A Representation Learning Perspective on the Importance of Train-Validation Splitting in Meta-Learning

Nikunj Saunshi [1] **Arushi Gupta** [1] **Wei Hu** [1]

Abstract

An effective approach in meta-learning is to utilize multiple "train tasks" to learn a good initialization for model parameters that can help solve unseen "test tasks" with very few samples by fine-tuning from this initialization. Although successful in practice, theoretical understanding of such methods is limited. This work studies an important aspect of these methods: splitting the data from each task into train (support) and validation (query) sets during meta-training. Inspired by recent work (Raghu et al., 2020), we view such meta-learning methods through the lens of representation learning and argue that the train-validation split encourages the learned representation to be *low-rank* without compromising on expressivity, as opposed to the non-splitting variant that encourages high-rank representations. Since sample efficiency benefits from low-rankness, the splitting strategy will require very few samples to solve unseen test tasks. We present theoretical results that formalize this idea for linear representation learning on a subspace meta-learning instance, and experimentally verify this practical benefit of splitting in simulations and on standard meta-learning benchmarks.

1. Introduction

Humans can learn from prior experiences, summarize them into skills or concepts and leverage those to solve new tasks with very few demonstrations (Ahn & Brewer, 1993). Since labeled data is often expensive to obtain, it would be desirable for machine learning agents to emulate this behavior of exploiting data from prior experiences to make solving new tasks sample efficient. In this regard, we consider the meta-learning or learning-to-learn paradigm (Schmidhuber, 1987; Thrun & Pratt, 1998), where a learner utilizes data from many "train" tasks to learn a useful prior that can help solve new "test" tasks. The goal is to do well on test tasks with way fewer samples per task than would be required without access to train tasks. There is a rich history of successful methods in meta-learning and related fields of multi-task, lifelong and few-shot learning (Evgeniou & Pontil, 2004; Ruvolo & Eaton, 2013; Vinyals et al., 2016). The advent of the deep learning pipeline has led to more recent model-agnostic methods (Finn et al., 2017) that learn a good initialization for model parameters by using train tasks (meta-training) and solve test tasks by fine-tuning from this initialization using little data (meta-testing). Such methods have helped with many problems like few-shot classification in computer vision (Nichol et al., 2018), reinforcement learning (Finn et al., 2017), federated learning (McMahan et al., 2017), neural machine translation (Gu et al., 2018).

This empirical success has encouraged theoretical studies (cf. Section 2) of the statistical and optimization properties of such methods. We are interested in the statistical aspect, where a meta-learner is evaluated based on the number of test task samples it requires to achieve small error. Specifically, we focus our attention on the design choice of data splitting that is used in many meta-learning methods (Finn et al., 2017; Rajeswaran et al., 2019; Raghu et al., 2020). In this setting, a typical meta-learner learns a good initialization by 1) splitting the train task data into train (support) and validation (query) sets, 2) running the inner loop or *base-learner* of choice on the *train set* of the task starting from the current initialization, 3) minimizing the loss incurred on the *validation set* of the task by parameters from the inner loop, 4) updating the initialization in outer loop. The learned initialization is used to solve an unseen *test task*, typically in the few-shot regime with very few samples available. Here we study the sample complexity benefit offered by such train-validation (tr-val) splitting by arguing that it learns *low-rank & expressive* representations.

We further compare this splitting strategy to the non-splitting variant where the entire task data is used for both, inner loop and outer loop updates (Nichol et al., 2018; Zhou et al., 2019). Recent theoretical work (Bai et al., 2020) makes the case for the non-splitting variant (referred to as tr-tr variant) with an analysis in the centroid learning setting for

[1]Princeton University, Princeton, NJ, USA. Correspondence to: Nikunj Saunshi <nsaunshi@cs.princeton.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

noiseless Gaussian linear regression and arguing that the non-splitting variant uses the available data more effectively in an asymptotic regime (cf. Section 5.3). We instead take a representation learning perspective, inspired by Raghu et al. (2020) arguing that the power of modern meta-learning methods can be derived from the good representations they learn. Our main contributions are summarized below:

- **Big picture:** We study the setting of representation learning, where the meta-learner trains a representation function (all weights before the linear classifier head) in the outer loop, while the inner loop learns a linear classifier on top of fixed representations. We show, theoretically and experimentally, that tr-val splitting helps with sample complexity of few-shot learning by inducing an *implicit regularization* that encourages the learning of *expressive low-rank representations*. On the theoretical side, we prove the sample efficiency of the tr-val objective for linear representations, going beyond previous theoretical results that require explicit regularization or low rank constraints. We also show potential failures of the tr-tr variant without explicit regularization, contrasting recent theoretical work (Bai et al., 2020) that shows that tr-tr is better than tr-val in a different setting. Experiments support our hypothesis on simulations and standard benchmarks.
- **Theory:** Our theoretical study is for a representation function that *linearly* maps d-dimensional inputs to d-dimensional[1] representations, while the inner loop performs ridge regression on fixed representation. We consider a meta-learning instance where tasks are linear regression problems with regressors on a k-dimensional subspace ($k \ll d$). Firstly we show that approximately minimizing the tr-val objective guarantees learning an expressive low-rank representation, which is precisely the underlying k-dimensional subspace. This results in a new task sample complexity of $\mathcal{O}(k)$, i.e. only $\mathcal{O}(k) \ll d$ samples are required from test tasks to achieve a small error. In contrast, no matter how well the tr-tr objective is optimized, one cannot guarantee learning a "good" representation and we may end up having an $\Omega(d)$ new task samples complexity. By exploiting the practical design choice of tr-val splitting, we go beyond previous analysis that require explicit regularization or low-rank constraints on linear representations. Our analysis allows for different number of samples per task in meta-training and testing, unlike previous excess risk based analyses (Baxter, 2000; Maurer et al., 2016; Denevi et al., 2018a).
- **Experiments:** Simulations on the subspace meta-learning instance verify that the tr-val splitting leads to smaller few-shot error by virtue of learning the right subspace. In contrast, the tr-tr objective trained using standard methods has poor performance on the same problem. We also test on standard few-shot classification benchmarks like Omniglot (Lake et al., 2015) and MiniImageNet (Vinyals et al., 2016), where again the tr-val objective turns out to be superior to tr-tr objective, for representation learning and a gradient based method implicit MAML (Rajeswaran et al., 2019). We further find that while the tr-tr and tr-val representations are both expressive enough to separate all test classes with a lot of training data, the tr-val representation has lower effective rank. Thus although the theory is for the simplistic model of linear representations, it provide useful insights into the practice of meta-learning.

[1]Dimensionality of representation can be smaller too. Larger makes the results stronger.

After discussing related work in Section 2, we present meta-learning preliminaries and describe the representation learning framework and the tr-val, tr-tr objectives in Section 3. Section 4 defines the subspace meta-learning instance for our theoretical analysis and formulates linear representation learning. We present our theoretical results in Section 5 for the tr-tr and tr-val objectives, along with intuitive explanations and comparisons to previous resutls. We present our experimental verifications and findings in Section 6. Proofs and additional experiment details are in the Appendix.

2. Related Work

Background and empirical work: Learning-to-learn or meta-learning has a rich history (Schmidhuber, 1987; Bengio et al., 1990; Naik & Mammone, 1992; Caruana, 1997; Thrun & Pratt, 1998). Existing deep learning based methods can be broadly classified as metric-based (Vinyals et al., 2016; Snell et al., 2017), model-based (Andrychowicz et al., 2016; Ravi & Larochelle, 2017) and more relevant to us, gradient-based (Finn et al., 2017; Nichol et al., 2018) methods. Recent empirical work (Raghu et al., 2020) shows that most of the power of gradient-based methods like MAML (Finn et al., 2017) can be derived by fixing representations and learning a linear classifier for new tasks. Their proposed representation learning algorithm ANIL and other methods (Lee et al., 2019; Bertinetto et al., 2019) show that representation learning performs comparably to gradient-based methods on many benchmarks. We note that Oh et al. (2020) highlights the importance of updating representations in the inner loop. Recent works also try to demystify these methods by studying the role of depth (Arnold et al., 2019) and clustering of representations (Goldblum et al., 2020).

