Aujol (June 2013). "Regularized Discrete Optimal Transport". In: *SSVM 2013 - International Conference on Scale Space and Variational Methods in Computer Vision*. Ed. by Arjan Kuijper, Kristian Bredies, Thomas Pock, and Horst Bischof. Vol. 7893. Lecture Notes in Computer Science. Schloss Seggau, Leibnitz, Aus-

tria: Springer, pp. 428–439. URL: https://hal.archives-ouvertes.fr/hal-00797078 (cit. on p. 164).

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (June 2017). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 1126–1135. URL: http://proceedings.mlr.press/v70/finn17a.html (cit. on pp. 15, 99, 110, 205).

Flennerhag, Sebastian, Andrei A. Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell (2020). "Meta-Learning with Warped Gradient Descent". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rkeiQlBFPB (cit. on p. 16).

Foret, Pierre, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur (2021). "Sharpness-aware Minimization for Efficiently Improving Generalization". In: *ICLR* (cit. on pp. 80, 175, 176).

Fort, Stanislav, Huiyi Hu, and Balaji Lakshminarayanan (2019). "Deep ensembles: A loss landscape perspective". In: *arXiv preprint* (cit. on pp. 19, 78).

Fraccaro, Marco (2018). "Deep Latent Variable Models for Sequential Data". PhD thesis. Danmarks Tekniske Universitet (cit. on p. 123).

Franceschi, Jean-Yves, Aymeric Dieuleveut, and Martin Jaggi (2019). "Unsupervised Scalable Representation Learning for Multivariate Time Series". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2019/file/53c6de78244e9f528eb3e1cda69699bb-Paper.pdf (cit. on p. 3).

Frankle, Jonathan and Michael Carbin (2019). "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rJl-b3RcF7 (cit. on p. 106).

Frankle, Jonathan, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin (2020). "Linear Mode Connectivity and the Lottery Ticket Hypothesis". In: *ICML* (cit. on pp. 18, 78, 84, 85).

Fresca, Stefania, Andrea Manzoni, Luca Dedè, and Alfio Quarteroni (Oct. 2020). "Deep learning-based reduced order models in cardiac electrophysiology". In: *PloS one* 15.10, e0239416–e0239416. URL: https://pubmed.ncbi.nlm.nih.gov/33002014 (cit. on pp. 25, 98).

Galewsky, Joseph, Richard K. Scott, and Lorenzo M. Polvani (2004). "An initial-value problem for testing numerical models of the global shallow-water equations". In: *Tellus A: Dynamic Meteorology and Oceanography* 56.5, pp. 429–440 (cit. on pp. 126, 225).

Ganin, Yaroslav and Victor Lempitsky (2015). "Unsupervised Domain Adaptation by Backpropagation". In: *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1180–1189 (cit. on pp. 14, 40, 43, 49, 146, 149, 150).

Ganin, Yaroslav, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky (2016a). "Domain-Adversarial Training of Neural Networks". In: *Journal of Machine Learning Research* 17.59, pp. 1–35 (cit. on pp. 14, 68).

Ganin, Yaroslav, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky (2016b). "Domain-adversarial training of neural networks". In: *JMLR* (cit. on p. 17).

Garg, Saurabh, Yifan Wu, Sivaraman Balakrishnan, and Zachary Lipton (2020). "A Unified View of Label Shift Estimation". In: *NeurIPS*. Vol. 33. Curran Associates, Inc., pp. 3290–3300 (cit. on pp. 63, 163).

Garnelo, Marta, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Jimenez Rezende, and S. M. Ali Eslami (2018). "Conditional Neural Processes". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1690–1699. URL: http://proceedings.mlr.press/v80/garnelo18a.html (cit. on pp. 16, 106, 115).

Ghojogh, Benyamin, Ali Ghodsi, Fakhri Karray, and Mark Crowley (2021). "Reproducing Kernel Hilbert Space, Mercer's Theorem, Eigenfunctions, Nystrom Method, and Use of Kernels in Machine Learning: Tutorial and Survey". In: *arXiv preprint* (cit. on pp. 83, 185).

Gholaminejad, Amir, Kurt Keutzer, and George Biros (July 2019). "ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, pp. 730–736. URL: https://doi.org/10.24963/ijcai.2019/103 (cit. on p. 22).

Gong, Mingming, Kun Zhang, Tongliang Liu, Dacheng Tao, Clark Glymour, and Bernhard Schölkopf (20–22 Jun 2016). "Domain Adaptation with Conditional Transferable Components". In: *Proceedings of The 33rd ICML*. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 2839–2848 (cit. on pp. 58, 62, 63, 70, 71, 164).

Gontijo-Lopes, Raphael, Yann Dauphin, and Ekin Dogus Cubuk (2022). "No One Representation to Rule Them All: Overlapping Features of Training Methods". In: *ICLR* (cit. on pp. 78, 85, 192, 193).

Grandvalet, Yves and Yoshua Bengio (2005). "Semi-supervised Learning by Entropy Minimization". In: *Proceedings of the 17th International Conference on Neural Information Processing Systems*, pp. 529–536 (cit. on pp. 12, 47).

Gretton, Arthur, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola (2012). "A Kernel Two-Sample Test". In: *Journal of Machine Learning Research* 13.25, pp. 723–773. URL: http://jmlr.org/papers/v13/gretton12a.html (cit. on p. 187).

Greydanus, Samuel, Misko Dzamba, and Jason Yosinski (2019). "Hamiltonian Neural Networks". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 15353–15363. URL: https://proceedings.neurips.cc/paper/2019/file/26cd8ecadce0d4efd6cc8a8725cbd1f8-Paper.pdf (cit. on p. 98).

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville (2017). "Improved Training of Wasserstein GANs". In: *NeurIPS*. Curran Associates, pp. 5767–5777 (cit. on pp. 168, 169).

Gulrajani, Ishaan and David Lopez-Paz (2021a). "In Search of Lost Domain Generalization". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=lQdXeXDoWtI (cit. on pp. 6, 11, 136, 137).

Gulrajani, Ishaan and David Lopez-Paz (2021b). "In Search of Lost Domain Generalization". In: *ICLR* (cit. on pp. 18, 78–80, 85, 86, 88, 174, 188, 195–199).

Guo, Hao, Jiyong Jin, and Bin Liu (2022). "Stochastic Weight Averaging Revisited". In: *arXiv preprint* (cit. on p. 18).

Gupta, Jayesh K and Johannes Brandstetter (2023). *Towards Multi-spatiotemporal-scale Generalized PDE Modeling*. URL: https://openreview.net/forum?id=Uk40pC45YJG (cit. on pp. 24, 26).

