A Simple Recipe for Language-guided Domain Generalized Segmentation

Mohammad Fahes[1] Tuan-Hung Vu[1,2] Andrei Bursuc[1,2] Patrick Pérez[3] Raoul de Charette[1]
[1] Inria [2] Valeo.ai [3] Kyutai

https://astra-vision.github.io/FAMix

Abstract

Generalization to new domains not seen during training is one of the long-standing challenges in deploying neural networks in real-world applications. Existing generalization techniques either necessitate external images for augmentation, and/or aim at learning invariant representations by imposing various alignment constraints. Large-scale pretraining has recently shown promising generalization capabilities, along with the potential of binding different modalities. For instance, the advent of vision-language models like CLIP has opened the doorway for vision models to exploit the textual modality. In this paper, we introduce a simple framework for generalizing semantic segmentation networks by employing language as the source of randomization. Our recipe comprises three key ingredients: (i) the preservation of the intrinsic CLIP robustness through minimal fine-tuning, (ii) language-driven local style augmentation, and (iii) randomization by locally mixing the source and augmented styles during training. Extensive experiments report state-of-the-art results on various generalization benchmarks. Code is accessible on the project page[1].

1. Introduction

A prominent challenge associated with deep neural networks is their constrained capacity to generalize when confronted with shifts in data distribution. This limitation is rooted in the assumption of data being independent and identically distributed, a presumption that frequently proves unrealistic in real-world scenarios. For instance, in safety-critical applications like autonomous driving, it is imperative for a segmentation model to exhibit resilient generalization capabilities when dealing with alterations in lighting, variations in weather conditions, and shifts in geographic location, among other considerations.

To address this challenge, domain adaptation [13, 18, 33, 34, 47, 48] has emerged; its core principle revolves around aligning the distributions of both the source and target domains. However, DA hinges on having access to target data, which may not always be available. Even when accessible, this data might not encompass the full spectrum of distributions encountered in diverse real-world scenarios. Domain generalization [31, 32, 49, 52, 62, 63] overcomes this limitation by enhancing the robustness of models to arbitrary and previously unseen domains.

The training of segmentation networks is often backed by large-scale pretraining as initialization for the feature representation. Until now, to the best of our knowledge, domain generalization for semantic segmentation (DGSS) networks [7, 19, 23, 24, 29, 36, 37, 51, 53, 58] are pretrained with ImageNet [9]. The underlying concept is to *transfer* the representations from the upstream task of classification to the downstream task of segmentation.

Lately, contrastive language image pretraining (CLIP) [22, 39, 55, 56] has demonstrated that transferable visual representations could be learned from the sole supervision of loose natural language descriptions at *very* large scale. Subsequently, a plethora of applications have been proposed using CLIP [39], including zero-shot semantic segmentation [30, 59], image editing [27], transfer learning [10, 40], open-vocabulary object detection [16], few-shot learning [64, 65] etc. A recent line of research proposes fine-tuning techniques to preserve the robustness of CLIP under distribution shift [15, 26, 45, 50], but they are limited to classification.

In this paper, we aim at answering the following question: *How to leverage CLIP pretraining for enhanced domain generalization for semantic segmentation?* The motivation for rethinking DGSS with CLIP is twofold. On one hand, distribution robustness is a notable characteristic of CLIP [12]. On the other hand, the language modality offers an extra source of information compared to unimodal pretrained models.

A direct comparison of training two segmentation models under identical conditions but with different pretraining, *i.e.* ImageNet *vs.* CLIP, shows that CLIP pretraining does not yield promising results. Indeed, Tab. 1 shows that fine-tuning CLIP-initialized network performs worse than

[1] https://astra-vision.github.io/FAMix

Pretraining	C	B	M	S	AN	AS	AR	AF	Mean
ImageNet	**29.04**	**32.17**	**34.26**	**29.87**	**4.36**	**22.38**	**28.34**	**26.76**	**25.90**
CLIP	16.81	16.31	17.80	27.10	2.95	8.58	14.35	13.61	14.69

Table 1. **Comparison of ImageNet and CLIP pretraining for out-of-distribution semantic segmentation.** The network is DeepLabv3+ with ResNet-50 as backbone. The models are trained on GTAV and the performance (mIoU %) is reported on Cityscapes (C), BDD-100K (B), Mapillary (M), Synthia (S), and ACDC Night (AN), Snow (AS), Rain (AR) and Fog (AF).

its ImageNet counterpart on out-of-distribution (OOD) data. This raises doubts about the suitability of CLIP pretraining for DGSS and indicates that it is more prone to *overfitting* the source distribution at the expense of degrading its original distributional robustness properties. Note that both models converge and achieve similar results on in-domain data. More details are provided in Appendix A.

This paper shows that we can prevent such behavior with a simple recipe involving minimal fine-tuning, language-driven style augmentation, and mixing. Our approach is coined FAMix, for *Freeze, Augment and Mix*.

It was recently argued that fine-tuning might distort the pretrained representations and negatively affect OOD generalization [26]. To maintain the integrity of the representation, one extreme approach is to entirely freeze the backbone. However, this can undermine representation adaptability and lead to subpar OOD generalization. As a middle-ground strategy balancing adaptation and feature preservation, we suggest minimal fine-tuning of the backbone, where a substantial portion remains frozen, and only the final layers undergo fine-tuning.

For generalization, we show that rethinking MixStyle [62] leads to significant performance gains. As illustrated in Fig. 1, we mix the statistics of the original source features with augmented statistics mined using language. This helps explore styles beyond the source distribution at training time without using additional image.

We summarize our contributions as follows:

- We propose a simple framework for DGSS based on minimal fine-tuning of the backbone and language-driven style augmentation. To the best of our knowledge, we are the first to study DGSS with CLIP pretraining.
- We propose language-driven class-wise local style augmentation. We mine class-specific local statistics using prompts that express random styles and names of patch-wise dominant classes. During training, randomization is performed through patch-wise style mixing of the source and mined styles.
- We conduct careful ablations to show the effectiveness of FAMix. Our framework outperforms state-of-the-art approaches in single and multi-source DGSS settings.

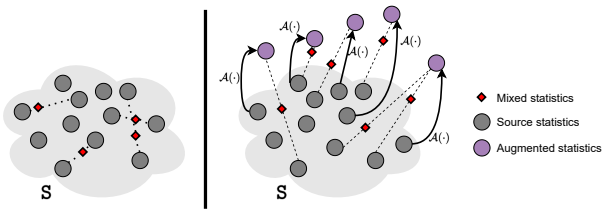


Figure 1. **Mixing strategies.** (*Left*) MixStyle [62] consists of a linear mixing between the feature statistics of the source domain(s) **S** samples. (*Right*) We apply an augmentation $\mathcal{A}(.)$ on the source domain statistics, then perform linear mixing between original and augmented statistics. Intuitively, this enlarges the support of the training distribution by leveraging statistics beyond the source domain(s), as well as discovering intermediate domains. $\mathcal{A}(.)$ could be a language-driven or Gaussian noise augmentation, and we show that the former leads to better generalization results.

