A Re-Parameterized Vision Transformer (ReVT) for Domain-Generalized Semantic Segmentation

Jan-Aike Termöhlen Timo Bartels Tim Fingscheidt
Technische Universitat Braunschweig, Germany
{j.termoehlen, timo.bartels, t.fingscheidt}@tu-bs.de

Abstract

The task of semantic segmentation requires a model to assign semantic labels to each pixel of an image. However, the performance of such models degrades when deployed in an unseen domain with different data distributions compared to the training domain. We present a new augmentation-driven approach to domain generalization for semantic segmentation using a re-parameterized vision transformer (ReVT) with weight averaging of multiple models after training. We evaluate our approach on several benchmark datasets and achieve state-of-the-art mIoU performance of 47.3% (prior art: 46.3%) for small models and of 50.1% (prior art: 47.8%) for midsized models on commonly used benchmark datasets. At the same time, our method requires fewer parameters and reaches a higher frame rate than the best prior art. It is also easy to implement and, unlike network ensembles, does not add any computational complexity during inference.[1]

1. Introduction

Many methods for machine perception, e.g., for semantic segmentation, employ deep neural networks (DNNs) [11]. Due to the high labeling cost for semantic segmentation data, more and more synthetic data are used for training these DNNs. After training on the labeled (source) domain they should operate as robustly as possible in similar, but unseen (target) domains. However, this is often not the case since the data of the target domain differ from those of the training domain, leading to a so-called domain gap. There are many methods to deal with this domain gap that either require samples from the target domain during training [2, 3, 35], or alter the target data or the network parameters during inference [18, 19, 38]. An approach that does not have these drawbacks is domain generalization (DG). The aim of domain generalization is to train a network in a way that it generalizes well to unseen domains without any adaptation steps. Although neural networks that employ vision transformer encoders currently achieve the best performance in segmentation tasks, modern DG methods are mostly presented with ResNet-based models, such as DeepLabv3+ [4] and FCN [23]. Due to its strong performance with a comparable or smaller number of parameters, we employ the transformer-based SegFormer [42] as the baseline for our domain generalization method.

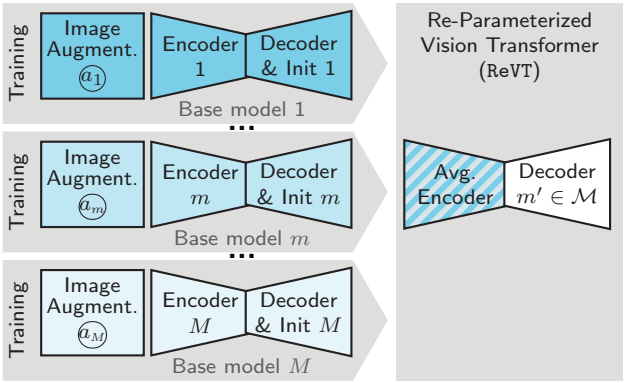

Figure 1. **High-level overview** of the generalization method. The set of all M base models trained on individual augmentations a_m is denoted by $\mathcal{M} = \{1, ..., m, ..., M\}$. For the re-parameterized vision transformer (ReVT), any decoder $m' \in \mathcal{M}$ can be used.

A training or post-processing method that has proven itself in many applications is re-parameterization. Here, either individual layers, e.g., convolutional layers, or entire models trained with potentially different augmentations can be averaged to improve performance and generalization of the final model. The averaging can be performed either during training [15, 36] or after training [40, 41]. As sketched in Figure 1, in our work we advantageously combine the strengths of selected image augmentations [13, 39, 42] with the re-parameterization and show that this method leads to a significantly better generalization capability for transformer-based models. We also show that the method does not improve the performance of the commonly used ResNet-based models when trained with standard stochastic gradient descent (SGD), but that this can be

[1] Code is available at https://github.com/ifnspaml/ReVT

overcome by the use of the AdamW optimizer [24].

As shown in Figure 1, first, M base models are trained, with pre-trained encoders but different random decoder seeding, and potentially with dissimilar augmentations $\textcircled{a}_m$. Afterwards, the encoder networks can be averaged into one new encoder (re-parameterization), which extracts better generalizing features. This encoder can then be combined with any of the previously trained decoders and be used directly for segmentation.

Our contribution with this work is fourfold. First, we propose a re-parameterized vision transformer `ReVT` for domain-generalized semantic segmentation, resulting from M augmentation-individual base models. We achieve higher mIoU on unseen domains compared to methods that employ `ResNet`-based models, while requiring fewer parameters and achieving higher frame rates than the best prior art. Second, we analyze the effect of different network architectures, network parts, layer types, and optimizers on the re-parameterization. Third, we report on two more real datasets as common in the field and also going beyond customs in the field, we follow a stringent divison of data splits into training, development, and test set. Finally, we set a new state-of-the-art benchmark on the synthetic-to-real domain generalization task for semantic segmentation.