Gupta, Vipul, Santiago Akle Serrano, and Dennis DeCoste (2020). "Stochastic Weight Averaging in Parallel: Large-Batch Training That Generalizes Well". In: *ICLR* (cit. on p. 18).

Gur-Ari, Guy, Daniel A. Roberts, and Ethan Dyer (2019). *Gradient Descent Happens in a Tiny Subspace*. URL: https://openreview.net/forum?id=ByeTHsAqtX (cit. on p. 106).

Ha, David, Andrew M. Dai, and Quoc V. Le (2017). "HyperNetworks". In: *International Conference on Learning Representations* (cit. on pp. 103, 123).

Hairer, E., S. P. Nørsett, and G. Wanner (2000). *Solving Ordinary Differential Equations I: Nonstiff problems*. Second. Berlin: Springer (cit. on pp. 3, 20, 107).

Hansen, Lars Kai and Peter Salamon (1990). "Neural network ensembles". In: *IEEE transactions on pattern analysis and machine intelligence* (cit. on p. 18).

He, Hangfeng and Weijie Su (2020). "The Local Elasticity of Neural Networks". In: *ICLR* (cit. on pp. 83, 185).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (June 2016a). "Deep Residual Learning for Image Recognition". In: pp. 770–778 (cit. on p. 9).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016b). "Deep residual learning for image recognition". In: *CVPR* (cit. on pp. 88, 197).

Hendrycks, Dan and Thomas Dietterich (2019). "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations". In: *ICLR* (cit. on p. 78).

Hoffman, Judy, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell (Oct. 2018). "CyCADA: Cycle-Consistent Adversarial Domain Adaptation". In: *Proceedings of the 35th ICML*. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, pp. 1989–1998 (cit. on p. 15).

Hull, J. J. (May 1994). "A Database for Handwritten Text Recognition Research". In: *IEEE TPAMI* 16.5, pp. 550–554 (cit. on pp. 49, 67).

Iakovlev, Valerii, Markus Heinonen, and Harri Lähdesmäki (2021). "Learning continuous-time PDEs from sparse data with graph neural networks". In: *International Conference on Learning Representations* (cit. on p. 24).

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *ICML* (cit. on p. 78).

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei Efros (2017). "Image-to-Image Translation with Conditional Adversarial Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5967–5976 (cit. on p. 36).

Izmailov, Pavel, Wesley Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson (2019). "Subspace Inference for Bayesian Deep Learning". In: *UAI* (cit. on p. 18).

Izmailov, Pavel, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson (2018). "Averaging weights leads to wider optima and better generalization". In: *UAI* (cit. on pp. 18, 78, 80, 81, 177).

Jacot, Arthur, Franck Gabriel, and Clement Hongler (2018). "Neural Tangent Kernel: Convergence and Generalization in Neural Networks". In: *NeurIPS* (cit. on pp. 83, 185, 188).

Johansson, Fredrik D., David Sontag, and Rajesh Ranganath (2019). "Support and Invertibility in Domain-Invariant Representations". In: *Proceedings of the 32th International Conference on Artificial Intelligence and Statistics*, pp. 527–536 (cit. on p. 46).

Kaddour, Jean, Linqing Liu, Ricardo Silva, and Matt J. Kusner (2022). "Questions for Flat-Minima Optimization of Modern Neural Networks". In: *arXiv preprint* (cit. on pp. 80, 176).

Kalman, R. E. (Mar. 1960). "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1, pp. 35–45. eprint: https://asmedigitalcollection.asme.org/fluidsengineering/article-pdf/82/1/35/5518977/35_1.pdf. URL: https://doi.org/10.1115/1.3662552 (cit. on p. 98).

Karkar, Skander, Ibrahim Ayed, Emmanuel de Bezenac, and P. Gallinari (Sept. 2020). "A Principle of Least Action for the Training of Neural Networks". In: *ECML PKDD*. Ghent, Belgium (cit. on pp. 62, 66).

Kim, Hyunjik, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh (2019a). "Attentive Neural Processes". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=SkE6PjC9KX (cit. on p. 106).

Kim, Hyunjik, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh (2019b). "Attentive Neural Processes". In: *International Conference on Learning Representations* (cit. on p. 123).

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (cit. on pp. 109, 197, 215, 228).

Kirchmeyer, M., P. Gallinari, A. Rakotomamonjy, and A. Mantrach (2021). "Unsupervised domain adaptation with non-stochastic missing data". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD) & Data Mining and Knowledge Discovery (DMKD)* 35.6, pp. 2714–2755 (cit. on pp. 4, 5, 33).

Kirchmeyer, M., A. Rakotomamonjy, Emmanuel de Bezenac, and P. gallinari (2022a). "Mapping conditional distributions for domain adaptation under generalized target shift". In: *ICLR* (cit. on p. 131).

Kirchmeyer, M., A. Rakotomamonjy, E. de Bézenac, and P. Gallinari (2022b). "Mapping conditional distributions for domain adaptation under generalized target shift". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 6, 57).

Kirchmeyer*, M., Y. Yin*, J. Dona, N. Baskiotis, A. Rakotomamonjy, and P. Gallinari (17–23 Jul 2022). "Generalizing to New Physical Systems via Context-Informed Dynamics Model". In: *Proceedings of the 39th International Conference on Machine Learning (ICML)*. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 11283–11301 (cit. on pp. 7, 97).

Kirichenko, Polina, Pavel Izmailov, and Andrew Gordon Wilson (2022). "Last layer re-training is sufficient for robustness to spurious correlations". In: *ICML SCIS Workshop* (cit. on pp. 173, 200, 201, 203).

Kirkpatrick, James, Brendan McMorrow, David H. P. Turban, Alexander L. Gaunt, James S. Spencer, Alexander G. D. G. Matthews, Annette Obika, Louis Thiry, Meire Fortunato, David Pfau, Lara Román Castellanos, Stig Petersen, Alexan-

der W. R. Nelson, Pushmeet Kohli, Paula Mori-Sánchez, Demis Hassabis, and Aron J. Cohen (2021). "Pushing the frontiers of density functionals by solving the fractional electron problem". In: *Science* 374.6573, pp. 1385–1389. URL: https://www.science.org/doi/abs/10.1126/science.abj6511 (cit. on p. 3).

Kochkov, Dmitrii, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer (2021). "Machine learning accelerated computational fluid dynamics". In: *Proceedings of the National Academy of Sciences* 118 (21). URL: https://www.pnas.org/content/118/21/e2101784118 (cit. on pp. 3, 98, 118).