2. Related works

Domain generalization (DG). The goal of DG is to train, from a single or multiple source domains, models that perform well under arbitrary domain shifts. The DG literature spans a broad range of approaches, including adversarial learning [32, 57], meta-learning [4, 38], data augmentation [60–62] and domain-invariant representation learning [1, 3, 7, 25]. We refer the reader to [49, 63] for comprehensive surveys on DG.

Domain generalization with CLIP. CLIP [39] exhibits a remarkable distributional robustness [12]. Nevertheless, fine-tuning comes at the expense of sacrificing generalization. Kumar *et al.* [26] observe that full fine-tuning can distort the pretrained representation, and propose a two-stage strategy, consisting of training a linear probe with a frozen feature extractor, then fine-tuning both. Wortsman *et al.* [50] propose ensembling the weights of zero-shot and fine-tuned models. Goyal *et al.* [15] show that preserving the pretraining paradigm (*i.e.* contrastive learning) during the adaptation to the downstream task improves both in-domain (ID) and OOD performance without multi-step fine-tuning or weight ensembling. CLIPood [45] introduces margin metric softmax training objective and Beta moving average for optimization to handle both open-class and open-domain at test time. On the other hand, distributional robustness could be improved by training a small amount of parameters on top of a frozen CLIP backbone in a teacher-student manner [21, 28]. Other works show that specialized prompt ensembling and/or image ensembling strategies [2, 14] coupled with label augmentation using the WordNet hierarchy improve robustness in classification.

Domain Generalized Semantic Segmentation. DGSS methods could be categorized into three main groups: normalization methods, domain randomization (DR) and in-

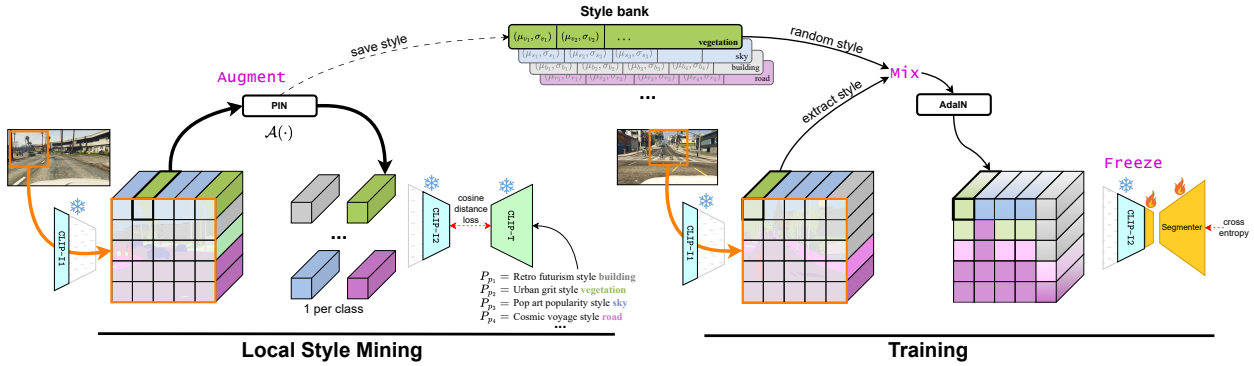


Figure 2. **Overall process of FAMix.** FAMix consists of two steps. (*Left*) *Local style mining* consists of dividing the low-level feature activations into patches, which are used for style mining using Prompt-driven Instance Normalization (PIN) [10]. Specifically, for each patch, the dominant class is queried from the ground truth, and the mined style is added to corresponding class-specific style bank. (*Right*) *Training* the segmentation network is performed with minimal fine-tuning of the backbone. At each iteration, the low-level feature activations are viewed as grids of patches. For each patch, the dominant class is queried using the ground truth, then a style is sampled from the corresponding style bank. Style randomization is performed by normalizing each patch in the grid by its statistics, and transferring the new style which is a mixing between the original style and the sampled one. The network is trained using only a cross-entropy loss.

variant representation learning. Normalization methods aim at removing style contribution from the representation. For instance, IBN-Net [36] shows that Instance Normalization (IN) makes the representation invariant to variations in the scene appearance (e.g., change of colors, illumination, etc.), and that combining IN and batch normalization (BN) helps the synthetic-to-real generalization. SAN & SAW [37] proposes semantic-aware feature normalization and whitening, while RobustNet [7] proposes an instance selective whitening loss, where only feature covariances that are sensitive to photometric transformations are whitened. DR aims instead at diversifying the data during training. Some methods use additional data for DR. For example, WildNet [29] uses ImageNet [9] data for content and style extension learning, while TLDR [24] proposes learning texture from random style images. Other methods like SiamDoGe [51] perform DR solely by data augmentation, using a Siamese [6] structure. Finally in the invariant representation learning group, SPC-Net [19] builds a representation space based on style and semantic projection and clustering, and SHADE [58] regularizes the training with a style consistency loss and a retrospection consistency loss.

3. Method

FAMix proposes an effective recipe for DGSS through the blending of simple ingredients. It consists of two stages (see Fig. 2): (i) Local style mining from language (Sec. 3.2); (ii) Training of a segmentation network with minimal fine-tuning and local style mixing (Sec. 3.3). In Fig. 2 and in the following, `CLIP-I1` denotes the *stem layers* and *Layer1* of CLIP image encoder, `CLIP-I2` the remaining layers excluding the attention pooling, and `CLIP-T` the text encoder.

We start with some preliminary background knowledge, introducing AdaIN and PIN which are essential to our work.

3.1. Preliminaries

Adaptive Instance Normalization (AdaIN). For a feature map $\mathbf{f} \in \mathbb{R}^{h \times w \times c}$, AdaIN [20] shows that the channel-wise mean $\boldsymbol{\mu} \in \mathbb{R}^c$ and standard deviation $\boldsymbol{\sigma} \in \mathbb{R}^c$ capture information about the style of the input image, allowing style transfer between images. Hence, stylizing a source feature $\mathbf{f}_\text{s}$ with an arbitrary target style $(\mu(\mathbf{f}_\text{t}), \sigma(\mathbf{f}_\text{t}))$ reads:

$$\texttt{AdaIN}(\mathbf{f}_\text{s}, \mathbf{f}_\text{t}) = \sigma(\mathbf{f}_\text{t})\Big(\frac{\mathbf{f}_\text{s} - \mu(\mathbf{f}_\text{s})}{\sigma(\mathbf{f}_\text{s})}\Big) + \mu(\mathbf{f}_\text{t}), \tag{1}$$

with $\mu(\cdot)$ and $\sigma(\cdot)$ the mean and standard deviation of input feature; multiplications and additions being element-wise.