Theoretical work: The seminal work of Baxter (2000) introduced a framework to study the statistical benefits of meta-learning, subsequently used to show excess risk bounds for ERM-like methods using techniques like covering numbers (Baxter, 2000), algorithmic stability (Maurer, 2005) and

Gaussian complexities (Maurer et al., 2016). Sample complexity guarantees for *initialization-based* methods have been shown for inner loop losses that are convex in model parameters (Khodak et al., 2019a;b; Zhou et al., 2019); in particular for squared-error loss (Denevi et al., 2018b; Bai et al., 2020), a.k.a. centroid meta-learning. The underlying structure exploited in these guarantees is the closeness of optimal task parameters for different tasks in some metric.

Another paradigm for analyzing meta-learning methods is representation learning (Maurer et al., 2016; Du et al., 2020; Tripuraneni et al., 2020b). *Linear representation* learning is a popular playground to theoretically study various aspects of meta-learning (Argyriou et al., 2008; Maurer, 2009; Maurer & Pontil, 2013; Bullins et al., 2019; Denevi et al., 2018a; Du et al., 2020; Tripuraneni et al., 2020a;b). These analyses do not employ a train-validation split but instead require explicit regularizers or low-rank constraints on representations to show meaningful guarantees, which we do not need. Saunshi et al. (2020) show that Reptile (Nichol et al., 2018) and gradient descent on linear representation learning (without data splitting or regularization) learn an underlying 1-dimensional subspace and show a separation from centroid meta-learning methods.

Other theoretical analyses show PAC-Bayes bounds (Pentina & Lampert, 2014), and regret bounds for life-long learning (Alquier et al., 2017) and study mixed linear regression (Kong et al., 2020). Methods like MAML are also studied from an optimization perspective (Rajeswaran et al., 2019; Fallah et al., 2020; Wang et al., 2020a; Collins et al., 2020), but we are interested in statistical aspects.

Data splitting: The role of data splitting was theoretically considered in a recent work (Bai et al., 2020) for the centroid learning problem. While Denevi et al. (2018b) shows bounds for tr-val splitting, as noted in Bai et al. (2020), their bounds cannot differentiate between tr-tr and tr-val objectives. (Wang et al., 2020b) shows that tr-tr is worse than tr-val splitting for tuning learning rate using meta-learning. We do a detailed comparison to data splitting and linear representation learning results in Section 5.3.

3. Meta-Learning Preliminaries

Notation: We use $\boldsymbol{x}$ is used to denote a vector, S to denote a set, $\boldsymbol{A}$ to denote matrix, θ to denote model parameters (like weights of neural network). $\boldsymbol{x} \sim S$ denotes sampling an element $\boldsymbol{x}$ from the uniform distribution over a finite set S. $\mathcal{N}(\mathbf{0}_d, \Sigma)$ is Gaussian distribution with mean $\mathbf{0}_d$ and covariance Σ. I_d denotes a $d \times d$ identity matrix. The ReLU function is denoted by $(x)_+ = \mathbf{1}_{x>0}\, x$. For $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, $\boldsymbol{A}^\dagger$ denotes the pseudo-inverse of $\boldsymbol{A}$ and $P_{\boldsymbol{A}} = (\boldsymbol{A}\boldsymbol{A}^\dagger)$ denotes the projection matrix for the column span of $\boldsymbol{A}$.

3.1. Data Splitting and Meta-Learning

We formalize the train-validation split that is used in practice and define the tr-tr and tr-val objective functions. Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ denote space of data points, where $\mathcal{X}$ is the input space, $\mathcal{Y}$ is the label space. Let $\theta \in \Theta \subseteq \mathbb{R}^D$ denote the model parameters (e.g. weights of a neural network) and $f_\theta(\boldsymbol{x}) \in \mathbb{R}^c$ denote the model prediction[2] for input $\boldsymbol{x} \in \mathcal{X}$. A meta-learner has access to T train tasks as datasets $S_1, \ldots, S_T$, with each dataset S_t of n points being split into $S_t^{\text{tr}} = \{(\boldsymbol{x}_{t,i}^{\text{tr}}, y_{t,i}^{\text{tr}})\}_{i=1}^{n_1}$ and $S_t^{\text{val}} = \{(\boldsymbol{x}_{t,i}^{\text{val}}, y_{t,i}^{\text{val}})\}_{i=1}^{n_2}$, with $S_t = S_t^{\text{tr}} \cup S_t^{\text{val}}$, where S_t^{tr} and S_t^{val} refer to the train and validation splits respectively. For a loss function ℓ, e.g. logistic loss or squared error loss, we define the average loss incurred by model parameters $\theta \in \Theta$ on a dataset S as

$$\ell(\theta; S) = \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim S}[\ell(f_\theta(\boldsymbol{x}), y)] \tag{1}$$

Initialization-based meta-learning methods (Finn et al., 2017) aim to learn an initialization $\theta_0 \in \Theta$ for model parameters such that solving a task using its train data and the initialization θ_0 with an algorithm $\mathcal{A}$ will lead to parameters that do well on the test data for the task. Formally, $\mathcal{A} : \Theta \times \mathcal{Z}^n \to \Theta$ is an *inner algorithm* or base-learner, e.g. $\mathcal{A}(\theta_0, S)$ can run a few steps of gradient descent starting from θ_0 on the loss $\ell(\cdot; S)$. We now describe the train-validation (tr-val) and train-train (tr-tr) *outer algorithms* or meta-learners as ones that minimize the following objectives

$$\widehat{\mathcal{L}}^{\text{tr-val}}(\theta; \mathcal{A}) = \frac{1}{T}\sum_{t=1}^{T} \ell(\mathcal{A}(\theta, S_t^{\text{tr}}); S_t^{\text{val}}) \tag{2}$$

$$\widehat{\mathcal{L}}^{\text{tr-tr}}(\theta; \mathcal{A}) = \frac{1}{T}\sum_{t=1}^{T} \ell(\mathcal{A}(\theta, S_t); S_t) \tag{3}$$

Our tr-tr and tr-val objective definitions are similar to those from (Bai et al., 2020). We now describe the representation learning objective and the corresponding inner algorithm $\mathcal{A}$.

3.2. Representation Learning

As in Raghu et al. (2020), we define a representation learning objective where the inner algorithm only learns a linear predictor on top of fixed learned representations. Let θ^{rep} parametrize the representation function $f_{\theta^{\text{rep}}} : \mathcal{X} \to \mathbb{R}^d$

Base-learner: For any task, we define the inner algorithm $\mathcal{A}_{\lambda,\text{rep}}$ that only learns a linear classifier $\boldsymbol{W} \in \mathbb{R}^{d\times c}$ as

$$\ell_{\lambda,\text{rep}}(\boldsymbol{W}; \theta^{\text{rep}}, S) = \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim S}[\ell(\boldsymbol{W}^\top f_{\theta^{\text{rep}}}(\boldsymbol{x}), y)] + \frac{\lambda}{2}\|\boldsymbol{W}\|_F^2$$

$$\mathcal{A}_{\lambda,\text{rep}}(\theta^{\text{rep}}; S) = \arg\min_{\boldsymbol{W}} \ell_{\lambda,\text{rep}}(\boldsymbol{W}; \theta^{\text{rep}}, S) \tag{4}$$

[2]For classification, model prediction is the logits and c is number of classes. For regression, c is target dimension.

Meta-learner: The corresponding tr-val and tr-tr objectives that use the base-learner $\mathcal{A}_{\lambda,\text{rep}}$ are

$$\widehat{\mathcal{L}}^{\{\text{tr-tr,tr-val}\}}_{\lambda,\text{rep}}(\theta^{\text{rep}}) := \widehat{\mathcal{L}}^{\{\text{tr-tr,tr-val}\}}(\theta^{\text{rep}}; \mathcal{A}_{\lambda,\text{rep}}) \tag{5}$$

where $\widehat{\mathcal{L}}^{\{\text{tr-tr,tr-val}\}}$ are defined in Equations (2) and (3). The inner algorithm from Equation (4), that uses $\|\cdot\|_F$ regularization, has been used in empirical work, for ℓ being the logistic and squared error losses in Bertinetto et al. (2019) and the margin loss in Lee et al. (2019); also used in theoretical work (Saunshi et al., 2020). As evidenced in these works, minimizing $\widehat{\mathcal{L}}^{\text{tr-val}}_{\lambda,\text{rep}}$ performs very well in practice. In the subsequent sections we will show, theoretically and experimentally, that $\widehat{\mathcal{L}}^{\text{tr-val}}_{\lambda,\text{rep}}$ learned representations are better than $\widehat{\mathcal{L}}^{\text{tr-tr}}_{\lambda,\text{rep}}$ for few-shot learning.