Koh, Pang Wei, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang (2021). "WILDS: A Benchmark of in-the-Wild Distribution Shifts". In: *ICML* (cit. on p. 18).

Kohavi, Ron et al. (1996). "Bias plus variance decomposition for zero-one loss functions". In: *ICML* (cit. on pp. 80, 179, 189).

Kornblith, Simon, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton (2019). "Similarity of Neural Network Representations Revisited". In: *ICML* (cit. on pp. 84, 87, 191, 193).

Kovachki, Nikola, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar (2021). "Neural operator: Learning maps between function spaces". In: *arXiv preprint arXiv:2108.08481* (cit. on p. 25).

Krishnapriyan, Aditi, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney (2021). "Characterizing possible failure modes in physics-informed neural networks". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 26548–26560. URL: https://proceedings.neurips.cc/paper/2021/file/df438e5206f31600e6ae4af72f2725f1-Paper.pdf (cit. on p. 24).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *NeurIPS* (cit. on pp. 9, 85, 188).

Krogh, Anders and Jesper Vedelsby (1995). "Neural network ensembles, cross validation, and active learning". In: *NeurIPS* (cit. on p. 18).

Krueger, David, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Rémi LePriol, Dinghuai Zhang, and Aaron Courville (2021). "Out-of-Distribution Generalization via Risk Extrapolation (REx)". In: *International Conference on Machine Learning* (cit. on pp. 17, 78).

Kumar, Ananya, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang (2022). "Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution". In: *ICLR* (cit. on pp. 85, 89, 188, 196, 197).

Kuncheva, Ludmila I and Christopher J Whitaker (2003). "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy". In: *Machine learning* (cit. on p. 84).

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell (2017). "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *NeurIPS* (cit. on pp. 18, 78, 81, 84, 89, 175, 189, 190).

Lam, Remi, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Alexander Pritzel, Suman Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Jacklynn Stott, Oriol Vinyals, Shakir Mohamed, and Peter Battaglia (2022). *GraphCast: Learning skillful medium-range global weather forecasting* (cit. on p. 138).

Le Dret, Hervé and Brigitte Lucquin (2016). "Partial Differential Equations: Modeling, Analysis and Numerical Approximation". In: International Series of Numerical Mathematics. Cham, Switzerland: Springer International Publishing. Chap. The Heat Equation, pp. 219–251 (cit. on p. 125).

LeCun, Yann, Leon Bottou, Yoshua Bengio, and P Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on pp. 49, 67).

LeCun, Yann, Corinna Cortes, and Chris Burges (2010). *MNIST handwritten digit database* (cit. on p. 199).

Lee, Jaehoon, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein (2017). "Deep neural networks as gaussian processes". In: *ICLR* (cit. on pp. 83, 185).

Lee, Kimin, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin (13–18 Jul 2020). "Context-aware Dynamics Model for Generalization in Model-Based Reinforcement Learning". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 5757–5766. URL: http://proceedings.mlr.press/v119/lee20g.html (cit. on p. 99).

Lee, Kwonjoon, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto (2019). "Meta-Learning With Differentiable Convex Optimization". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, pp. 10657–10665. URL: http://openaccess.thecvf.com/content%5C_CVPR%5C_2019/html/Lee%5C_Meta-Learning%5C_With%5C_Differentiable%5C_Convex%5C_Optimization%5C_CVPR%5C_2019%5C_paper.html (cit. on p. 205).

Lee, Seungjun, Haesang Yang, and Woojae Seong (2021). "Identifying Physical Law of Hamiltonian Systems via Meta-Learning". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=45NZvF1UHam (cit. on p. 27).

Lee, Yoonho and Seungjin Choi (2018). "Gradient-based meta-learning with learned layerwise metric and subspace". In: *International Conference on Machine Learning*, pp. 2933–2942 (cit. on p. 16).

Lee, Yoonho, Huaxiu Yao, and Chelsea Finn (2022). "Diversify and Disambiguate: Learning From Underspecified Data". In: *arXiv preprint* (cit. on p. 18).

Li, ChunY., Heerad Farkhoor, Rosanne Liu, and Jason Yosinski (2018a). "Measuring the Intrinsic Dimension of Objective Landscapes". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=ryup8-WCW (cit. on p. 106).

Li, Da, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales (2017a). "Deeper, broader and artier domain generalization". In: *ICCV* (cit. on pp. 86, 88, 198).

Li, Hao, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein (2018b). "Visualizing the Loss Landscape of Neural Nets". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. (cit. on pp. 104, 106).

Li, S.C, Jiang B., and B Marlin (2019). "MisGAN: Learning From Incomplete Data with generative Adversarial Networks". In: *International Conference on Learning Representations* (cit. on p. 36).

Li, Shuang, Chi Harold Liu, Limin Su, Binhui Xie, Zhengming Ding, C. L. Philip Chen, and Dapeng Wu (2020a). "Discriminative Transfer Feature and Label Consistency for Cross-Domain Image Classification". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.11, pp. 4842–4856 (cit. on p. 58).

Li, Xiang Lisa and Percy Liang (2021a). "Prefix-Tuning: Optimizing Continuous Prompts for Generation". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* abs/2101.00190 (cit. on p. 4).

Li, Zhenguo, Fengwei Zhou, Fei Chen, and Hang Li (2017b). "Meta-SGD: Learning to Learn Quickly for Few Shot Learning". In: *CoRR* abs/1707.09835. arXiv: 1707.09835. URL: http://arxiv.org/abs/1707.09835 (cit. on pp. 16, 110).

Li, Ziyue, Kan Ren, Xinyang Jiang, Bo Li, Haipeng Zhang, and Dongsheng Li (2022). "Domain Generalization using Pretrained Models without Fine-tuning". In: *arXiv preprint* (cit. on p. 18).

Li, Zongyi, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar (2020b). "Multipole Graph Neural Operator for Parametric Partial Differential Equations". In:

Advances in Neural Information Processing Systems. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 6755–6766 (cit. on pp. 25, 118).

Li, Zongyi, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar (2021b). "Markov neural operators for learning chaotic systems". In: *arXiv preprint arXiv:2106.06898* (cit. on pp. 25, 126).

Li, Zongyi, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar (2021c). "Fourier Neural Operator for Parametric Partial Differential Equations". In: *International Conference on Learning Representations* (cit. on pp. 25, 98, 109, 118, 119, 126, 215, 224, 229).