2. Related Work

In this section, we discuss related works for our single-source domain generalization method. We start with the task of domain generalization, followed by related work on image augmentation and model re-parameterization.

2.1. Domain Generalization (DG)

In domain generalization for semantic segmentation, a model is trained on a set of labeled data from a specific (source) domain $\mathcal{D}^{\mathrm{S}}$ and then evaluated on new data from unseen (target) domains $\mathcal{D}^{\mathrm{T}}$. The goal is to train a model that can generalize well to different domains and accurately segment new images. Following Qiao et al. [31], we distinguish between domain generalization and *single-source* domain generalization. The main difference between these two is that in the former, the model can be exposed to multiple domains during training, e.g., multiple labeled source domains or additional auxiliary domains. A dataset often used as an auxilliary domain is ImageNet [9], which is used to learn the style of real images [14,44]. In the *single-source* domain generalization task, the model is trained solely on one single domain.

Muandet et al. [25] proposed a so-called domain-invariant component analysis (DICA) minimizing the dissimilarity across domains during training. Liet al. [21] learn a domain-agnostic model on multiple domains via low-rank parameterized CNNs. Zhang et al. [47] employ meta-learning for domain generalization and an adaptation of batch norm statistics in the target domain, and therefore present no pure DG method. Li et al. [22] propose an episodic training with a simple approach of aggregating data from multiple source domains for training. Yue et al. [44] first randomize the images with the style from real domains and then also enforce pyramid consistency between different styles. Their approach is not single-source domain, but requires an auxiliary domain for the style transfer. Huang et al. [14] follow a similar approach, but proposed to perform the domain randomization in the frequency domain of the images. Pan et al. [28] proposed a new instance-batch normalization (IBN) that is more robust w.r.t. appearance changes such as color shifts or brightness changes. Choi et al. [5] proposed an advanced loss that uses instance selective whitening. Peng et al. [30] proposed a network that includes semantic-aware normalization (SAN) as well as semantic-aware whitening (SAW). WildNet [20] employs feature stylization with styles from an auxiliary domain and enforces semantic consistency between the segmentation masks of stylized and original images and also between the segmentation masks of stylized images and the labels. Other than previous methods, that either perform checkpoint selection[2] [14, 44] or hyperparameter tuning on evaluation data (official validation sets) of the target domains, we follow a stringent approach with distinct development sets for method design and hyperparameter tuning and perform no checkpoint selection (cf. Section 4.3). We also evaluate our approach on additional real domains, some of which represent strong domain shifts (cf. Section 4.1), and have not been explored by previous approaches.

2.2. Image Augmentation

Image augmentation techniques [8, 12, 13, 27, 45, 46] aim at improving the performance of DNNs by increasing the variability of the training data. They reduce the risk of overfitting, e.g., to synthetic textures [17], and can improve the generalization capability of the model. Some recent augmentation methods mix full images [46], parts of images [45], specific class pixels [27], or combine the previously mentioned augmentation strategies with other image transformations [12, 13]. We propose to use a number of M so-called base models with *individual* augmentations $\textcircled{a}_m$ drawn from PixMix [13], bilateral filtering [39], and the baseline augmentations from the `SegFormer` method [42].

2.3. Model Re-Parameterization

The stochastic weight averaging (SWA) [15] method averages the network weights of the model during the training process with stochastic gradient descent (SGD) using a cyclical or constant learning rate. Similarly, Sämann et

[2] cf. https://github.com/jxhuang0508/FSDR/issues/2#issuecomment-910089417

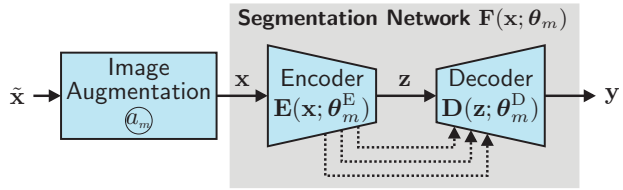

Figure 2. **Training setup** of base model m and notations. Dotted lines indicate skip connections.

al. [36] also employ the model averaging during training. A related method was also investigated by Kamp et al. [16] as an efficient decentralized learning protocol.

In contrast to methods that employ the re-parameterization during the training process [15, 36, 37], we adopt the re-parameterization approach by Wortsman et al. [41] that the authors dubbed "model soups" and performed the averaging of the model weights after various training processes. Note that it is also possible to re-parameterize specific layers and alter the architecture after re-parameterization, e.g., with RepVGG [10]. Wang et al. [40] analyzed these re-parameterization strategies for convolutional layers in different networks and proposed an advanced planned re-parameterized model.

3. Proposed Method

In this section, we will describe the mathematical notations and our new re-parameterized vision transformer (ReVT) training, including augmentations.