4. Meta-Learning of Linear Representations

In this section we discuss the subspace meta-learning instance for which we show theoretical guarantees. We also define the tr-tr and tr-val objectives and the meta-testing metric for linear representation learning.

4.1. Subspace Meta-Learning Instance

We construct a simple meta-learning instance where each task is a regression problem. We assume that the T train tasks and the unseen test tasks will be sampled from an underlying distribution over tasks, as in many prior theoretical work (Baxter, 2000; Maurer et al., 2016; Denevi et al., 2018b; Bai et al., 2020). Each task is a regression problem, with the input and target spaces being $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$ respectively. A task $\rho_{\boldsymbol{v}}$ is characterized by a vector $\boldsymbol{v} \in \mathbb{R}^d$ and has an associated distribution[3] over $\mathcal{X} \times \mathcal{Y}$ that satisfies

$$(\boldsymbol{x}, y) \sim \rho_{\boldsymbol{v}} \;\equiv\; \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}_d, I_d), y \sim \mathcal{N}(\boldsymbol{v}^\top \boldsymbol{x}, \sigma^2) \tag{6}$$

Thus the input distribution is standard normal, while the label is a linear function of input with some Gaussian noise[4] with variance σ^2 added to it. We are interested in a meta-learning instance that only considers a subset of the tasks $\{\rho_{\boldsymbol{v}}\}$, namely for $\boldsymbol{v} \in \mathbb{R}^d$ that lie on an *unknown* k-dimensional subspace for some $k \ll d$. Let $\boldsymbol{A}^* \in \mathbb{R}^{d\times k}$ be an orthogonal matrix[5] ($\boldsymbol{A}^{*\top}\boldsymbol{A}^* = I_k$) whose columns define the basis for this subspace. We define the distribution μ over tasks $\{\rho_{\boldsymbol{v}}\}$ that is supported on span($\boldsymbol{A}^*$) as follows

$$\rho_{\boldsymbol{v}} \sim \mu \;\equiv\; \boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}_d, \boldsymbol{A}^*\boldsymbol{A}^{*\top}) \tag{7}$$

The optimal classifier for each regression task $\rho_{\boldsymbol{v}}$ is $\boldsymbol{v}$. Since all classifiers $\boldsymbol{v}$ of interest will be on a k-dimensional subspace spanned by $\boldsymbol{A}^*$, we only need to know the projections of the d-dimensional inputs onto $\boldsymbol{A}^*$ to solve all tasks in μ. Thus an expressive low rank linear representation $\boldsymbol{x} \to \boldsymbol{A}^*\boldsymbol{A}^{*\top}\boldsymbol{x}$ exists for the inputs, but this subspace $\boldsymbol{A}^*$ is unknown to the meta-learner. We note that a 1-dimensional subspace meta-learning instance ($k = 1$) was considered in Saunshi et al. (2020) to show guarantees for Reptile.

Meta-training dataset We assume access to T tasks $\rho_{\boldsymbol{v}_1}, \ldots, \rho_{\boldsymbol{v}_T}$ sampled independently from the distribution μ defined in Equation (7). For the task $\rho_{\boldsymbol{v}_t}$ a dataset of n points, $S_t = (\boldsymbol{X}_t, \boldsymbol{Y}_t)$, where $\boldsymbol{X}_t \in \mathbb{R}^{n\times d}, \boldsymbol{Y} \in \mathbb{R}^n$ is sampled from $\rho_{\boldsymbol{v}_t}$. Each dataset $S_t = (\boldsymbol{X}_t, \boldsymbol{Y}_t)$ is split into $S_t^{\text{tr}} = (\boldsymbol{X}_t^{\text{tr}}, \boldsymbol{Y}_t^{\text{tr}})$ and $S_t^{\text{val}} = (\boldsymbol{X}_t^{\text{val}}, \boldsymbol{Y}_t^{\text{val}})$ with sizes n_1 and n_2 respectively, with $n_1 + n_2 = n$.

4.2. Linear Representation Learning

We parametrize the predictor as a two-layer linear network, denoting the representation layer as $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ and the linear classifier layer as $\boldsymbol{w} \in \mathbb{R}^d$. The loss function is the squared-error loss $\ell(\hat{y}, y) = (\hat{y} - y)^2$. We now describe the base-learner, meta-learners and meta-testing metric.

Base-learner: Recall that for each task $t \in [T]$, the task data of size $S_t = (\boldsymbol{X}_t, \boldsymbol{Y}_t)$ of size n is split into $S_t^{\text{tr}} = (\boldsymbol{X}_t^{\text{tr}}, \boldsymbol{Y}_t^{\text{tr}})$ and $S_t^{\text{val}} = (\boldsymbol{X}_t^{\text{val}}, \boldsymbol{Y}_t^{\text{val}})$ of sizes n_1 and n_2 respectively. For the squared error loss, the inner loop for data $S = (\boldsymbol{X}, \boldsymbol{Y})$ reduces to the following:

$$\begin{aligned}\ell_{\lambda,\text{rep}}(\boldsymbol{w}; \boldsymbol{A}, S) &= \frac{1}{n}\|\boldsymbol{X}\boldsymbol{A}\boldsymbol{w} - \boldsymbol{Y}\|^2 + \lambda\|\boldsymbol{w}\|_2^2 \\ \boldsymbol{w}_\lambda(\boldsymbol{A}; S) &= \operatorname*{arg\,min}_{\boldsymbol{w}\in\mathbb{R}^d} \ell_{\lambda,\text{rep}}(\boldsymbol{w}; \boldsymbol{A}, S)\end{aligned} \tag{8}$$

Meta-learner: The tr-val and tr-tr objectives to learn representation layer $\boldsymbol{A}$ on T train tasks are described below:

$$\begin{aligned}&\widehat{\mathcal{L}}^{\text{tr-val}}_{\lambda,\text{rep}}(\boldsymbol{A}; (n_1, n_2)) \\ &\qquad = \frac{1}{Tn_2}\sum_{t=1}^{T} \left\|\boldsymbol{X}_t^{\text{val}}\boldsymbol{A}\boldsymbol{w}_\lambda(\boldsymbol{A}; S_t^{\text{tr}}) - \boldsymbol{Y}_t^{\text{val}}\right\|^2\end{aligned} \tag{9}$$

$$\widehat{\mathcal{L}}^{\text{tr-tr}}_{\lambda,\text{rep}}(\boldsymbol{A}; n) = \frac{1}{Tn}\sum_{t=1}^{T} \|\boldsymbol{X}_t\boldsymbol{A}\boldsymbol{w}_\lambda(\boldsymbol{A}; S_t) - \boldsymbol{Y}_t\|^2 \tag{10}$$

We note that the linear representation learning literature considers a similar objective to $\widehat{\mathcal{L}}^{\text{tr-tr}}_{\lambda,\text{rep}}$ (Bullins et al., 2019; Denevi et al., 2018a; Du et al., 2020; Tripuraneni et al., 2020b), albeit with either a Frobenius norm constraint on $\boldsymbol{A}$ or a rank constraint with $\boldsymbol{A} \in \mathbb{R}^{d\times k}$ rather than $\mathbb{R}^{d\times d}$. We show that such a constraint is not needed for the $\widehat{\mathcal{L}}^{\text{tr-val}}_{\lambda,\text{rep}}(\boldsymbol{A})$ objective, since it implicitly induces such a regularization.

Meta-testing/evaluation We evaluate the learned representation layer $\boldsymbol{A}$ on a test task $\rho_{\boldsymbol{v}}$ with only $\bar{n}_1$ train samples from the task. The inner algorithm first uses $\boldsymbol{A}$ to learn $\boldsymbol{w}_{\bar{\lambda}}(\boldsymbol{A}; S)$[6] from Equation (8) on $\bar{n}_1$ samples S sampled

[3] We abuse notation and use $\rho_{\boldsymbol{v}}$ as a task and its distribution.

[4] We only need $\mathbb{E}[y|\boldsymbol{x}] = \boldsymbol{v}^\top\boldsymbol{x}$, $\text{Var}[y|\boldsymbol{x}] = \sigma^2 I_d$.

[5] For simplicity. Results hold even if $\boldsymbol{A}^*$ is not orthonormal.