Prompt-driven Instance Normalization (PIN). PIN was introduced for prompt-driven zero-shot domain adaptation in PØDA [10]. It replaces the target style $(\mu(\mathbf{f}_\text{t}), \sigma(\mathbf{f}_\text{t}))$ in AdaIN (1) with two optimizable variables $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ guided by a single prompt in natural language. The rationale is to leverage a frozen CLIP [39] to mine visual styles from the prompt representation in the shared space. Given a prompt P and a feature map $\mathbf{f}_\text{s}$, PIN reads as:

$$\texttt{PIN}_{(P)}(\mathbf{f}_\text{s}) = \boldsymbol{\sigma}\Big(\frac{\mathbf{f}_\text{s} - \mu(\mathbf{f}_\text{s})}{\sigma(\mathbf{f}_\text{s})}\Big) + \boldsymbol{\mu}, \tag{2}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are optimized using gradient descent, such that the cosine distance between the visual feature representation and the prompt representation is minimized.

Different from PØDA which mines styles globally with a predetermined prompt describing the target domain, we make use of PIN to mine class-specific styles using local patches of the features, leveraging *random style prompts*.

Further, we show the effectiveness of incorporating the class name in the prompt for better style mining.

3.2. Local Style Mining

Our approach is to leverage PIN to mine class-specific style banks that used for feature augmentation when training FAMix. Given a set of cropped images $\mathcal{I}_s$, we encode them using `CLIP-I1` to get a set of low-level features $\mathcal{F}_s$. Each batch b of features $\mathbf{f}_s \in \mathcal{F}_s$ is cropped into m patches, resulting in $b \times m$ patches $\mathbf{f}_p$, associated ground-truth annotation $\mathbf{y}_p$, of size $^h/\sqrt{m} \times {}^w/\sqrt{m} \times c$.

We aim at populating K style banks, K being the total number of classes. For a feature patch $\mathbf{f}_p$, we compute the dominant class from the corresponding label patch $\mathbf{y}_p$, and get its name t_p from the predefined classes in the training dataset. Given a set of prompts describing random styles $\mathcal{R}$, the target prompt P_p is formed by concatenating a randomly sampled style prompt r from $\mathcal{R}$ and t_p (*e.g.*, `retro futurism style building`). We show in the experiments (Sec. 4.4) that our method is not very sensitive to the prompt design, yet our prompt construction works best.

The idea is to mine *proxy* domains and explore intermediate ones in a class-aware manner (as detailed in Sec. 3.3), which makes our work fundamentally different from [10], that steers features towards a particular target style and corresponding domain, and better suited to generalization.

To handle the class imbalance problem, we simply select one feature patch $\mathbf{f}_p$ per class among the total $b \times m$ patches, as shown in Fig. 2. Consequently, we apply PIN (2) to optimize the local styles to match the representations of their corresponding prompts, and use the mined styles to populate the corresponding style banks. The complete procedure is outlined in Algorithm 1.

The resulting style banks $\{\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(K)}\}$ are used for *domain randomization* during training.

3.3. Training FAMix

Style randomization. During training, randomly cropped images $\mathcal{I}_s$ are encoded into $\mathbf{f}_s$ using `CLIP-I1`. Each batch of feature maps $\mathbf{f}_s$ is viewed as a grid of m patches, without cropping them. For each patch $\mathbf{f}_s^{(ij)}$ within the grid, the dominant class $c_p{}^{(ij)}$ is queried using the corresponding ground truth patch $\mathbf{y}_s{}^{(ij)}$, and a style is randomly sampled from the corresponding mined bank $\mathcal{T}^{(c_p{}^{(ij)})}$. We then apply patch-wise convex combination (*i.e.*, style mixing) of the original style of the patch and the mined style. Specifically, for an arbitrary patch $\mathbf{f}_s^{(ij)}$, our local style mixing reads:

$$\mu_{mix} \leftarrow (1-\alpha)\mu(\mathbf{f}_s^{(ij)}) + \alpha\boldsymbol{\mu}^{(ij)} \tag{3}$$

$$\sigma_{mix} \leftarrow (1-\alpha)\sigma(\mathbf{f}_s^{(ij)}) + \alpha\boldsymbol{\sigma}^{(ij)}, \tag{4}$$

with $(\boldsymbol{\mu}^{(ij)}, \boldsymbol{\sigma}^{(ij)}) \in \mathcal{T}^{(c_p^{(ij)})}$ and $\alpha \in [0,1]^c$.

Algorithm 1: Local Style Mining.

Input : Set $\mathcal{F}_s$ of source features batches.
Label set $\mathcal{Y}_s$ in $\mathcal{D}_s$.
Set of random prompts $\mathcal{R}$ and class names $\mathcal{C}$.
Param : Number of patches m.
Number of classes K.
Output: K sets $\{\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(K)}\}$ of class-wise augmented statistics.

$\{\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(K)}\} \leftarrow \emptyset$
foreach $(\mathbf{f}_s \in \mathcal{F}_s, \mathbf{y}_s \in \mathcal{Y}_s)$ **do**
  $\{\mathbf{y}_p\} \leftarrow$ `crop-patch`$(\mathbf{y}_s, m)$
  $\{c_p\}, \{P_p\}, \{f_p\} \leftarrow \emptyset$
  foreach $\mathbf{y}_p \in \{\mathbf{y}_p\}$ **do**
    $c_p \leftarrow$ `get-dominant-class`$(\mathbf{y}_p)$
    if c_p *not in* $\{c_p\}$ **then**
      $\{c_p\} \leftarrow c_p$
      $\{P_p\} \leftarrow$ `concat`$($`sample`$(\mathcal{R}),$ `get-name`$(c_p))$
      $\{f_p\} \leftarrow f_p$
    end
  end
  $\boldsymbol{\mu}^{(c_p)}, \boldsymbol{\sigma}^{(c_p)}, \mathbf{f}'_p \leftarrow \texttt{PIN}_{(P_p)}(\mathbf{f}_p)$
  $\mathcal{T}^{(c_p)} \leftarrow \mathcal{T}^{(c_p)} \cup \{(\boldsymbol{\mu}^{(c_p)}, \boldsymbol{\sigma}^{(c_p)})\}$
end

Algorithm 2: Training FAMix.