Liang, Jian, Dapeng Hu, and Jiashi Feng (13–18 Jul 2020). "Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 6028–6039. URL: http://proceedings.mlr.press/v119/liang20a.html (cit. on p. 67).

Lipton, Z., Yu-Xiang Wang, and A. Smola (2018). "Detecting and correcting for label shift with black box predictors". In: *Proceedings of the 35th International Conference on Machine Learning*, pp. 3122–3130 (cit. on pp. 65, 68, 70, 168, 170, 172).

Little, R.J.A. and D.B. Rubin (1986). *Statistical analysis with missing data*. John Wiley & Sons, Inc. (cit. on pp. 34, 36, 39).

Liu, Hong, Mingsheng Long, Jianmin Wang, and Michael Jordan (Sept. 2019). "Transferable Adversarial Training: A General Approach to Adapting Deep Classifiers". In: *Proceedings of 36th ICML*. Vol. 97. Proceedings of ML Research. PMLR, pp. 4013–4022 (cit. on pp. 15, 58, 71).

Long, M., Z. Cao, J. Wang, and M. I. Jordan (2018a). "Conditional Adversarial Domain Adaptation". In: *Advances in Neural Information Processing Systems*. Vol. 31 (cit. on p. 14).

Long, Mingsheng, Yue Cao, Jianmin Wang, and Michael Jordan (2015). "Learning Transferable Features with Deep Adaptation Networks". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning*. Vol. 37, pp. 97–105 (cit. on pp. 14, 15).

Long, Zichao, Yiping Lu, Xianzhong Ma, and Bin Dong (July 2018b). "PDE-Net: Learning PDEs from Data". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm, Sweden: PMLR, pp. 3208–3216 (cit. on pp. 24, 106).

Lotka, A.J. (1925). "Elements of Physical Biology". In: *Nature* 116.2917, pp. 461–461 (cit. on pp. 108, 213).

Lu, Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis (2021). "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nature Machine Intelligence* 3, pp. 218–229 (cit. on pp. 25, 118, 119, 126).

Maddox, Wesley J, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson (2019). "A simple baseline for bayesian uncertainty in deep learning". In: *NeurIPS* (cit. on pp. 18, 19).

Madec, Gurvan and NEMO System Team (n.d.). *NEMO ocean engine*. Tech. rep. 27. Scientific Notes of Climate Modelling Center, Institut Pierre-Simon Laplace (IPSL). Zenodo (cit. on p. 231).

Magnus, Jan R and Heinz Neudecker (2019). *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons (cit. on p. 187).

Matena, Michael and Colin Raffel (2021). "Merging Models with Fisher-Weighted Averaging". In: *arXiv preprint* (cit. on pp. 18, 78).

Mathematical Theory of Optimal Processes (1962) (cit. on p. 22).

Mattei, Pierre-Alexandre and Jes Frellsen (2019). "MIWAE: Deep Generative Modelling and Imputation of Incomplete Data". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97, pp. 4413–4423 (cit. on p. 36).

Mesbah, Yusuf, Youssef Youssry Ibrahim, and Adil Mehood Khan (2022). "Domain Generalization Using Ensemble Learning". In: *ISWA* (cit. on p. 18).

Michaelis, Claudio, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S. Ecker, Matthias Bethge, and Wieland Brendel (2019). *Benchmarking Robustness in Object Detection: Autonomous Driving when Winter is Coming* (cit. on p. 4).

Mildenhall, Ben, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng (2020). "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *The European Conference on Computer Vision (ECCV)*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Cham, Switzerland: Springer International Publishing, pp. 405–421 (cit. on p. 118).

Mishra, Nikhil, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel (2018). "A Simple Neural Attentive Meta-Learner". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=B1DmUzWAW (cit. on pp. 99, 112).

Morton, K. W. and D. F. Mayers (2005). *Numerical Solution of Partial Differential Equations: An Introduction*. 2nd ed. Cambridge University Press (cit. on p. 23).

Muandet, Krikamol, David Balduzzi, and Bernhard Schölkopf (2013). "Domain generalization via invariant feature representation". In: *ICML* (cit. on p. 78).

Nagarajan, Vaishnavh and J Zico Kolter (2019). "Uniform convergence may be unable to explain generalization in deep learning". In: *NeurIPS* (cit. on p. 18).

Neic, Aurel, Fernando Otaviano Campos, Anton J. Prassl, Steven A. Niederer, Martin J. Bishop, Edward J. Vigmond, and Gernot Plank (2017). "Efficient computation of electrograms and ECGs in human whole heart simulations using a reaction-eikonal model". In: *J. Comput. Phys.* 346, pp. 191–211. URL: https://doi.org/10.1016/j.jcp.2017.06.020 (cit. on p. 98).

Netzer, Yuval, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng (2011). "NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011". In: *Proceedings of the IEEE* (cit. on p. 49).

Neyshabur, Behnam, Hanie Sedghi, and ChiY. Zhang (2020). "What is being transferred in transfer learning?" In: *NeurIPS* (cit. on pp. 78, 85, 191, 193).

Nixon, Jeremy, Balaji Lakshminarayanan, and Dustin Tran (2020). "Why Are Bootstrapped Deep Ensembles Not Better?" In: *NeurIPS Workshop* (cit. on p. 18).

Olver, Peter J. (2014). *Introduction to partial differential equations*. Undergraduate Texts in Mathematics. Springer Cham (cit. on p. 118).

Ovadia, Yaniv, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek (2019). "Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift". In: *NeurIPS* (cit. on pp. 18, 78).

Pagliardini, Matteo, Martin Jaggi, François Fleuret, and Sai Praneeth Karimireddy (2022). "Agree to Disagree: Diversity through Disagreement for Better Transferability". In: *arXiv preprint* (cit. on p. 18).

Pajot, A., E. de Bezenac, and P. Gallinari (2019). "Unsupervised Adversarial Image Reconstruction". In: *International Conference on Learning Representations* (cit. on p. 36).

Pan, S. J. and Q. Yang (Oct. 2010). "A Survey on Transfer Learning". In: *TKDE* (cit. on pp. 14, 35, 57).

Pan, Shaowu, Steven L Brunton, and J Nathan Kutz (2022). "Neural Implicit Flow: a mesh-agnostic dimensionality reduction paradigm of spatio-temporal data". In: *arXiv preprint arXiv:2204.03216* (cit. on p. 26).

Pang, Tianyu, Kun Xu, Chao Du, Ning Chen, and Jun Zhu (2019). "Improving Adversarial Robustness via Promoting Ensemble Diversity". In: *ICML* (cit. on p. 18).