[6] The regularization parameter $\bar{\lambda}$ can also be different from λ.

from $\rho_{\boldsymbol{v}}$. The final predictor $\boldsymbol{A}\boldsymbol{w}_{\bar{\lambda}}(\boldsymbol{A}; S) \in \mathbb{R}^d$ is then evaluated on $\rho_{\boldsymbol{v}}$: $\mathbb{E}_{(\boldsymbol{x},y)\sim\rho_{\boldsymbol{v}}} \|\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{w} - y\|^2 = \sigma^2 + \|\boldsymbol{A}\boldsymbol{w} - \boldsymbol{v}\|^2$. Formally we define the meta-testing metric as an average loss over a test tasks sampled from $\bar{\mu}$

$$\mathcal{L}^{\text{test}}_{\bar{\lambda},\text{rep}}(\boldsymbol{A}; \bar{n}_1) = \mathop{\mathbb{E}}_{\rho_{\boldsymbol{v}}\sim\bar{\mu}} \left[\mathop{\mathbb{E}}_{S\sim\rho_{\boldsymbol{v}}^{\bar{n}_1}} \left[\|\boldsymbol{A}\boldsymbol{w}_{\bar{\lambda}}(\boldsymbol{A}; S) - \boldsymbol{v}\|^2 \right] \right] + \sigma^2 \tag{11}$$

This metric is similar to the one used in prior work (Baxter, 2000; Maurer et al., 2016; Denevi et al., 2018a; Saunshi et al., 2020), however the distribution over test tasks $\bar{\mu}$ need not be the same as μ. All we need is that test tasks lies on the subspace spanned on $\boldsymbol{A}^*$.

Assumption 4.1. *For all $\rho_{\boldsymbol{v}} \sim \bar{\mu}$, $\boldsymbol{A}^*\boldsymbol{A}^{*\top}\boldsymbol{v} = \boldsymbol{v}$*

Note also that samples per test task $\bar{n}_1$ can also be different from n or n_1 available during meta-training.

5. Theoretical Results

We discuss theoretical results for linear representation learning on the meta-learning instance defined in Section 4.1. First we show that trying to minimize the tr-tr objective without any explicit regularization can end up learning a full rank representation, thus leading to $\Omega(d)$ sample complexity on a new task. While this is not too surprising or hard to show, it suggests that for the non-splitting variant to succeed, it needs to depend on either some explicit regularization term or low-rank constraints as imposed in prior work, or rely on inductive biases of the training algorithm. We then show our main result for the tr-val objective showing that minimizing it implicitly imposes a low-rank constraint on representations and leads to new task sample complexity of $\mathcal{O}(k)$. Proving this results entails deriving a closed form expression for the asymptotic tr-val objective using symmetry in the Gaussian distribution and properties of the inverse Wishart distribution (Mardia et al., 1979).

5.1. Results for Train-Train Objective

For theoretical results in this section, we look at the asymptotic limit as the number of train tasks T goes to infinity, but with each task having just n samples[7].

$$\mathcal{L}^{\text{tr-tr}}_{\lambda,\text{rep}}(\boldsymbol{A}; n) = \lim_{T\to\infty} \widehat{\mathcal{L}}^{\text{tr-tr}}_{\lambda,\text{rep}}(\boldsymbol{A}; n) \tag{12}$$

This limit gives an indication of the best one could hope to do with access to many tasks with small amount of data (n) per task. Note that the results in this subsection can be extended to the non-asymptotic case of finite T as well.

The goal of $\mathcal{L}^{\text{tr-tr}}_{\lambda,\text{rep}}$ is to learn a first representation layer $\boldsymbol{A}$ such the linear classifier $\boldsymbol{w}_\lambda(\boldsymbol{A}; S_t)$ learned using the task data $S = (\boldsymbol{X}, \boldsymbol{Y})$ can fit the same task data S well. While a low rank layer ($\boldsymbol{A} = \boldsymbol{A}^*$) can perfectly predict the signal component of the the labels ($\boldsymbol{X}_t\boldsymbol{v}_t$) for tasks in μ, fitting the random noise in the labels ($\boldsymbol{Y}_t - \boldsymbol{X}_t\boldsymbol{v}_t$) requires a representation that is as expressive as possible. However as we show subsequently, a full rank representation layer is bad for the final meta-evaluation $\mathcal{L}^{\text{test}}_{\bar{\lambda},\text{rep}}$. We formalize this idea below and show the existence of such "bad" full rank representations that make the $\mathcal{L}^{\text{tr-tr}}_{\lambda,\text{rep}}$ arbitrarily small.

Theorem 5.1. *For every $\lambda, n > 0$, for every $\tau > 0$, there exists a "bad" representation layer $\boldsymbol{A}_{bad} \in \mathbb{R}^{d\times d}$ that satisfies $\mathcal{L}^{tr\text{-}tr}_{\lambda,rep}(\boldsymbol{A}_{bad}; n) \le \inf_{\boldsymbol{A}\in\mathbb{R}^{d\times d}} \mathcal{L}^{tr\text{-}tr}_{\lambda,rep}(\boldsymbol{A}; n) + \tau$, but has the following lower bound on meta-testing loss*

$$\inf_{\bar{\lambda}>0} \mathcal{L}^{test}_{\bar{\lambda},rep}(\boldsymbol{A}_{bad}; \bar{n}_1) - \sigma^2 \ge \min\left\{1 - \bar{n}_1/d(1+\sigma^2), d\sigma^2/(1+\sigma^2)\bar{n}_1\right\}$$

The fixed error of σ^2 is unavoidable for any method due to noise in labels. In the few-shot regime of $\bar{n}_1 \ll d$, the lower bound on the error is close to 1. To get an error of $\sigma^2 + \epsilon$ on a new test task for a very small ϵ, the number of samples $\bar{n}_1$ for the new tasks must satisfy $\bar{n}_1 = \Omega(\frac{d}{\epsilon})$.

Implications Theorem 5.1 implies that no matter how close $\mathcal{L}^{\text{tr-tr}}_{\lambda,\text{rep}}(\boldsymbol{A})$ gets to the optimal value, one cannot guarantee good performance for $\boldsymbol{A}$ in meta-testing $\mathcal{L}^{\text{test}}_{\bar{\lambda},\text{rep}}(\boldsymbol{A})$. This does not, however, rule out the existence of "good" representations that have a small $\mathcal{L}^{\text{tr-tr}}_{\lambda,\text{rep}}$ and small $\mathcal{L}^{\text{test}}_{\bar{\lambda},\text{rep}}$. However our result suggests that simply trying to minimize $\mathcal{L}^{\text{tr-tr}}_{\lambda,\text{rep}}$ may not be enough to learn good representations. We show in Section 6 that standard meta-learning algorithms with a tr-tr objective can end up learning bad representations, even if not the worst one. We demonstrate this for the above meta-learning instance and also for standard benchmarks, thus suggesting that our result has practical bearing.

Prior work has shown guarantees for linear representation learning algorithms without any data splitting. The main difference is that these methods either add a norm regularization/constraint on the representation layer (Argyriou et al., 2008; Maurer, 2009; Maurer & Pontil, 2013; Bullins et al., 2019; Denevi et al., 2018a) to encourage low-rankness or an explicit low rank constraint on the representation (Du et al., 2020; Tripuraneni et al., 2020a;b). We show that even for this simple meta-learning instance, lack of rank constraint or norm regularization can lead to terrible generalization to new tasks, as also demonstrated in experiments in Section 6. In Table 8 we find that adding a regularization term on the representation does lead to much better performance, albeit still slightly worse that the tr-val variant.

Proof sketch We prove Theorem 5.1 by first arguing that $\lim_{\kappa\to\infty} \mathcal{L}^{\text{tr-tr}}_{\lambda,\text{rep}}(\kappa I_d; n) = \inf_{\boldsymbol{A}\in\mathbb{R}^{d\times d}} \mathcal{L}^{\text{tr-tr}}_{\lambda,\text{rep}}(\boldsymbol{A}; n)$. Thus pick-

[7] Standard benchmarks that solve N-way K-shot tasks with C classes can effective produce $\binom{C}{N}$ tasks which can be large

ing $\boldsymbol{A}_{\text{bad}} = \kappa I_d$ for a large enough κ suffices to get an ϵ-good $\mathcal{L}^{\text{tr-tr}}_{\lambda,\text{rep}}$ solution. On the other hand since κI_d treats all directions alike, it does not encode anything about the k-dimensional subspace $\boldsymbol{A}^*$. The test task, thus, reduces to linear regression on d-dimensional isotropic Gaussians data, which is known to have $\Omega\left(\frac{d}{\epsilon}\right)$ sample complexity to get loss at most ϵ. The lemma below shows that κI_d converges to the infimum of $\mathcal{L}^{\text{tr-tr}}_{\lambda,\text{rep}}$, since a lower rank layer will not fit the noise in the train labels well enough.