Input : Set $\mathcal{F}_s$ of source features batches.
Label set $\mathcal{Y}_s$ in $\mathcal{D}_s$.
K sets $\{\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(K)}\}$ of class-wise augmented statistics.
Param: Number of patches m.

foreach $(\mathbf{f}_s \in \mathcal{F}_s, \mathbf{y}_s \in \mathcal{Y}_s)$ **do**
  $\alpha \sim$ `Beta`$(0.1, 0.1)$
  for $(i,j) \in [1, \sqrt{m}] \times [1, \sqrt{m}]$ **do**
    $c_p^{(ij)} \leftarrow$ `get-dominant-class`$(\mathbf{y}_s^{(ij)})$
    $\boldsymbol{\mu}^{(ij)}, \boldsymbol{\sigma}^{(ij)} \leftarrow$ `sample`$(\mathcal{T}^{(c_p^{(ij)})})$
    $\mu_{mix} \leftarrow (1-\alpha).\mu(\mathbf{f}_s^{(ij)}) + \alpha.\boldsymbol{\mu}^{(ij)}$
    $\sigma_{mix} \leftarrow (1-\alpha).\sigma(\mathbf{f}_s^{(ij)}) + \alpha.\boldsymbol{\sigma}^{(ij)}$
    $\mathbf{f}_s^{(ij)} \leftarrow$ `AdaIN`$(\mathbf{f}_s^{(ij)}, \mu_{mix}, \sigma_{mix})$
  end
  $\tilde{\mathbf{y}}_s \leftarrow$ `CLIP-I2`$(\mathbf{f}_s)$
  `Loss` $=$ `cross-entropy`$(\tilde{\mathbf{y}}_s, \mathbf{y}_s)$
end

As shown in Fig. 1, our style mixing strategy differs from [62] which applies a linear interpolation between styles extracted from the images of a limited set of *source* domain(s) assumed to be available for training. Here, we view the mined styles as variations of multiple *proxy* target domains defined by the prompts. Training is conducted over all the paths in the feature space between the source and proxy domains without requiring any additional image during training other than the one from source.

Method	arch.	C	B	M	S	AN	AS	AR	AF	Mean
RobustNet [7]	RN50	36.58	35.20	40.33	28.30	6.32	29.97	33.02	32.56	30.29
SAN & SAW [37]		39.75	37.34	41.86	30.79	-	-	-	-	-
Pin the memory [23]		41.00	34.60	37.40	27.08	3.84	5.51	5.89	7.27	20.32
SHADE [58]		44.65	39.28	43.34	28.41	8.18	30.38	35.44	36.87	33.32
SiamDoGe [51]		42.96	37.54	40.64	28.34	10.60	30.71	35.84	36.45	32.89
DPCL [53]		44.87	40.21	46.74	-	-	-	-	-	-
SPC-Net [19]		44.10	40.46	45.51	-	-	-	-	-	-
NP [11]		40.62	35.56	38.92	27.65	-	-	-	-	-
WildNet* [29]		44.62	38.42	46.09	31.34	8.27	30.29	36.32	35.39	33.84
TLDR* [24]		46.51	42.58	46.18	30.57	13.13	36.02	**38.89**	**40.58**	36.81
FAMix (ours)		**48.15**	**45.61**	**52.11**	**34.23**	**14.96**	**37.09**	38.66	40.25	**38.88**
SAN & SAW [37]	RN101	45.33	41.18	40.77	31.84	-	-	-	-	-
SHADE† [58]		46.66	43.66	45.50	31.58	7.58	32.48	36.90	36.69	35.13
WildNet* [29]		45.79	41.73	47.08	32.51	-	-	-	-	-
TLDR* [24]		47.58	44.88	48.80	33.14	-	-	-	-	-
FAMix (ours)		**49.47**	**46.40**	**51.97**	**36.72**	**19.89**	**41.38**	**40.91**	**42.15**	**41.11**

Table 2. **Single-source DGSS trained on G.** Performance (mIoU %) of FAMix compared to other DGSS methods trained on G and evaluated on C, S, M, S, A for ResNet-50 ('RN50') and ResNet-101 ('RN101') backbone architecture ('arch.'). * indicates the use of extra-data. † indicates the use of the full data for training. We emphasize **best** and second best results.

Style transfer is applied through AdaIN (1). Only the standard cross-entropy loss between the ground truth $\mathbf{y}_s$ and the prediction $\tilde{\mathbf{y}}_s$ is applied for training the network. Algorithm 2 shows the training steps of FAMix.

Minimal fine-tuning. During training, we fine-tune only the last few layers of the backbone. Subsequently, we examine various alternatives and show that the minimal extent of fine-tuning is the crucial factor in witnessing the effectiveness of our local style mixing strategy.

Previous works [11, 36, 62] suggest that shallow feature statistics capture style information while deeper features encode semantic content. Consequently, some DGSS methods focus on learning style-agnostic representations [7, 36, 37], but this can compromise the expressiveness of the representation and suppress content information. In contrast, our intuition is to retain these identified traits by introducing variability to the shallow features through augmentation and mixing. Simultaneously, we guide the network to learn invariant high-level representations by training the final layers of the backbone with a label-preserving assumption, using a standard cross-entropy loss.

4. Experiments

4.1. Experimental setup

Synthetic datasets. GTAV [41] and SYNTHIA [42] are used as synthetic datasets. GTAV consists of 24 966 images split into 12 403 images for training, 6 382 for validation and 6 181 for testing. SYNTHIA consists of 9 400 images: 6 580 for training and 2 820 for validation. GTAV and SYNTHIA are denoted by G and S, respectively.

Real datasets. Cityscapes [8], BDD-100K [54], and Mapillary [35] contain 2 975, 7 000, and 18 000 images for training and 500, 1 000, and 2 000 images for validation, respectively. ACDC [44] is a dataset of driving scenes in adverse conditions: night, snow, rain and fog with respectively 106, 100, 100 and 100 images in the validation sets. C, B, and M denote Cityscapes, BDD-100K and Mapillary, respectively; AN, AS, AR and AF denote night, snow, rain and fog subsets of ACDC, respectively.

Implementation details. Following previous works [7, 19, 23, 24, 29, 37, 51, 53, 58], we adopt DeepLabv3+ [5] as segmentation model. ResNet-50 and ResNet-101 [17], initialized with CLIP pretrained weights, are used in our experiments as backbones. Specifically, we remove the attention pooling layer and add a randomly initialized decoder head. The output stride is 16. Single-source and multi-source models are trained respectively for $40K$ and $60K$ iterations with a batch size of 8. The training images are cropped to 768×768. Stochastic Gradient Descent (SGD) with a momentum of 0.9 and weight decay of 10^{-4} is used as optimizer. Polynomial decay with a power of 0.9 is used, with an initial learning rate of 10^{-1} for the classifier and 10^{-2} for the backbone. We use color jittering and horizontal flip as data augmentation. Label smoothing regularization [46] is adopted. For style mining, *Layer1* features are divided into 9 patches. Each patch is resized to 56×56, corresponding to the dimensions of *Layer1* features for an input image of size 224×224 (*i.e.* the input dimension of CLIP). We use ImageNet templates[2] for each prompt.

Evaluation metric. We evaluate our models on the validation sets of the unseen target domains with mean Intersection over Union (mIoU%) of the 19 shared semantic classes. For each experiment, we report the average of *three runs*.

4.2. Comparison with DGSS methods

Single-source DGSS. We compare FAMix with state-of-the-art DGSS methods under the single-source setting.

Training on GTAV (G) as source, Tab. 2 reports models trained with either ResNet-50 or ResNet-101 backbones. The unseen target datasets are C, B, M, S, and the four subsets of A. Tab. 2 shows that our method significantly outperforms all the baselines on all the datasets for both backbones. We note that WildNet [29] and TLDR [24] use extra-data, while SHADE [58] uses the full G dataset (24,966 images) for training with ResNet-101. Class-wise performances are reported in Appendix B.