Park, Jeong Joon, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove (2019). "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, pp. 165–174. URL: http://openaccess.thecvf.com/content%5C_CVPR%5C_2019/html/Park%5C_DeepSDF%5C_Learning%5C_

`Continuous%5C_Signed%5C_Distance%5C_Functions%5C_for%5C_Shape%5C_` `Representation%5C_CVPR%5C_2019%5C_paper.html` (cit. on pp. 16, 122).

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 8026–8037 (cit. on p. 228).

Pathak, D., P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros (2016). "Context Encoders: Feature Learning by Inpainting". In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536–2544 (cit. on pp. 36, 151).

Pathak, Jaideep, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar (2022). *FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators*. URL: `https://arxiv.org/abs/2202.11214` (cit. on pp. 3, 138).

Pearson, John E. (1993). "Complex Patterns in a Simple System". In: *Science* 261.5118, pp. 189–192. eprint: `https://www.science.org/doi/pdf/10.1126/science.261.5118.189`. URL: `https://www.science.org/doi/abs/10.1126/science.261.5118.189` (cit. on pp. 109, 214).

Peng, Xingchao, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang (2019). "Moment matching for multi-source domain adaptation". In: *ICCV* (cit. on pp. 88, 198, 199).

Peng, Xingchao, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko (2017). "Visda: The visual domain adaptation challenge". In: *arXiv preprint arXiv:1710.06924* (cit. on p. 67).

Perez, Ethan, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville (Apr. 2018). "FiLM: Visual Reasoning with a General Conditioning Layer". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1, pp. 3942–3951 (cit. on pp. 110, 125, 207).

Pérez-Cruz, Fernando, Steven Van Vaerenbergh, Juan José Murillo-Fuentes, Miguel Lázaro-Gredilla, and Ignacio Santamaria (2013). "Gaussian processes for nonlinear signal processing: An overview of recent advances". In: *EEE Signal Process. Mag.* (cit. on p. 184).

Peters, Jonas, Peter Bühlmann, and Nicolai Meinshausen (2016). "Causal inference by using invariant prediction: identification and confidence intervals". In: *JSTOR* (cit. on p. 78).

Petzka, Henning, Michael Kamp, Linara Adilova, Cristian Sminchisescu, and Mario Boley (2021). "Relative Flatness and Generalization". In: *NeurIPS* (cit. on p. 174).

Peyré, Gabriel and Marco Cuturi (2019). "Computational Optimal Transport". In: *Foundations and Trends in Machine Learning* 11 (5-6), pp. 355–602 (cit. on pp. 40, 141, 163).

Pfaff, Tobias, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia (2021). "Learning Mesh-Based Simulation with Graph Networks". In: *International Conference on Learning Representations* (cit. on pp. 3, 24, 118).

Ah-Pine, Julien (2010). "Normalized kernels as similarity indices". In: *PAKDD* (cit. on pp. 83, 185).

Prasthofer, Michael, Tim De Ryck, and Siddhartha Mishra (2022). "Variable-Input Deep Operator Networks". In: *arXiv preprint arXiv:2205.11404* (cit. on p. 118).

Pustelnik, N., C. Chaux, and J. Pesquet (2011). "Parallel Proximal Algorithm for Image Restoration Using Hybrid Regularization". In: *IEEE TIP* 20.9, pp. 2450–2462 (cit. on p. 172).

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. (2021). "Learning transferable visual models from natural language supervision". In: *ICML* (cit. on pp. 9, 137, 188).

Raghu, Aniruddh, Maithra Raghu, Samy Bengio, and Oriol Vinyals (2020). "Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rkgMkCEtPB (cit. on pp. 16, 112, 205).

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis (2019). "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378, pp. 686–707 (cit. on pp. 24, 25, 105, 118).

Rakotomamonjy, A., R. Flamary, G. Gasso, M. El Alaya, M. Berar, and N. Courty (2021). "Optimal transport for conditional domain matching and label shift". In: *Machine Learning* (cit. on pp. 58, 62, 63, 65, 68, 70, 71, 167, 171, 172).

Ramachandran, Prajit, Barret Zoph, and Quoc V. Le (2018). *Searching for Activation Functions*. URL: https://openreview.net/forum?id=SkBYYyZRZ (cit. on p. 215).

Rame, Alexandre and M. Cord (2021a). "DICE: Diversity in Deep Ensembles via Conditional Redundancy Adversarial Estimation". In: *ICLR* (cit. on pp. 18, 191).

Rame, Alexandre, Corentin Dancette, and M. Cord (2022). "Fishr: Invariant Gradient Variances for Out-of-Distribution Generalization". In: *ICML* (cit. on pp. 78, 200).

Rame, Alexandre, Remy Sun, and M. Cord (2021b). "MixMo: Mixing Multiple Inputs for Multiple Outputs via Deep Subnetworks". In: *ICCV* (cit. on pp. 18, 191).

Rame*, A., M. Kirchmeyer*, T. Rahier, A. Rakotomamonjy, P. Gallinari, and M. Cord (2022). "Diverse Weight Averaging for Out-of-Distribution Generalization". In: *Neural Information Processing Systems (NeurIPS)* (cit. on pp. 7, 77).

Ramesh, Aditya, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen (2022). "Hierarchical text-conditional image generation with clip latents". In: *arXiv preprint* (cit. on pp. 3, 176).

Rasmussen, Carl Edward (2003). "Gaussian processes in machine learning". In: *Summer school on machine learning* (cit. on p. 184).

Rebuffi, Sylvestre-Alvise, Hakan Bilen, and Andrea Vedaldi (2017). "Learning multiple visual domains with residual adapters". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc. (cit. on p. 13).

Rebuffi, Sylvestre-Alvise, Hakan Bilen, and Andrea Vedaldi (2018). "Efficient Parametrization of Multi-Domain Deep Neural Networks". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, pp. 8119–8127 (cit. on p. 13).

Redko, Ievgen, N. Courty, Rémi Flamary, and Devis Tuia (16–18 Apr 2019). "Optimal Transport for Multi-source Domain Adaptation under Target Shift". In: vol. 89. Proceedings of Machine Learning Research. PMLR, pp. 849–858 (cit. on pp. 63, 64, 70, 163).

Reichstein, Markus, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, and Prabhat (2019). "Deep learning and process understanding for data-driven Earth system science". In: *Nature* 566.7743, pp. 195–204 (cit. on p. 98).

Rennie, Jason (2005). "How to normalize a kernel matrix". In: *MIT Computer Science - Artificial Intelligence Lab Tech Rep* (cit. on pp. 83, 185).