Lemma 5.2. *For every $\lambda > 0$ and $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ with rank r,*

$$\mathcal{L}^{\textit{tr-tr}}_{\lambda,\textit{rep}}(\boldsymbol{A}; n) \geq \lim_{\kappa\to\infty} \mathcal{L}^{\textit{tr-tr}}_{\lambda,\textit{rep}}(\kappa\boldsymbol{A}; n) \geq \sigma^2 \frac{(n-r)_+}{n}$$
$$\& \quad \lim_{\kappa\to\infty} \mathcal{L}^{\textit{tr-tr}}_{\lambda,\textit{rep}}(\kappa I_d; n) = \sigma^2 \frac{(n-d)_+}{n}$$

We note that this result can be shown without needing Gaussianity assumptions. The ideas from this result are applicable beyond linear representations. In general, making the $\mathcal{L}^{\text{tr-tr}}_{\lambda,\text{rep}}$ objective small can be achieved by learning a very expressive representation can fit the training data perfectly. There is nothing in the objective function that encourages forgetting redundant information in the representation. We now present the main results for tr-val objective.

5.2. Result for Train-Validation Objective

We again look at the asymptotic limit as the number of train tasks T goes to infinity, with each task having an (n_1, n_2) train-validation split.

$$\mathcal{L}^{\text{tr-val}}_{\lambda,\text{rep}}(\boldsymbol{A}; (n_1, n_2)) = \lim_{T\to\infty} \widehat{\mathcal{L}}^{\text{tr-val}}_{\lambda,\text{rep}}(\boldsymbol{A}; (n_1, n_2)) \tag{13}$$

Here we demonstrate two benefits of the tr-val.

Correct objective to minimize The first result formalizes the intuitive benefit of the $\mathcal{L}^{\text{tr-val}}_{\lambda,\text{rep}}$ objective: if number of train samples in train tasks and test tasks are the same, i.e. $n_1 = \bar{n}_1$, then $\mathcal{L}^{\text{tr-val}}_{\lambda,\text{rep}}$ is the the correct objective to minimize

Proposition 5.3. *$\mathcal{L}^{\textit{tr-val}}_{\lambda,\textit{rep}}(\cdot; (n_1, n_2))$ and $\mathcal{L}^{\textit{test}}_{\bar{\lambda},\textit{rep}}(\cdot; \bar{n}_1)$ are equivalent if $\bar{n}_1 = n_1$ and $\bar{\lambda} = \lambda$*

Thus when $n_1 = \bar{n}_1$, $\widehat{\mathcal{L}}^{\text{tr-val}}_{\lambda,\text{rep}}$ is an unbiased estimate of $\mathcal{L}^{\text{test}}_{\lambda,\text{rep}}$ and minimizing it makes sense. This is easy to prove by exploiting the independence of S^{tr} and S^{val} in the expression for $\widehat{\mathcal{L}}^{\text{tr-val}}_{\lambda,\text{rep}}$ in Equation (2). However when $\bar{n}_1 \neq n_1$ or $\lambda \neq \bar{\lambda}$, the objective functions are different, so it is apriori unclear why minimizing $\mathcal{L}^{\text{tr-val}}_{\lambda,\text{rep}}$ should be a good idea if we care about $\mathcal{L}^{\text{test}}_{\lambda,\text{rep}}$. This is what we handle next.

Low rank representation We show here that for $\lambda = 0$, the minimizers of $\mathcal{L}^{\text{tr-val}}_{\lambda,\text{rep}}$ are the same for almost all n_1 and are in fact rank-k representations that span the subspace of interest $\boldsymbol{A}^*$. This low rankness also ensures that the sample complexity of a new task is $\mathcal{O}\left(\frac{k}{\epsilon}\right)$.

Theorem 5.4. *Let $\lambda = 0$. If $n_1 \geq c_1 k$, $\sigma^2 \in (0, c_2)$, $\tau \leq c_3 \frac{\sigma^2}{n_1}$ for small constants c_1, c_2, c_3, then any $\boldsymbol{A}_{\textit{good}} \in \mathbb{R}^{d\times d}$ that is τ-optimal, i.e. $\mathcal{L}^{\textit{tr-val}}_{\lambda,\textit{rep}}(\boldsymbol{A}_{\textit{good}}; (n_1, n_2)) \leq \inf_{\boldsymbol{A}\in\mathbb{R}^{d\times d}} \mathcal{L}^{\textit{tr-val}}_{\lambda,\textit{rep}}(\boldsymbol{A}; (n_1, n_2)) + \tau$, will satisfy*

$$\textit{rank}(\boldsymbol{A}_{\textit{good}}) = k, \quad \|P_{\boldsymbol{A}_{\textit{good}}}\boldsymbol{A}^* - \boldsymbol{A}^*\|^2 \leq \tau$$

and the meta-testing performance $\bar{n}_1 > 2k + 2$ satisfies

$$\inf_{\bar{\lambda}\geq 0} \mathcal{L}^{\textit{test}}_{\bar{\lambda},\textit{rep}}(\boldsymbol{A}_{\textit{good}}; \bar{n}_1) - \sigma^2 \leq 2\tau + \sigma^2 \frac{2k}{\bar{n}_1}$$

Thus under mild conditions on the noise in labels and size of train split during meta-training, this result shows that minimizers ($\tau = 0$) of $\mathcal{L}^{\text{tr-val}}_{\lambda,\text{rep}}$ will have meta-testing loss smaller than ϵ with $\bar{n}_1 = \mathcal{O}\left(\frac{k}{\epsilon}\right)$ samples per test task, even when $n_1 \neq \bar{n}_1$. It does so by learning a rank-k representation that spans $\boldsymbol{A}^*$. Compared to the result from Theorem 5.1, the bad representation $\boldsymbol{A}_{\text{bad}}$ learned using the $\mathcal{L}^{\text{tr-tr}}_{\lambda,\text{rep}}$ objective will need $\bar{n}_1 = \Omega\left(\frac{d}{\epsilon}\right)$ samples to get the same error. The above result also shows that getting close enough to optimality is also sufficient for good meta-testing performance; full version of the theorem and its proof can be found in Section C. We reiterate that we did not need any explicit regularization to show this sample complexity benefit. The proof sketch below highlights the implicit regularization in $\mathcal{L}^{\text{tr-val}}_{\lambda,\text{rep}}$ and discusses the role of label noise σ and number of train task samples n_1 in ensuring the desired low-rankness.

Proof sketch The crux of the proof is in a stronger result in Theorem C.1 that provides a *closed form expression* for $\mathcal{L}^{\text{tr-val}}_{\lambda,\text{rep}}(\boldsymbol{A}; (n_1, n_2))$. For any representation $\boldsymbol{A}$, let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$ be the singular value decomposition (SVD) with $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{d\times r}, \boldsymbol{S} \in \mathbb{R}^{r\times r}$ and $r = \text{rank}(\boldsymbol{A})$. The expression for $\mathcal{L}^{\text{tr-val}}_{\lambda,\text{rep}}(\boldsymbol{A}; (n_1, n_2))$ from Theorem C.1 is

$$\mathcal{L}^{\text{tr-val}}_{\lambda,\text{rep}}(\boldsymbol{A}) \approx \underbrace{\min\left\{1, \frac{n_1}{r}\right\} \|P^{\perp}_{\boldsymbol{A}}\boldsymbol{A}^*\|^2}_{\boldsymbol{\alpha}_1(\boldsymbol{A})} + \underbrace{\frac{(r-n_1)_+}{r}}_{\boldsymbol{\alpha}_2(\boldsymbol{A})}$$
$$+ \underbrace{\sigma^2\left[\mathbb{1}_{r>n_1+1}\frac{n_1}{r-n_1-1} + \mathbb{1}_{r<n_1-1}\frac{r}{n_1-r-1}\right]}_{\boldsymbol{\alpha}_3(\boldsymbol{A})}$$

where $P_{\boldsymbol{A}}$ is the projection matrix for the column span of $\boldsymbol{A}$ and $P^{\perp}_{\boldsymbol{A}} = I_d - P_{\boldsymbol{A}}$. We now look at how each term contributes to the final result.