Training on Cityscapes (C) as source, Tab. 3 reports performance with ResNet-50 backbone. The unseen target datasets are B, M, G, and S. The table shows that our method outperforms the baseline in average, and is competitive to SOTA on G and M.

Multi-source DGSS. We also show the effectiveness of FAMix in the multi-source setting, training on G+S and

[2] https://github.com/openai/CLIP/

Method	B	M	G	S	Mean
RobustNet [7]	50.73	58.64	45.00	26.20	45.14
Pin the memory [23]	46.78	55.10	-	-	-
SiamDoGe [51]	51.53	**59.00**	45.08	26.67	45.57
WildNet* [29]	50.94	58.79	**47.01**	27.95	46.17
DPCL [53]	52.29	-	46.00	26.60	-
FAMix (ours)	**54.07**	58.72	45.12	**32.67**	**47.65**

Table 3. **Single-source DGSS trained on C.** Performance (mIoU %) of FAMix compared to other DGSS methods trained on C and evaluated on B, M, G and S for ResNet-50 backbone. * indicates the use of extra-data. We emphasize **best** and second best results.

Method	C	B	M	Mean
RobustNet [7]	37.69	34.09	38.49	36.76
Pin the memory [23]	44.51	38.07	42.70	41.76
SHADE [58]	47.43	40.30	47.60	45.11
SPC-Net [19]	46.36	43.18	48.23	45.92
TLDR* [24]	48.83	42.58	47.80	46.40
FAMix (ours)	**49.41**	**45.51**	**51.61**	**48.84**

Table 4. **Multi-source DGSS.** Performance (mIoU %) of FAMix compared to other DGSS methods trained on G+S and evaluated on C, B, M for ResNet-50 backbone. * indicates the use of extra-data. We emphasize **best** and second best results.

evaluating on C, B and M. The results reported in Tab. 4 for ResNet-50 backbone outperform state-of-the-art.

Qualitative results. We visually compare the segmentation results with Pin the memory [23], SHADE [58] and WildNet [29] in Fig. 3. FAMix clearly outperforms other DGSS methods on "stuff" (*e.g.*, road and sky) and "things" (*e.g.*, bicycle and bus) classes.

4.3. Decoder-Probing Fine-Tuning (DP-FT)

Kumar *et al.* [26] show that standard fine-tuning may distort the pretrained feature representation, leading to degraded OOD performances for classification. Consequently, they propose a two-step training strategy: (1) Training a linear probe (LP) on top of the frozen backbone features, (2) Fine-tuning (FT) both the linear probe and the backbone. Inspired by it, Saito *et al.* [43] apply the same strategy for object detection, which is referred to as Decoder-probing Fine-tuning (DP-FT). They observe that DP-FT improves over DP depending on the architecture. We hypothesize that the effect is also dependent on the pretraining paradigm and the downstream task. As observed in Tab. 1, CLIP might remarkably *overfit* the source domain when fine-tuned. In Tab. 5, we compare fine-tuning (FT), decoder-probing (DP) and DP-FT. DP brings improvements over FT since it completely preserves the pretrained representation. Yet, DP major drawback lies in its limitation to adapt features for the downstream task, resulting in suboptimal results. Surprisingly, DP-FT largely falls behind DP, meaning

Method	C	B	M	S	AN	AS	AR	AF	Mean
FT	16.81	16.31	17.80	27.10	2.95	8.58	14.35	13.61	14.69
DP	34.13	37.67	42.21	29.10	10.71	26.26	29.47	30.40	29.99
DP-FT	25.62	21.71	26.39	31.45	4.22	18.26	20.07	20.85	21.07
FAMix (ours)	**48.15**	**45.61**	**52.11**	**34.23**	**14.96**	**37.09**	**38.66**	**40.25**	**38.88**

Table 5. **FAMix *vs.* DP-FT**. Performance (mIoU%) of FAMix compared to Fine-tuning (FT), Decoder-probing (DP) and Decoder-probing Fine-tuning (DP-FT). We use here ResNet-50, trained on G. We emphasize **best** and second best results.

Freeze	Augment	Mix	C	B	M	S	AN	AS	AR	AF	Mean
✗	✗	✗	16.81	16.31	17.80	27.10	2.95	8.58	14.35	13.61	14.69
✗	✓	✗	22.48	26.05	24.15	25.40	4.83	17.61	22.86	19.75	20.39
✗	✗	✓	20.07	21.24	22.91	26.52	1.28	14.99	22.09	20.51	18.70
✗	✓	✓	27.53	26.59	26.27	26.91	4.90	18.91	25.60	22.14	22.36
✓	✗	✗	37.83	38.88	44.24	31.93	12.41	29.59	31.56	33.05	32.44
✓	✓	✗	36.65	35.73	37.32	30.44	14.72	34.65	34.91	38.98	32.93
✓	✗	✓	43.43	43.79	48.19	33.70	11.32	35.55	36.15	38.19	36.29
✓	✓	✓	**48.15**	**45.61**	**52.11**	**34.23**	**14.96**	**37.09**	**38.66**	**40.25**	**38.88**

Table 6. **Ablation of FAMix components.** Performance (mIoU %) after removing one or more components of FAMix.

that the learned features over-specialize to the source domain distribution even with a "decoder warm-up".

The results advocate for the need of specific strategies to preserve CLIP robustness for semantic segmentation. This need emerges from the additional gap between pretraining (*i.e.* aligning object-level and language representations) and fine-tuning (*i.e.* supervised pixel classification).

4.4. Ablation studies

We conduct all the ablations on a ResNet-50 backbone with GTAV (G) as source dataset.

Removing ingredients from the recipe. FAMix is based on minimal fine-tuning of the backbone (*i.e.*, Freeze), style augmentation and mixing. We show in Tab. 6 that the best generalization results are only obtained when combining the *three* ingredients. Specifically, when the backbone is fine-tuned (*i.e.*, Freeze ✗), the performances are largely harmed. When minimal fine-tuning is performed (*i.e.*, Freeze ✓), we argue that the augmentations are too strong to be applied without style mixing; the latter brings both effects of domain interpolation and use of the original statistics. Subsequently, when style mixing is not applied (*i.e.* Freeze ✓, Augment ✓, Mix ✗), the use of mined styles brings mostly no improvement on OOD segmentation compared to training without augmentation (*i.e.* Freeze ✓, Augment ✗, Mix ✗). Note that for Freeze ✓, Augment ✓, Mix ✗, the line 8 in Algorithm 2 becomes:

$$\mathbf{f}_s^{(ij)} \leftarrow \texttt{AdaIN}(\mathbf{f}_s^{(ij)}, \boldsymbol{\mu}^{(ij)}, \boldsymbol{\sigma}^{(ij)}) \tag{5}$$