Requeima, James, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E. Turner (2019). "Fast and Flexible Multi-Task Classification using Conditional Neural Adaptive Processes". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 7957–7968 (cit. on pp. 13, 115).

Roberts, Michael, Derek Driggs, Matthew Thorpe, Julian Gilbey, Michael Yeung, Stephan Ursprung, Angelica I Aviles-Rivero, Christian Etmann, Cathal McCague, Lucian Beer, et al. (2021). "Common pitfalls and recommendations for using

machine learning to detect and prognosticate for COVID-19 using chest radiographs and CT scans". In: *Nature Machine Intelligence* (cit. on p. 4).

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *ArXiv* abs/1505.04597 (cit. on p. 24).

Rosenfeld, Elan, Pradeep Ravikumar, and Andrej Risteski (2022). "Domain-adjusted regression or: Erm may already learn features sufficient for out-of-distribution generalization". In: (cit. on p. 200).

Ruan, Yangjun, Yann Dubois, and Chris J. Maddison (2022). "Optimal Representations for Covariate Shift". In: *ICLR* (cit. on pp. 82, 137, 182, 188).

Rubin, Donald B. (Dec. 1976). "Inference and missing data". In: *Biometrika* 63.3, pp. 581–592 (cit. on p. 34).

Ruder, Sebastian (2017). "An Overview of Multi-Task Learning in Deep Neural Networks". In: *CoRR* abs/1706.05098. arXiv: 1706.05098. URL: http://arxiv.org/abs/1706.05098 (cit. on p. 205).

Rusu, Andrei A., Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell (2019). "Meta-Learning with Latent Embedding Optimization". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=BJgklhAcK7 (cit. on p. 110).

Saenko, Kate, Brian Kulis, Mario Fritz, and Trevor Darrell (2010). "Adapting Visual Category Models to New Domains". In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 213–226 (cit. on p. 67).

Sagawa, Shiori, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang (2020). "Distributionally Robust Neural Networks". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: https://openreview.net/forum?id=ryxGuJrFvS (cit. on pp. 17, 115).

Sahebi, Shaghayegh and Peter Brusilovsky (2013). "Cross-Domain Collaborative Recommendation in a Cold-Start Context: The Impact of User Profile Size on the Quality of Recommendation". In: *User Modeling, Adaptation, and Personalization*. Springer Berlin Heidelberg, pp. 289–295 (cit. on p. 37).

Santambrogio, Filippo (2015). *Optimal Transport for Applied Mathematicians*. cvgmt preprint. URL: http://cvgmt.sns.it/paper/2813/ (cit. on p. 61).

Schiesser, W.E. (1991). *The Numerical Method of Lines: Integration of Partial Differential Equations*. Academic Press. URL: https://books.google.fr/books?id=1vLFQgAACAAJ (cit. on p. 23).

Seleznova, Mariia and Gitta Kutyniok (2022). "Neural Tangent Kernel Beyond the Infinite-Width Limit: Effects of Depth and Initialization". In: *ICML* (cit. on pp. 83, 185).

Shah, Harshay, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli (2020). "The Pitfalls of Simplicity Bias in Neural Networks". In: *NeurIPS* (cit. on p. 78).

Shaier, Sagi, Maziar Raissi, and Padmanabhan Seshaiyer (2021). "Data-driven approaches for predicting spread of infectious diseases through DINNs: Disease Informed Neural Networks". In: (cit. on p. 98).

Shen, Jian, Yanru Qu, Weinan Zhang, and Yong Yu (2018). "Wasserstein distance guided representation learning for domain adaptation". In: *32nd AAAI Conference on Artificial Intelligence* (cit. on pp. 14, 15, 68, 141, 166).

Shi, Yuge, Jeffrey Seely, Philip HS Torr, N Siddharth, Awni Hannun, N. Usunier, and Gabriel Synnaeve (2021). "Gradient Matching for Domain Generalization". In: *arXiv preprint* (cit. on p. 18).

Shimodaira, Hidetoshi (2000). "Improving predictive inference under covariate shift by weighting the log-likelihood function". In: *Journal of Statistical Planning and Inference* 90.2, pp. 227–244. URL: https://www.sciencedirect.com/science/article/pii/S0378375800001154 (cit. on pp. 13, 58).

Shui, Changjian, Zijian Li, Jiaqi Li, Christian Gagné, Charles X Ling, and Boyu Wang (18–24 Jul 2021). "Aggregating From Multiple Target-Shifted Sources". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 9638–9648. URL: http://proceedings.mlr.press/v139/shui21a.html (cit. on pp. 58, 64, 65, 70, 71).

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis (2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587, pp. 484–489 (cit. on p. 3).

Singh, Saurabh, Derek Hoiem, and David Forsyth (2016). "Swapout: Learning an ensemble of deep architectures". In: *NeurIPS* (cit. on p. 190).

Sirignano, Justin and Konstantinos Spiliopoulos (2018). "DGM: A deep learning algorithm for solving partial differential equations". In: *Journal of Computational Physics* 375 (Dms 1550918), pp. 1339–1364 (cit. on pp. 98, 118).

Sitzmann, Vincent, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein (2020). "Implicit Neural Representations with Periodic Activation Functions". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 7462–7473 (cit. on pp. 25, 118, 119, 126, 228, 229).

Skorokhodov, Ivan, Sergey Tulyakov, and Mohamed Elhoseiny (June 2022). "StyleGAN-V: A Continuous Video Generator With the Price, Image Quality and Perks of StyleGAN2". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3626–3636 (cit. on p. 25).

Stokes, George Gabriel (1851). "On the Effect of the Internal Friction of Fluids on the Motion of Pendulums". In: *Transactions of the Cambridge Philosophical Society* 9.2, pp. 8–106 (cit. on pp. 109, 126, 215, 224).

Sun, Baochen, Jiashi Feng, and Kate Saenko (2016). "Return of Frustratingly Easy Domain Adaptation". In: *AAAI* (cit. on pp. 17, 78, 88, 89, 192, 197).

Szegedy, Christian, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi (2017). "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI'17. San Francisco, California, USA: AAAI Press, pp. 4278–4284 (cit. on p. 3).

Tancik, Matthew, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng (2020). "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. Curran Associates, Inc., pp. 7537–7547 (cit. on pp. 25, 118).

Teney, Damien, Ehsan Abbasnejad, Simon Lucey, and Anton van den Hengel (2021). "Evading the Simplicity Bias: Training a Diverse Set of Models Discovers Solutions with Superior OOD Generalization". In: *arXiv preprint* (cit. on p. 18).