1) The term $\boldsymbol{\alpha_1(A)}$ penalizes inexpressiveness of the representation through the $\|P^{\perp}_{\boldsymbol{A}}\boldsymbol{A}^*\|^2$ term. When $r \geq k$, this can be made 0 by letting $\boldsymbol{A}$ span all the k directions in $\boldsymbol{A}^*$; thus this term also encourages the rank being at least k.

2) The second term $\boldsymbol{\alpha_2(A)}$ can be as low as 0 only when the rank satisfies $r \leq n_1$. In the noiseless setting ($\sigma = 0$), $\mathcal{L}^{\text{tr-val}}_{\lambda,\text{rep}}$

already encourage learning an *expressive representation* with rank between k and n_1. Thus if the number of samples n_1 in the train split of train tasks is small, we are already guaranteed a low rank representation and new task sample complexity of $\mathcal{O}(n_1)$; thus it might be beneficial to use fewer samples in the train split and more in the validation split, similar to the result in (Bai et al., 2020).

3) Unlike $\sigma = 0$, $\sigma > 0$ can differentiate in $r \in [k, n_1]$ through the $\boldsymbol{\alpha_3(A)}$ term. Under the mild assumptions on σ and n_1, this term ensures that the loss is minimized exactly at $r = k$, and thus $\|P_{\boldsymbol{A}}^{\perp} \boldsymbol{A}^*\|^2 = 0$ from $\boldsymbol{\alpha_1(A)}$. Thus label noise helps learn low rank representations even for large n_1.

This leads to the first part of Theorem 5.4 showing that any minimizer $\boldsymbol{A}_{\text{good}}$ of the tr-val objective will be low-rank and expressive. The extension to τ-optimal solutions also follows the same strategy. The second part showing that such a representation $\boldsymbol{A}_{\text{good}}$ can reduce the sample complexity of a new test task follows by noting that $\boldsymbol{A}_{\text{good}}$ effectively projects inputs onto the k-dimensional subspace $\boldsymbol{A}^*$ and thus lets us use the upper bound on sample complexity for linear regression on k dimensions, which is $\mathcal{O}(\frac{k}{\epsilon})$.

The above result holds for any output dimensionality $D \geq k$ of representation $\boldsymbol{A} \in \mathbb{R}^{d \times D}$; the tr-val objective can automatically adapt and learn low rank and expressive representations, unlike the tr-tr objective. This is demonstrated in experiments on Omniglot and MiniImageNet in Section 6 where increasing dimensionality of representations improves the performance of tr-val representations, but the added expressivity hurts the tr-tr performance (cf. Tables 3 & 4). Theorem C.1 is proved using symmetry in Gaussian distribution and properties of Wishart distributions. We now make a closer comparison to prior work on linear representation learning and role of tr-val splitting.

5.3. Comparison to Prior Work and Discussions

Data splitting: Bai et al. (2020) analyze the tr-tr and tr-val objectives for linear centroid meta-learning, where the prediction function for parameters $\theta \in \mathbb{R}^D$ on input $\boldsymbol{x}$ is $f_\theta(\boldsymbol{x}) = \theta^\top \boldsymbol{x}$. Using random matrix theory, they show in the realizable setting that tr-tr objective learns parameters that have better new task sample complexity than tr-val parameters, by a constant factor, because it can use the data from train tasks more efficiently. For experiments they use iMAML (Rajeswaran et al., 2019) for tr-val method and MiniBatchProx (Zhou et al., 2019) for the tr-tr method and show that tr-tr outperforms tr-val on standard benchmarks. We note that these findings do not contradict ours, since MiniBatchProx does not minimize the vanilla tr-tr objective method but adds a regularization term. In Section 6 we verify that a tr-tr variant of iMAML without regularization indeed does poorly on the same dataset.

Table 1. Performance of meta-learning models trained using the tr-val and tr-tr objectives on Omniglot few-shot classification. RepLearn method is described in Section D and iMAML method is from (Rajeswaran et al., 2019)

		5-way 1-shot	20-way 1-shot
tr-val	RepLearn	97.30 ± 0.01	92.30 ± 0.01
	iMAML	97.90 ± 0.58	91.00 ± 0.54
tr-tr	RepLearn	89.00 ± 0.01	88.20 ± 0.01
	iMAML	49.20 ± 1.91	18.30 ± 0.67

Table 2. Performance of linear representations $\boldsymbol{A}$ on simulation dataset with $d = 50, k = 5, \sigma = 0.5$ (refer to Section 4.1). tr-tr and tr-val correspond to representations trained using $\widehat{\mathcal{L}}^{\text{tr-tr}}_{\lambda,\text{rep}}$ and $\widehat{\mathcal{L}}^{\text{tr-val}}_{\lambda,\text{rep}}$ respectively. Representations are evaluated on test tasks with $\bar{n}_1$ samples using $\mathcal{L}^{\text{test}}_{\bar{\lambda},\text{rep}}(\boldsymbol{A}; \bar{n}_1)$; ($\bar{\lambda}$ is tuned for all entries). Includes performance of $\boldsymbol{A} = I_d$ and $\boldsymbol{A} = \boldsymbol{A}^* \boldsymbol{A}^{*\top}$ that are the "worst" and "best" representations for this task respectively.

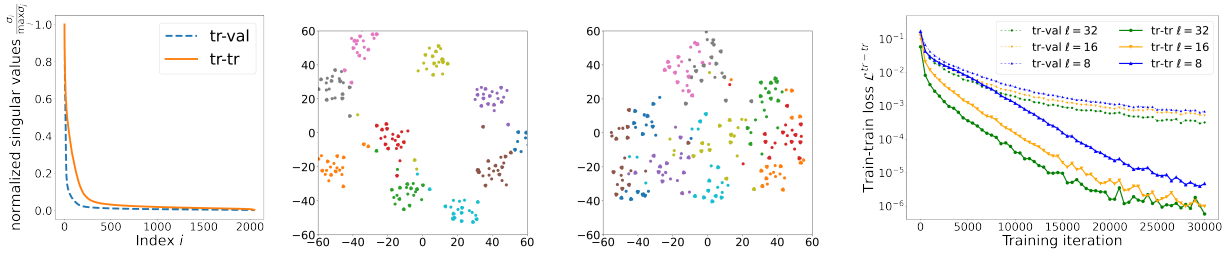
$\boldsymbol{A}$	$\mathcal{L}^{\text{test}}_{\bar{\lambda},\text{rep}}(\cdot; \bar{n}_1)$ $\bar{n}_1 = 5$	$\bar{n}_1 = 15$	$\bar{n}_1 = 25$
I_d	1.10 ± 0.09	0.99 ± 0.08	0.97 ± 0.17
tr-tr ($\lambda = 0$)	1.04 ± 0.08	0.85 ± 0.06	0.82 ± 0.06
tr-tr ($\lambda = 1$)	0.94 ± 0.07	0.69 ± 0.05	0.66 ± 0.05
tr-tr ($\lambda = 10$)	0.92 ± 0.07	0.69 ± 0.05	0.69 ± 0.05
tr-val ($\lambda = 0$)	0.72 ± 0.06	0.40 ± 0.03	0.38 ± 0.03
$\boldsymbol{A}^* \boldsymbol{A}^{*\top}$	0.70 ± 0.05	0.38 ± 0.03	0.36 ± 0.03

Linear representation learning: We have already discussed differences in settings from many linear representation learning guarantees: lack of data splitting and presence of regularization or rank constraints. Our work incorporates a practical choice of data splitting in linear representation learning theory. Another such result is from Saunshi et al. (2020) that employs a trajectory based analysis to show that Reptile and also gradient descent $\mathcal{L}^{\text{tr-tr}}$ learn good representations without explicit regularization or data splitting. This provides evidence that inductive biases of training algorithms is another way to avoid high rank representations. Their result is for the subspace meta-learning instance with $k = 1$ and $n_1 \to \infty$. While interesting, it is not clear how to extend it to the more interesting case of $n_1 = \mathcal{O}(k)$ that our analysis for $\mathcal{L}^{\text{tr-val}}_{\lambda,\text{rep}}$ can handle.