Our style mixing is different from MixStyle [62] for being applied: (1) patch-wise and (2) between original styles of

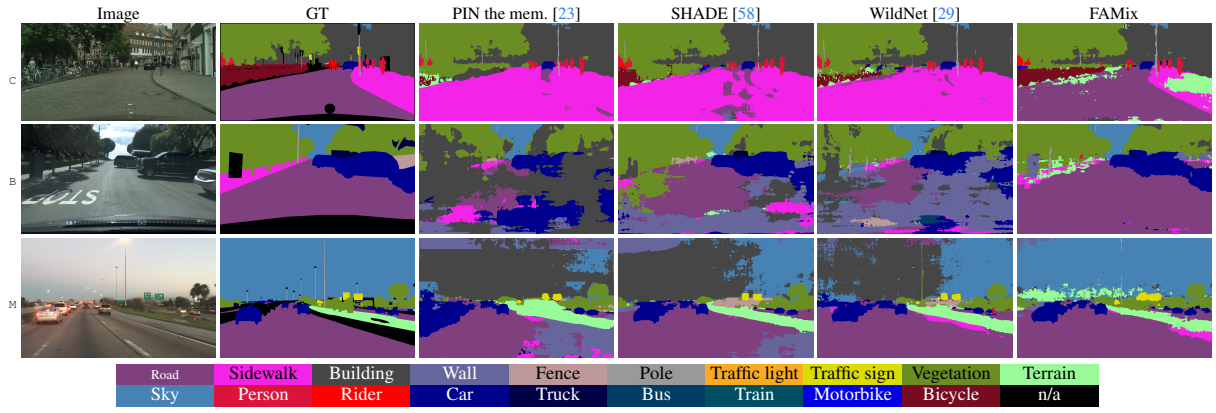


Figure 3. **Qualitative results.** *Columns 1-2*: Image and ground truth (GT), *Columns 3-4-5*: DGSS methods results, *Column 6*: Our results. The models are trained on G with ResNet-50 backbone.

RCP	RSP	CN	C	B	M	S	AN	AS	AR	AF	Mean
		✓	45.99	43.71	50.48	**34.75**	15.22	35.09	34.92	38.17	37.29
✓			46.10	44.24	48.90	33.62	13.39	35.99	36.68	39.86	37.35
	✓		45.64	44.59	49.13	33.64	**15.33**	**37.32**	35.98	38.85	37.56
✓		✓	47.83	44.83	50.38	34.27	14.43	37.07	37.07	38.76	38.08
	✓	✓	**48.15**	**45.61**	**52.11**	34.23	14.96	37.09	**38.66**	**40.25**	**38.88**

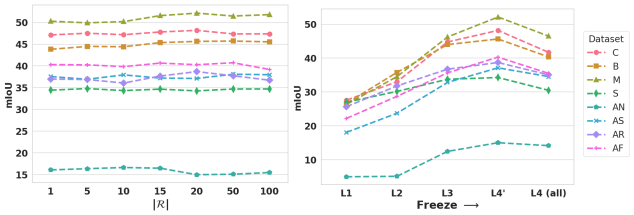
Table 7. **Ablation on the prompt construction.** Performance (mIoU %) for different prompt constructions. RCP, RSP and CN refer to <random character prompt>, <random style prompt> and <class name>, respectively.

the source data and augmented versions of them. Note that the case (Freeze ✓, Augment ✗, Mix ✓) could be seen as a variant of MixStyle, yet applied locally and class-wise. Our complete recipe is proved to be significantly more effective with a boost of $\approx +6$ mean mIoU w.r.t. the baseline of training without augmentation and mixing.

Prompt construction. Tab. 7 reports results when ablating the prompt construction. In FAMix, the final prompt is derived by concatenating <random style prompt> and <class name>; removing either of those leads to inferior results. Interestingly, replacing the style prompt by random characters – *e.g.* "ioscjspa" – does not significantly degrade the performance. In certain aspects, using random prompts still induces a randomization effect within the FAMix framework. However, meaningful prompts still consistently lead to the best results.

Number of style prompts. FAMix uses a set $\mathcal{R}$ of random style prompts which are concatenated with the class names; $\mathcal{R}$ is formed by querying ChatGPT[3] using <give me 20 prompts of 2 to 5 words describing random image styles>. The output prompts are provided in Appendix C. Fig. 4a shows that the size of $\mathcal{R}$ has a marginal impact on FAMix performance. Yet, the mIoU scores on C, B, M and AR are higher for $|\mathcal{R}| = 20$ compared to $|\mathcal{R}| = 1$ and almost equal for the other datasets.

[3] https://chat.openai.com/



(a) Number of prompts (b) Effect of layer freezing

Figure 4. **Ablation of prompt set and freezing strategy.** (a) Performance (mIoU %) on test datasets w.r.t. the number of random style prompts in $\mathcal{R}$. (b) Effect of freezing layers reporting on x-axis the last frozen layer. For example, 'L3' means freezing L1, L2 and L3. 'L4' ' indicates that the *Layer4* is partially frozen.

The low sensitivity of the performance to the size of $\mathcal{R}$ could be explained by two factors. First, mining even from a single prompt results in different style variations as the optimization starts from different anchor points in the latent space, as argued in [10]. Second, mixing style between the *source* and the mined *proxy* domains is the crucial factor making the network explore intermediate domains during training. This does not contradict the effect of our prompt construction which leads to the best results (Tab. 7).

Local *vs*. global style mining. To highlight the effect of our class-wise local style mining, we perform an ablation replacing it with global style mining. Specifically, the same set of <random style prompt> are used, though being concatenated with <driving> as a global description instead of local class name. Intuitively, local style mining and mixing induces richer style variations and more contrast among patches. The results in Tab. 8 show the effectiveness of our local style mining and mixing strategy, bringing about 3 mIoU improvement on G → C.

What to mix? Let $\mathcal{S} = \bigcup_{k=1}^{K} \mathcal{S}^{(k)}$ and $\mathcal{T} = \bigcup_{k=1}^{K} \mathcal{T}^{(k)}$ the sets of class-wise source and augmented features, respectively. In FAMix training, for an arbitrary patch $\mathbf{f}_s^{(ij)}$,

Syle mining		C	B	M	S	AN	AS	AR	AF	Mean
global w/	"street view"	45.51	45.12	50.40	33.65	14.59	36.92	37.38	40.53	38.01
	"urban scene"	46.59	45.38	51.33	33.67	14.42	35.96	37.30	40.52	38.15
	"roadscape"	45.49	45.55	50.63	33.66	14.77	36.75	37.07	40.33	38.03
	"commute snapshot"	45.39	45.08	50.50	33.68	13.65	36.63	37.93	**40.92**	37.97
	"driving"	45.06	44.98	50.67	33.36	14.84	35.11	36.21	39.52	37.47
local		**48.15**	**45.61**	**52.11**	**34.23**	**14.96**	**37.09**	**38.66**	40.25	**38.88**

Table 8. **Ablation on style mining.** Global style mining consists of mining one style per feature map, using <random style prompt> + <global class> as prompt.