Thopalli, Kowshik, Sameeksha Katoch, Jayaraman J. Thiagarajan, Pavan K. Turaga, and Andreas Spanias (2021). "Multi-Domain Ensembles for Domain Generalization". In: *NeurIPS Workshop* (cit. on p. 18).

Thrun, Sebastian and Lorien Y. Pratt (1998). "Learning to Learn: Introduction and Overview". In: *Learning to Learn*. Ed. by Sebastian Thrun and Lorien Y. Pratt. Springer, pp. 3–17. URL: https://doi.org/10.1007/978-1-4615-5529-2_1 (cit. on pp. 15, 99).

Thuerey, Nils, Philipp Holl, Maximilian Mueller, Patrick Schnell, Felix Trost, and Kiwon Um (2021). *Physics-based Deep Learning* (cit. on p. 3).

Tran, Luan, Xiaoming Liu, Jiayu Zhou, and Rong Jin (2017). "Missing Modalities Imputation via Cascaded Residual Autoencoder". In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4971–4980 (cit. on p. 35).

Tzeng, Eric, Judy Hoffman, Kate Saenko, and Trevor Darrell (2017). "Adversarial discriminative domain adaptation". In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2962–2971 (cit. on p. 14).

Ueda, Naonori and Ryohei Nakano (1996). "Generalization error of ensemble estimators". In: *ICNN* (cit. on pp. 18, 78, 81, 179).

Van Buuren, S. (2018). *Flexible imputation of missing data*. 2nd ed. Chapman and Hall/CRC (cit. on p. 36).

Venkateswara, Hemanth, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan (2017a). "Deep Hashing Network for Unsupervised Domain Adaptation". In: *(IEEE) CVPR* (cit. on p. 68).

Venkateswara, Hemanth, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan (2017b). "Deep hashing network for unsupervised domain adaptation". In: *CVPR* (cit. on pp. 86, 88, 175, 198).

Villani, C (Jan. 2008). "Optimal transport – Old and new". In: vol. 338, pp. xxii+973 (cit. on p. 163).

Vogels, Thijs, Sai Praneeth Karimireddy, and Martin Jaggi (2019). "PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2019/file/d9fbed9da256e344c1fa46bb46c34c5f-Paper.pdf (cit. on p. 106).

Wandel, Nils, Michael Weinmann, and Reinhard Klein (2021). "Learning Incompressible Fluid Dynamics from Scratch - Towards Fast, Differentiable Fluid Models that Generalize". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: https://openreview.net/forum?id=KUDUoRsEphu (cit. on p. 98).

Wang, C., M. Niepert, and H. Li (2018). "LRMM: Learning to Recommend with Missing Modalities". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3360–3370 (cit. on p. 35).

Wang, Haoxiang, Han Zhao, and Bo Li (2021a). "Bridging Multi-Task Learning and Meta-Learning: Towards Efficient Training and Effective Adaptation". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 10991–11002. URL: http://proceedings.mlr.press/v139/wang21ad.html (cit. on pp. 13, 115).

Wang, Jindong, Cuiling Lan, Chang Liu, Yidong Ouyang, and Tao Qin (2021b). "Generalizing to Unseen Domains: A Survey on Domain Generalization". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*. Ed. by Zhi-Hua Zhou. ijcai.org, pp. 4627–4635. URL: https://doi.org/10.24963/ijcai.2021/628 (cit. on pp. 16, 98).

Wang, Ruoxi, Bin Fu, Gang Fu, and Mingliang Wang (2017). "Deep Cross Network for Ad Click Predictions". In: *Proceedings of the ADKDD'17* (cit. on p. 52).

Wang, Sifan and Paris Perdikaris (2021c). "Long-time integration of parametric evolution equations with physics-informed DeepONets". In: *arXiv preprint arXiv:2106.05384* (cit. on p. 126).

Wang, Xuezhi and Jeff Schneider (2014). "Flexible Transfer Learning under Support and Model Shift". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc. (cit. on p. 58).

Wang, Xuezhi and Jeff Schneider (July 2015). "Generalization Bounds for Transfer Learning under Model Shift". In: *Proceedings of 31st Conference on Uncertainty in Artificial Intelligence (UAI '15)*, pp. 922–931 (cit. on p. 58).

Wei, P, Y Ke, and C. K. Goh (2017). "Domain Specific Feature Transfer for Hybrid Domain Adaptation". In: *2017 IEEE International Conference on Data Mining*, pp. 1027–1032 (cit. on p. 37).

Wei, P, Y Ke, and C. K. Goh (2019). "A General Domain Specific Feature Transfer Framework for Hybrid Domain Adaptation". In: *IEEE Transactions on Knowledge and Data Engineering* 31.8, pp. 1440–1451 (cit. on p. 37).

Weller, Hilary, John Thuburn, and Colin J. Cotter (2012). "Computational Modes and Grid Imprinting on Five Quasi-Uniform Spherical C Grids". In: *Monthly Weather Review* 140.8, pp. 2734–2755 (cit. on p. 130).

Wen, Yeming, Ghassen Jerfel, Rafael Muller, Michael W Dusenberry, Jasper Snoek, Balaji Lakshminarayanan, and Dustin Tran (2021). "Combining Ensembles and Data Augmentation Can Harm Your Calibration". In: *ICLR* (cit. on p. 18).

Wenzel, Florian, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton (2020). "Hyperparameter Ensembles for Robustness and Uncertainty Quantification". In: *NeurIPS* (cit. on p. 18).

Willard, Jared, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar (Jan. 2022). "Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems". In: *ACM Computing Surveys* (cit. on p. 118).

Willard, Jared D., Xiaowei Jia, Shaoming Xu, Michael S. Steinbach, and Vipin Kumar (2020). "Integrating Physics-Based Modeling with Machine Learning: A Survey". In: *ArXiv* abs/2003.04919 (cit. on p. 3).

Wortsman, Mitchell, Maxwell Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari (2021). "Learning Neural Network Subspaces". In: *ICML* (cit. on p. 18).

Wortsman, Mitchell, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt (2022a). "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time". In: *ICML* (cit. on pp. 18, 78, 81, 86, 91, 177, 190).

Wortsman, Mitchell, Gabriel Ilharco, Jong Wook Kim, Mike Li, Hanna Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt (2022b). "Robust fine-tuning of zero-shot models". In: *CVPR* (cit. on pp. 18, 78, 191).