6. Experiments

In this section we present experimental results[8] that demonstrate the practical relevance of our theoretical results. For simulations we use the subspace learning meta-learning instance defined in Section 4.1 and verify that the tr-val objective is indeed superior to the tr-tr variant due to learning low-rank representation. We also perform experiments on the standard meta-learning benchmarks of Omniglot

[8]Code available at https://github.com/nsaunshi/meta_tr_val_split



(a) (b) (c) (d)

Figure 1. **(a)**: Normalized singular values for tr-val versus tr-tr. Singular value decay for tr-val is faster, indicating that it has lower effective rank than tr-tr. **(b,c)**: tSNE for tr-val and tr-tr representations respectively for inputs from 10 randomly selected test classes from Omniglot dataset. Dots with the same color correspond to points from the same class. We find that the tr-val representations are more clustered than the tr-tr representations. **(d)**: tr-tr loss plotted in log scale versus training iteration for tr-val versus tr-tr runs at varying width factor ℓ. This verifies that tr-tr models are indeed optimizing the loss that they were meant to minimize.

(Lake et al., 2015) and MiniImageNet (Vinyals et al., 2016) datasets, where we again find that representation learning benefits from tr-val splitting. Furthermore we show that iMAML (Rajeswaran et al., 2019), a non-representation learning meta-learning algorithm, also benefits from tr-val splitting. iMAML was also used by (Bai et al., 2020) to conclude the tr-tr is better than tr-val. We also perform a more detailed study of the learned representations on these benchmarks and find that while tr-val and tr-tr representations are almost equally expressive, tr-val representations have lower effective rank and thus have better few-shot performance. In Table 8 we verify that adding a Frobenius norm regularization on the representation with the tr-tr objective helps significantly, but is still slightly worse than tr-val splitting.

For our representation learning experiments, similar to ANIL algorithm (Raghu et al., 2020), we optimize the objective defined in Equation (5) that fixes the representation in the inner loop. Details of our first-order variant of ANIL (akin to first-order MAML (Finn et al., 2017)) are deferred to Section D.1; Algorithm 1 summarizes this RepLearn algorithm. Please refer to Section D for additional experiments.

Simulation: We simulate the meta-learning instance from Section 4.1 with $d = 50$, $k = 5$ and $\sigma = 0.5$. The tr-tr models are trained by minimizing $\widehat{\mathcal{L}}^{\text{tr-tr}}_{\lambda,\text{rep}}(\cdot; n)$ with $n = 16$ and tr-val models trained with $\widehat{\mathcal{L}}^{\text{tr-val}}_{\lambda,\text{rep}}(\cdot; (n_1, n_2))$ with $n_1 = 8, n_2 = 8$. We tune $\bar{\lambda}$ for before meta-evaluation using validation tasks for each model. For meta-testing we evaluate with $\bar{n}_1 \in \{5, 15, 25\}$ train samples per test task in Table 2. We find that the tr-val objective, trained with $\lambda = 0$ as in Theorem 5.4, outperforms all tr-tr models, and does almost as well as the correct projection $\boldsymbol{A}^* \boldsymbol{A}^{*\top}$. We notice that the tr-tr objective does not do as badly as $\boldsymbol{A} = I_d$, indicating that the training algorithm could have some useful inductive biases, but not enough. For the $\boldsymbol{A} \in \mathbb{R}^{50\times 50}$ learned using tr-val in the simulation experiment, the top $k = 5$ singular values explain 95% of its total norm, thus implying that it is almost rank k. We also check the alignment of the top-k singular directions of $\boldsymbol{A}$ with the optimal subspace and find the principle angle to be as small as 0.1°. For reference, the corresponding numbers for the best tr-tr learned $\boldsymbol{A}$ are 24% and 5° respectively.

RepLearn on Omniglot: We investigate the performance of tr-val and tr-tr objectives for representation learning on the Omniglot dataset for the 5-way 1-shot and 20-way 1-shot variants with a 4 layer convolution backbone as in (Finn et al., 2017); results presented in Table 1. Following the same protocol as (Bai et al., 2020), for meta-training we use $n = 2N$ for tr-tr objective and $n_1 = n_2 = N$ for the tr-val objective for the N-way 1-shot task. For meta-testing we use $\bar{n}_1 = N$. We find that tr-val outperforms tr-tr. Additional details are in Section D.

Capacity and tr-val versus tr-tr: We examine the performance gap between tr-val versus tr-tr as we increase the expressive power of the representation network. We use a baseline 4 hidden layer fully connected network (FCN) inspired by (Rajeswaran et al., 2019), but with 64ℓ nodes at all 4 layers for different values of $\ell \in \{1, 4, 8, 16, 32\}$. Using FCNs gets rid of inductive biases of CNNs and helps distill the role of data splitting better. The models are trained using tr-tr and tr-val representation learning objectives on the Omniglot 5-way 1-shot dataset; results presented in Table 3.. We find that increasing width tends to slightly improve the performance of tr-val models but hurts the performance of tr-tr models, thus increasing the gap between tr-val and tr-tr. Additionally, we note that the tr-tr model succeeds in minimizing the tr-tr loss $\mathcal{L}^{\text{tr-tr}}$ as evident in Figure 1d. Thus its bad performance on meta-learning is not an optimization issue, but can be attributed to learning overly expressive representations due to lack of explicit regularization, as predicted by Theorem 5.1. This demonstrates that the tr-val objective is more robust to choice of model architecture.

Low-rankness and expressivity: We investigate the representations from the trained FCNs with $\ell = 32$. To ascertain expressivity of representations, we combine all data from

Table 3. Accuracies in % of representations parameterized by fully connected networks of varying widths on Omniglot 5-way 1-shot meta-testing and on a supervised dataset that is constructed using the 413 test classes from Omniglot. Representations trained using tr-val objective consistently outperforms those learned using tr-tr objective, and the gap increases as width increases.

width $= 64^*\ell$	Omniglot 5-way 1-shot		Supervised 413-way	
	tr-val	tr-tr	tr-val	tr-tr
$\ell = 1$	87.8 ± 1.1	80.6 ± 1.4	87.4	73.4
$\ell = 4$	90.8 ± 1.0	77.8 ± 1.4	100.0	100.0
$\ell = 8$	91.6 ± 1.0	73.9 ± 1.5	100.0	100.0
$\ell = 16$	91.7 ± 0.9	70.6 ± 1.6	100.0	100.0
$\ell = 32$	91.6 ± 1.0	67.8 ± 1.6	100.0	100.0

the test classes of Omniglot to produce a supervised learning dataset with 413 classes. We measure expressivity of representations by evaluating its linear classification performance on the 413-way supervised task; results in Table 3. We find that the tr-val representations do at least as well as tr-tr ones and thus are expressive enough. To compare the effective ranks, we plot the normalized singular values of both sets of representations in Figure 1a and find that the tr-val representations have a steeper drop in singular values than the tr-tr representations, thus confirming their lower effective rank. t-SNE plots (Figure 1b and Figure 1c) on the tr-val and tr-tr representations for data points from 10 randomly selected classes suggest that tr-val representations are better clustered than the tr-tr representations.

iMAML on Omniglot: We go beyond representation learning and consider a meta-learning method iMAML (Rajeswaran et al., 2019) that updates all parameters in the inner loop. We modify the authors code[9] to implement a tr-tr version and we investigate the performance of tr-val and tr-tr objectives on the Omniglot dataset for both 5-way 1-shot and 20-way 1-shot settings in Table 1. We find that tr-val outperforms tr-tr again, this time by a significant margin. We do not tune hyper-parameters for tr-val and just use the defaults, but for tr-tr we tune them to give it an edge.