Syle mining	C	B	M	S	AN	AS	AR	AF	Mean
$\mathcal{S}$	43.43	43.79	48.19	33.70	11.32	35.55	36.15	38.19	36.29
$\mathcal{S} \cup \mathcal{T}$	44.76	45.59	50.78	34.05	13.67	36.92	37.18	38.13	37.64
$\mathcal{T}$ (ours)	**48.15**	**45.61**	**52.11**	**34.23**	**14.96**	**37.09**	**38.66**	**40.25**	**38.88**

Table 9. **Ablation on the sets used for mixing.** The styles $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ used in (3) and (4) are sampled either from $\mathcal{S}$ or $\mathcal{S} \cup \mathcal{T}$ or $\mathcal{T}$.

style mixing is performed between the original source statistics and statistics sampled from the augmented set (*i.e.*, $(\boldsymbol{\mu}^{(ij)}, \boldsymbol{\sigma}^{(ij)}) \in \mathcal{T}^{(c_p^{(ij)})}$, see (3) and (4)). In class-wise vanilla MixStyle, $(\boldsymbol{\mu}^{(ij)}, \boldsymbol{\sigma}^{(ij)}) \in \mathcal{S}^{(c_p^{(ij)})}$. In Tab. 9, we show that sampling $(\boldsymbol{\mu}^{(ij)}, \boldsymbol{\sigma}^{(ij)})$ from $\mathcal{S}^{(c_p^{(ij)})} \cup \mathcal{T}^{(c_p^{(ij)})}$ does not lead to better generalization, despite sampling from a set with twice the cardinality. This supports our mixing strategy visualized in Fig. 1. Intuitively, sampling from $\mathcal{S} \cup \mathcal{T}$ could be viewed as applying either MixStyle or our mixing with a probability $p = 0.5$.

Minimal fine-tuning. We argue for minimal fine-tuning as a compromise between pretrained feature preservation and adaptation. Fig. 4b shows an increasing OOD generalization trend with more freezing. Interestingly, only fine-tuning the last layers of the last convolutional block (where the dilation is applied) achieves the best results. When training on Cityscapes, we observed that freezing all the layers except *Layer4* achieves the best results.

4.5. Does FAMix require language?

Inspired by the observation that target statistics deviate around the source ones in real cases [11], we conduct an experiment where we replace language-driven style mining by noise perturbation. The same procedure of FAMix is kept: (i) Features are divided into patches, perturbed with noise and then saved into a style bank based on the dominant class; (ii) During training, patch-wise style mixing of original and perturbed styles is performed.

Different from Fan *et al.* [11], who perform a perturbation on the feature statistics using a normal distribution with pre-defined parameters, we experiment perturbation with different magnitudes of noise controlled by the signal-to-noise ratio (SNR). Consider the mean of a patch $\mu \in \mathbb{R}^c$ as a signal, the goal is to perturb it with some noise

SNR	C	B	M	S	AN	AS	AR	AF	Mean
Baseline	37.83	38.88	44.24	31.93	12.41	29.59	31.56	33.05	32.44
5	28.78	29.24	30.32	21.67	12.60	24.00	25.95	25.87	24.80
10	40.09	39.50	43.45	29.09	13.36	33.47	33.11	36.17	33.53
15	45.02	44.16	48.63	32.96	14.55	36.09	35.99	**40.96**	37.30
20	45.52	44.29	49.26	33.45	12.40	35.96	36.52	38.60	37.00
25	44.82	44.26	48.54	33.30	11.38	34.51	35.46	37.61	36.24
30	43.07	43.80	48.31	33.47	12.33	35.05	35.58	38.10	36.21
∞	43.43	43.79	48.19	33.70	11.32	35.55	36.15	38.19	36.29
MixStyle [62]	40.97	42.04	48.36	33.15	13.14	31.26	34.94	38.12	35.25
Prompts	**48.15**	**45.61**	**52.11**	**34.23**	**14.96**	**37.09**	**38.66**	40.25	**38.88**

Table 10. **Noise vs prompt-driven augmentation.** The prompt-driven augmentation in FAMix is replaced by random noise with different levels defined by SNR. We also include vanilla MixStyle. The prompt-driven strategy is superior.

$n_\mu \in \mathbb{R}^c$. The SNR_{dB} between $\|\mu\|$ and $\|n_\mu\|$ is defined as $\text{SNR}_{\text{dB}} = 20 \log_{10} (\|\mu\| / \|n_\mu\|)$. Given μ, SNR_{dB}, and $n \sim \mathcal{N}(0, I)$, where $I \in \mathbb{R}^{c \times c}$ is the identity matrix, the noise is computed as $n_\mu = 10^{\frac{-\text{SNR}}{20}} \frac{\|\mu\|}{\|n\|} n$. We add $\mu + n_\mu$ to the style bank corresponding to the dominant class in the patch. The same applies to $\sigma \in \mathbb{R}^c$. The results of training for different noise levels are in Tab. 10. Using language as source of randomization outperforms any noise level. The baseline corresponds to the case where no augmentation nor mixing are performed (See Tab. 6, Freeze ✓, Augment ✗, Mix ✗). SNR=∞ could be seen as a variant of MixStyle, applied class-wise to patches (See Tab. 6, Freeze ✓, Augment ✗, Mix ✓). The vanilla MixStyle gets inferior results.

Besides lower OOD performance, one more disadvantage of noise augmentation compared to our language-driven augmentation is the need to select a value for the SNR, for which the optimal value might vary depending on the target domain encountered at the test time.

5. Conclusion

We presented FAMix, a simple recipe for domain generalized semantic segmentation with CLIP pretraining. We proposed to locally mix the styles of source features with their augmented counterparts obtained using language prompts. Combined with minimal fine-tuning, FAMix significantly outperforms the state-of-the-art approaches. Extensive experiments showcase the effectiveness of our framework. We hope that FAMix will serve as a strong baseline in future works, exploring the potential of leveraging large-scale vision-language models for perception tasks.

Acknowledgment. This work was partially funded by French project SIGHT (ANR-20-CE23-0016) and was supported by ELSA - European Lighthouse on Secure and Safe AI funded by the European Union under grant agreement No. 101070617. It was performed using HPC resources from GENCI–IDRIS (Grant AD011014477).