Wu, Yifan, Ezra Winston, Divyansh Kaushik, and Zachary Lipton (Sept. 2019). "Domain Adaptation with Asymmetrically-Relaxed Distribution Alignment". In: *ICML*. Vol. 97. Proceedings of ML Research. Long Beach, CA, USA: PMLR, pp. 6872–6881 (cit. on pp. 68, 70).

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean (2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144. arXiv: 1609.08144. URL: http://arxiv.org/abs/1609.08144 (cit. on p. 3).

Xiao, Ting, Peng Liu, Wei Zhao, Hongwei Liu, and Xianglong Tang (2019). "Structure preservation and distribution alignment in discriminative transfer subspace learning". In: *Neurocomputing* 337, pp. 218–234. URL: https://www.sciencedirect.com/science/article/pii/S0925231219300979 (cit. on p. 58).

Yan, Shen, Huan Song, Nanxiang Li, Lincan Zou, and Liu Ren (2020). "Improve unsupervised domain adaptation with mixup training". In: *arXiv preprint* (cit. on pp. 89, 192, 197).

Yan, Wilson, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas (2021). "VideoGPT: Video generation using VQ-VAE and transformers". In: *arXiv preprint arXiv:2104.10157* (cit. on p. 124).

Yang, Greg and Hadi Salman (2019). "A Fine-Grained Spectral Perspective on Neural Networks". In: *arXiv preprint* (cit. on p. 185).

Yao, Zhewei, Amir Gholami, Kurt Keutzer, and Michael W Mahoney (2020). "Pyhessian: Neural networks through the lens of the hessian". In: *Big Data* (cit. on pp. 174, 175).

Ye, Nanyang, Kaican Li, Lanqing Hong, Haoyue Bai, Yiting Chen, Fengwei Zhou, and Zhenguo Li (2022). "OoD-Bench: Benchmarking and Understanding Out-of-Distribution Generalization Datasets and Algorithms". In: *CVPR* (cit. on pp. 13, 18, 78, 79, 82, 83, 86, 88, 182, 198, 199).

Yeo, Teresa, Oguzhan Fatih Kar, and Amir Roshan Zamir (2021). "Robustness via Cross-Domain Ensembles". In: *ICCV* (cit. on p. 18).

Yin, Y., Ibrahim Ayed, Emmanuel de Bézenac, N. Baskiotis, and P. Gallinari (2021a). "LEADS: Learning Dynamical Systems that Generalize Across Environments". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer,

Y. Dauphin, P. Liang, and J. Wortman Vaughan. URL: https://openreview.net/forum?id=HD6CxZtbmIx (cit. on pp. 26, 99, 110, 115, 205, 216).

Yin, Y., Vincent Le Guen, Jérémie Dona, Emmanuel de Bézenac, Ibrahim Ayed, N. Thome, and P. Gallinari (Dec. 2021b). "Augmenting physical models with deep networks for complex dynamics forecasting". In: *Journal of Statistical Mechanics: Theory and Experiment* 2021.12, p. 124012. URL: https://doi.org/10.1088/1742-5468/ac3ae5 (cit. on pp. 3, 24, 26, 98, 106, 118).

Yin*, Y., M. Kirchmeyer*, J-Y Franceschi*, A. Rakotomamonjy, and P. Gallinari (2023). "Continuous PDE Dynamics Forecasting with Implicit Neural Representations". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 8, 117).

Yıldız, Çağatay, Markus Heinonen, and Harri Lähdesmäki (2019). "ODE²VAE: Deep generative second order ODEs with Bayesian neural networks". In: *Advances in Neural Information Processing Systems*. Ed. by Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Roman Garnett. Vol. 32. Curran Associates, Inc., pp. 13434–13443 (cit. on p. 232).

Yoon, Jinsung, James Jordon, and Mihaela Van Der Schaar (2018). "GAIN: Missing data imputation using generative adversarial nets". In: *Proceedings of the 35th International Conference on Machine Learning*, pp. 5689–5698 (cit. on p. 36).

You, K., X. Wang, M. Long, and M. Jordan (2019). "Towards Accurate Model Selection in Deep Unsupervised Domain Adaptation". In: *Proceedings of the 36th International Conference on Machine Learning*, pp. 7124–7133 (cit. on pp. 49, 149).

Yu, Sihyun, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung-Woo Ha, and Jinwoo Shin (2022). "Generating Videos with Dynamics-aware Implicit Generative Adversarial Networks". In: *International Conference on Learning Representations* (cit. on p. 25).

Zablocki, Eloi, P. Bordes, Laure Soulier, Benjamin Piwowarski, and P. Gallinari (Sept. 2019). "Context-Aware Zero-Shot Learning for Object Recognition". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97, pp. 7292–7303 (cit. on p. 37).

Zanna, Laure and Thomas Bolton (2021). "Deep Learning of Unresolved Turbulent Ocean Processes in Climate Models". In: *Deep Learning for the Earth Sciences*. John Wiley & Sons, Ltd. Chap. 20, pp. 298–306 (cit. on p. 118).

Zech, John R., Marcus A. Badgeley, Manway Liu, Anthony B. Costa, Joseph J. Titano, and Eric Karl Oermann (2018). "Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study". In: *PLOS Medicine* (cit. on p. 78).

Zhang, Kun, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang (17–19 Jun 2013). "Domain Adaptation under Target and Conditional Shift". In: *Pro-*

ceedings of the 30th ICML. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA, pp. 819–827 (cit. on pp. 13, 58, 59, 62, 63, 70, 71, 164).

Zhang, Michael, James Lucas, Jimmy Ba, and Geoffrey E Hinton (2019). "Lookahead optimizer: k steps forward, 1 step back". In: *NeurIPS* 32 (cit. on p. 18).

Zhao, Han, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon (Sept. 2019). "On Learning Invariant Representations for Domain Adaptation". In: *Proceedings of the 36th ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 7523–7532 (cit. on pp. 46, 70).

Zhao, Han, Shanghang Zhang, Guanhang Wu, José M. F. Moura, Joao P Costeira, and Geoffrey J Gordon (2018). "Adversarial Multiple Source Domain Adaptation". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2018/file/717d8b3d60d9eea997b35b02b6a4e867-Paper.pdf (cit. on p. 14).

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei Efros (2017). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *IEEE International Conference on Computer Vision*, pp. 2242–2251 (cit. on p. 36).

Zintgraf, Luisa, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson (Sept. 2019). "Fast Context Adaptation via Meta-Learning". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 7693–7702. URL: http://proceedings.mlr.press/v97/zintgraf19a.html (cit. on pp. 16, 106, 110, 115, 205).