RepLearn on MiniImageNet: We performs similar experiments on MiniImageNet using a standard CNN backbone with 32, 64, 128, and 128 filters, respectively. Table 5 shows again that RepLearn with tr-val splitting is much better than tr-tr on 5-way 1-shot and 5-way 5-shot settings, similar to the findings with Omniglot. We also perform the capacity experiment by increasing the number of filters by a factor ℓ; the convolutional layers contain 32ℓ, 64ℓ, 128ℓ, and 128ℓ output filters, respectively. Results in Table 4 suggest that increasing the network capacity improves the performance of tr-val representations, but slightly hurts tr-tr performance, just like the findings for Omniglot dataset with fully-connected networks. The drop might be lower

[9] https://github.com/aravindr93/imaml_dev

Table 4. Accuracies in % of representations parameterized by CNN networks with varying number of filters on MiniImageNet. Representations trained using tr-val objective consistently outperforms those learned using tr-tr objective; gap increases as width increases.

capacity $= num\_filters^*\ell$	MiniImageNet 5-way 1-shot	
	tr-val	tr-tr
$\ell = 0.5$	46.66 ± 1.69	26.25 ± 1.45
$\ell = 1$	48.44 ± 1.62	26.81 ± 1.44
$\ell = 4$	$52.22 \pm 1.68.$	24.66 ± 1.26
$\ell = 8$	52.25 ± 1.71	25.28 ± 1.37

Table 5. Tr-val v/s tr-tr meta-test accuracies in % for a CNN model trained with RepLearn on MiniImageNet 5-way 1-shot and 5-way 5-shot for varying values of the regularization parameter, λ.

	5-way 1-shot	5-way 5-shot
$\lambda = 0.0$ tr-val	46.16 ± 1.67	65.36 ± 0.91
$\lambda = 0.0$ tr-tr	25.53 ± 1.43	33.49 ± 0.82
$\lambda = 0.1$ tr-tr	24.69 ± 1.32	34.91 ± 0.85
$\lambda = 1.0$ tr-tr	25.88 ± 1.45	40.19 ± 1.12

here due to a CNN being used instead of fully connected network in Table 3. Thus the tr-val method is more robust to architecture choice/capacity and datasets.

7. Discussions and Future Work

We study the implicit regularization effect of the practical design choice of train-validation splitting popular in meta-learning, and show that it encourages learning low-rank but expressive enough representations. This is contrasted with the non-splitting variant that is shown to fail without explicit regularization. Both of these claims are justified theoretically for linear representation learning and experimentally for standard meta-learning benchmarks. Train-validation splitting provides a new mechanism for sample efficiency through implicit regularization in the objective, as opposed to explicit regularization and implicit bias of training algorithm, as discussed in Section 5.3. We show learning of exact low rank representations in our setting as opposed to approximate low-rankness observed in practice. Relaxing assumptions of Gaussianity, common input distribution across tasks and linearity of representations might explain the observed *effective* low-rankness. Finally an interesting problem is to get the best of all worlds, data efficiency from tr-tr style objective, explicit regularization and the implicit low rank regularization from tr-val splitting in a principled way. Identifying and understanding other training paradigms that intrinsically use data efficiently, even without explicit regularization is also an interesting direction.

Acknowlegments: We thank Mikhail Khodak for comments on an earlier draft of the paper. Nikunj Saunshi, Arushi Gupta and Wei Hu are supported by NSF, ONR, Simons Foundation, Amazon Research, DARPA and SRC.

References

Ahn, W.-K. and Brewer, W. F. Psychological studies of explanation—based learning. *Investigating explanation-based learning*, 1993.

Alquier, P., Mai, T. T., and Pontil, M. Regret bounds for lifelong learning. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, 2016.

Argyriou, A., Evgeniou, T., and Pontil, M. Convex multi-task feature learning. *Machine learning*, 2008.

Arnold, S. M., Iqbal, S., and Sha, F. When maml can adapt fast and how to assist when it cannot. *arXiv preprint arXiv:1910.13603*, 2019.

Bai, Y., Chen, M., Zhou, P., Zhao, T., Lee, J. D., Kakade, S., Wang, H., and Xiong, C. How important is the train-validation split in meta-learning? *arXiv preprint arXiv:2010.05843*, 2020.

Baxter, J. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 2000.

Belkin, M., Hsu, D., and Xu, J. Two models of double descent for weak features. *SIAM Journal on Mathematics of Data Science*, 2020.

Bengio, Y., Bengio, S., and Cloutier, J. *Learning a synaptic learning rule*. Citeseer, 1990.

Bertinetto, L., Henriques, J. F., Torr, P., and Vedaldi, A. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

Bullins, B., Hazan, E., Kalai, A., and Livni, R. Generalize across tasks: Efficient algorithms for linear representation learning. In *Proceedings of the 30th International Conference on Algorithmic Learning Theory*, 2019.

Caruana, R. Multitask learning. *Machine learning*, 28, 1997.

Collins, L., Mokhtari, A., and Shakkottai, S. Why does maml outperform erm? an optimization perspective. *arXiv preprint arXiv:2010.14672*, 2020.

Deleu, T., Würfl, T., Samiei, M., Cohen, J. P., and Bengio, Y. Torchmeta: A Meta-Learning library for PyTorch, 2019. Available at: https://github.com/tristandeleu/pytorch-meta.

Denevi, G., Ciliberto, C., Stamos, D., and Pontil, M. Incremental learning-to-learn with statistical guarantees. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2018a.

Denevi, G., Ciliberto, C., Stamos, D., and Pontil, M. Learning to learning around a common mean. In *Advances in Neural Information Processing Systems*, 2018b.

Du, S. S., Hu, W., Kakade, S. M., Lee, J. D., and Lei, Q. Few-shot learning via learning the representation, provably. *arXiv preprint arXiv:2002.09434*, 2020.

Evgeniou, T. and Pontil, M. Regularized multi-task learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

Fallah, A., Mokhtari, A., and Ozdaglar, A. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Goldblum, M., Reich, S., Fowl, L., Ni, R., Cherepanova, V., and Goldstein, T. Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Gu, J., Wang, Y., Chen, Y., Li, V. O. K., and Cho, K. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Khodak, M., Balcan, M.-F., and Talwalkar, A. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, 2019a.

Khodak, M., Balcan, M.-F., and Talwalkar, A. Provable guarantees for gradient-based meta-learning. In *Proceedings of the 36th International Conference on Machine Learning*, 2019b.

Kong, W., Somani, R., Song, Z., Kakade, S., and Oh, S. Meta-learning for mixed linear regression. In *International Conference on Machine Learning*, 2020.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 2015.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Mardia, K., Kent, J., and Bibby, J. Multivariate analysis, 1979. *Probability and mathematical statistics. Academic Press Inc*, 1979.

Maurer, A. Algorithmic stability and meta-learning. *Journal of Machine Learning Research*, 6, 2005.

Maurer, A. Transfer bounds for linear feature learning. *Machine Learning*, 2009.

Maurer, A. and Pontil, M. Excess risk bounds for multitask learning with trace norm regularization. In *Proceedings of the 26th Annual Conference on Learning Theory*, 2013.

Maurer, A., Pontil, M., and Romera-Paredes, B. The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17, 2016.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Aguera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artifical Intelligence and Statistics*, 2017.

Naik, D. K. and Mammone, R. J. Meta-neural networks that learn by learning. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*. IEEE, 1992.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Oh, J., Yoo, H., Kim, C., and Yun, S.-Y. Does maml really want feature reuse only? *arXiv preprint arXiv:2008.08882*, 2020.

Pentina, A. and Lampert, C. H. A PAC-Bayesian bound for lifelong learning. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, 2019.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

Ruvolo, P. and Eaton, E. ELLA: An efficient lifelong learning algorithm. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

Saunshi, N., Zhang, Y., Khodak, M., and Arora, S. A sample complexity separation between non-convex and convex meta-learning. In *International Conference on Machine Learning*, 2020.

Schmidhuber, J. *Evolutionary principles in self-referential learning, or on learning how to learn*. PhD thesis, Technische Universität München, 1987.

Snell, J., Swersky, K., and Zemel, R. S. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

Thrun, S. and Pratt, L. *Learning to Learn*. Springer Science & Business Media, 1998.

Tripuraneni, N., Jin, C., and Jordan, M. I. Provable meta-learning of linear representations. *arXiv preprint arXiv:2002.11684*, 2020a.

Tripuraneni, N., Jordan, M. I., and Jin, C. On the theory of transfer learning: The importance of task diversity. 2020b.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

Wang, H., Sun, R., and Li, B. Global convergence and induced kernels of gradient-based meta-learning with neural nets. *arXiv preprint arXiv:2006.14606*, 2020a.

Wang, X., Yuan, S., Wu, C., and Ge, R. Guarantees for tuning the step size using a learning-to-learn approach. *arXiv preprint arXiv:2006.16495*, 2020b.

Zhou, P., Yuan, X., Xu, H., Yan, S., and Feng, J. Efficient meta learning via minibatch proximal update. In *Advances in Neural Information Processing Systems*, 2019.