References

[1] Kartik Ahuja, Ethan Caballero, Dinghuai Zhang, Jean-Christophe Gagnon-Audet, Yoshua Bengio, Ioannis Mitliagkas, and Irina Rish. Invariance principle meets information bottleneck for out-of-distribution generalization. In *NeurIPS*, 2021. 2

[2] James Urquhart Allingham, Jie Ren, Michael W Dusenberry, Xiuye Gu, Yin Cui, Dustin Tran, Jeremiah Zhe Liu, and Balaji Lakshminarayanan. A simple zero-shot prompt weighting technique to improve prompt ensembling in text-image models. In *ICML*, 2023. 2

[3] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. 2

[4] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In *NeurIPS*, 2018. 2

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 5

[6] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021. 3

[7] Sungha Choi, Sanghun Jung, Huiwon Yun, Joanne T Kim, Seungryong Kim, and Jaegul Choo. Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In *CVPR*, 2021. 1, 2, 3, 5, 6

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 5

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 3

[10] Mohammad Fahes, Tuan-Hung Vu, Andrei Bursuc, Patrick Pérez, and Raoul de Charette. Poda: Prompt-driven zero-shot domain adaptation. In *ICCV*, 2023. 1, 3, 4, 7

[11] Qi Fan, Mattia Segu, Yu-Wing Tai, Fisher Yu, Chi-Keung Tang, Bernt Schiele, and Dengxin Dai. Towards robust object detection invariant to real-world domain shifts. In *ICLR*, 2023. 5, 8

[12] Alex Fang, Gabriel Ilharco, Mitchell Wortsman, Yuhao Wan, Vaishaal Shankar, Achal Dave, and Ludwig Schmidt. Data determines distributional robustness in contrastive language image pre-training (clip). In *ICML*, 2022. 1, 2

[13] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 2016. 1

[14] Yunhao Ge, Jie Ren, Andrew Gallagher, Yuxiao Wang, Ming-Hsuan Yang, Hartwig Adam, Laurent Itti, Balaji Lakshminarayanan, and Jiaping Zhao. Improving zero-shot generalization and robustness of multi-modal models. In *CVPR*, 2023. 2

[15] Sachin Goyal, Ananya Kumar, Sankalp Garg, Zico Kolter, and Aditi Raghunathan. Finetune like you pretrain: Improved finetuning of zero-shot vision models. In *CVPR*, 2023. 1, 2

[16] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *ICLR*, 2022. 1

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[18] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018. 1

[19] Wei Huang, Chang Chen, Yong Li, Jiacheng Li, Cheng Li, Fenglong Song, Youliang Yan, and Zhiwei Xiong. Style projected clustering for domain generalized semantic segmentation. In *CVPR*, 2023. 1, 3, 5, 6

[20] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 3

[21] Nishant Jain, Harkirat Behl, Yogesh Singh Rawat, and Vibhav Vineet. Efficiently robustify pre-trained models. In *ICCV*, 2023. 2

[22] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, 2021. 1

[23] Jin Kim, Jiyoung Lee, Jungin Park, Dongbo Min, and Kwanghoon Sohn. Pin the memory: Learning to generalize semantic segmentation. In *CVPR*, 2022. 1, 5, 6, 7

[24] Sunghwan Kim, Dae-hwan Kim, and Hoseong Kim. Texture learning domain randomization for domain generalized segmentation. In *ICCV*, 2023. 1, 3, 5, 6

[25] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *ICML*, 2021. 2

[26] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *ICLR*, 2022. 1, 2, 6

[27] Gihyun Kwon and Jong Chul Ye. Clipstyler: Image style transfer with a single text condition. In *CVPR*, 2022. 1

[28] Clement Laroudie, Andrei Bursuc, Mai Lan Ha, and Gianni Franchi. Improving clip robustness with knowledge distillation and self-training. *arXiv preprint arXiv:2309.10361*, 2023. 2

[29] Suhyeon Lee, Hongje Seong, Seongwon Lee, and Euntai Kim. Wildnet: Learning domain generalized semantic segmentation from the wild. In *CVPR*, 2022. 1, 3, 5, 6, 7

[30] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *ICLR*, 2022. 1

[31] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *CVPR*, 2018. 1

[32] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generaliza-

tion via conditional invariant adversarial networks. In *ECCV*, 2018. 1, 2

[33] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional learning for domain adaptation of semantic segmentation. In *CVPR*, 2019. 1

[34] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, 2018. 1

[35] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017. 5

[36] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *ECCV*, 2018. 1, 3, 5

[37] Duo Peng, Yinjie Lei, Munawar Hayat, Yulan Guo, and Wen Li. Semantic-aware domain generalized segmentation. In *CVPR*, 2022. 1, 3, 5

[38] Fengchun Qiao, Long Zhao, and Xi Peng. Learning to learn single domain generalization. In *CVPR*, 2020. 2

[39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1, 2, 3

[40] Yongming Rao, Wenliang Zhao, Guangyi Chen, Yansong Tang, Zheng Zhu, Guan Huang, Jie Zhou, and Jiwen Lu. Denseclip: Language-guided dense prediction with context-aware prompting. In *CVPR*, 2022. 1

[41] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 5

[42] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 5

[43] Kuniaki Saito, Donghyun Kim, Piotr Teterwak, Rogerio Feris, and Kate Saenko. Mind the backbone: Minimizing backbone distortion for robust object detection. *arXiv preprint arXiv:2303.14744*, 2023. 6

[44] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *ICCV*, 2021. 5

[45] Yang Shu, Xingzhuo Guo, Jialong Wu, Ximei Wang, Jianmin Wang, and Mingsheng Long. Clipood: Generalizing clip to out-of-distributions. In *ICML*, 2023. 1, 2

[46] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 5

[47] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017. 1

[48] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *CVPR*, 2019. 1

[49] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *T-KDE*, 2022. 1, 2

[50] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *CVPR*, 2022. 1, 2

[51] Zhenyao Wu, Xinyi Wu, Xiaoping Zhang, Lili Ju, and Song Wang. Siamdoge: Domain generalizable semantic segmentation using siamese network. In *ECCV*, 2022. 1, 3, 5, 6

[52] Qinwei Xu, Ruipeng Zhang, Ya Zhang, Yanfeng Wang, and Qi Tian. A fourier-based framework for domain generalization. In *CVPR*, 2021. 1

[53] Liwei Yang, Xiang Gu, and Jian Sun. Generalized semantic segmentation by self-supervised source domain projection and multi-level contrastive learning. In *AAAI*, 2023. 1, 5, 6

[54] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, 2020. 5

[55] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. LiT: Zero-shot transfer with locked-image text tuning. In *CVPR*, 2022. 1

[56] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023. 1

[57] Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. Domain generalization via entropy regularization. In *NeurIPS*, 2020. 2

[58] Yuyang Zhao, Zhun Zhong, Na Zhao, Nicu Sebe, and Gim Hee Lee. Style-hallucinated dual consistency learning for domain generalized semantic segmentation. In *ECCV*, 2022. 1, 3, 5, 6, 7

[59] Chong Zhou, Chen Change Loy, and Bo Dai. Extract free dense labels from clip. In *ECCV*, 2022. 1

[60] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Deep domain-adversarial image generation for domain generalisation. In *AAAI*, 2020. 2

[61] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization. In *ECCV*, 2020.

[62] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *ICLR*, 2021. 1, 2, 4, 5, 6, 8

[63] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *TPAMI*, 2022. 1, 2

[64] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, 2022. 1

[65] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *IJCV*, 2022. 1