3.1. Mathematical Notations

A high-level overview of the employed training setup is given in Figure 2. During training in the labeled source domain $\mathcal{D}^{\mathrm{S}}$, an image $\tilde{\mathbf{x}}$ is subject to augmentation methods and then denoted as $\mathbf{x} \in \mathbb{G}^{H\times W\times C}$, where $\mathbb{G}$ denotes the set of integer gray values, H and W the image height and width in pixels, and $C=3$ the number of color channels. The augmented images $\mathbf{x}$ are then transformed by the segmentation network $\mathbf{F}$ with network parameters $\boldsymbol{\theta}$ to obtain an output tensor $\mathbf{y} = \mathbf{F}(\mathbf{x};\boldsymbol{\theta}) = (y_{i,s}) \in \mathbb{I}^{H\times W\times S}$ that contains a pixel-wise posterior probability $y_{i,s} = \mathrm{P}(s|i,\mathbf{x})$ for all classes $s \in \mathcal{S}$ at each pixel index $i \in \mathcal{I} = \{1,2,...,H\cdot W\}$, with $\mathbb{I} = [0,1]$. The segmentation network consists of an encoder $\mathbf{z} = \mathbf{E}(\mathbf{x};\boldsymbol{\theta}^{\mathrm{E}})$ and a decoder (segmentation head) $\mathbf{y} = \mathbf{D}(\mathbf{z};\boldsymbol{\theta}^{\mathrm{D}})$, with the parameters $\boldsymbol{\theta}^{\mathrm{E}}$ and $\boldsymbol{\theta}^{\mathrm{D}}$, respectively, resulting in $\mathbf{y} = \mathbf{F}(\mathbf{x};\boldsymbol{\theta}) = \mathbf{D}(\mathbf{E}(\mathbf{x};\boldsymbol{\theta}^{\mathrm{E}});\boldsymbol{\theta}^{\mathrm{D}})$. The number of parameters in a parameter tensor is denoted as $|\boldsymbol{\theta}|$. Different parameter tensors $\boldsymbol{\theta}_m$ for the same architecture are marked by a subscript $m \in \mathcal{M}$, where $\mathcal{M} = \{1,2,...,M\}$ is the respective index set and M is the total number of models. The set of classes $\mathcal{S} = \{1,2,...,S\}$ contains the same S classes for source domain training and target domain inference (closed set). To obtain the final classification map $\mathbf{m} = (m_i) \in \mathcal{S}^{H\times W}$, we compute $m_i = \arg\max_{s\in\mathcal{S}} y_{i,s}$.

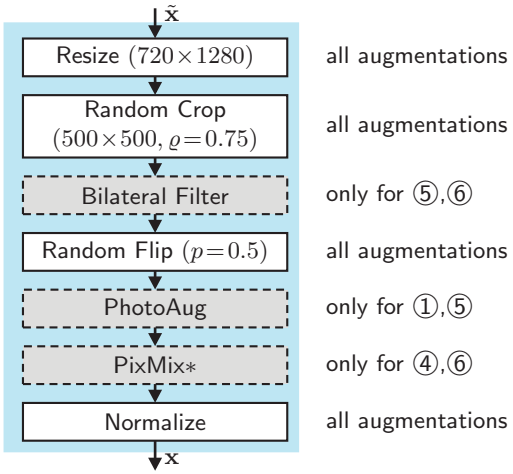

Figure 3. **Image augmentation pipeline** from Figures 1 and 2 employed during training. Use of blocks for augmentations @ noted at the side (cf. Section 4.2).

3.2. Re-Parameterized Vision Transformer (ReVT)

To the best of our knowledge we are the first to introduce vision transformer re-parameterization to domain generalization for semantic segmentation. In particular, each of the base models has seen an individual augmentation strategy in training. An illustrated overview of our proposed single-source domain generalization method for semantic segmentation is given in Figure 1. First, M segmentation networks of the same architecture are trained using ImageNet-pretrained encoders and different random decoder seeds and potentially also different augmentation strategies.

Image augmentation: The base model-individual augmentation steps employed during training are an important component of our method. We have illustrated the image augmentation pipeline in Figure 3. The baseline augmentation pipeline consists of resizing, random cropping, random flipping (Random Flip), photometric augmentation (PhotoAug), followed by a normalization to zero mean and unit variance. The bilateral filter [39] can be inserted before the random flipping (Figure 3, upper gray box). While PhotoAug is our default, it can optionally be replaced by the PixMix [13] algorithm (Figure 3, lower gray box). The original PixMix algorithm applies randomly selected augmentations. We employ the baseline augmentations (Random Flip + PhotoAug) here, which is why we refer to our PixMix variant as PixMix*. All non-self-explanatory augmentations are explained in more detail in Supplement Section A. As a result of either different random seeding or augmentation, the network parameters will differ after training (Figure 1, left side).

Re-parameterized vision transformer (ReVT): After

Table 1. **Employed datasets**. The synthetic datasets GTA5 [32] and SYNTHIA [33] are used as (single) source domains ($\mathcal{D}^{\mathrm{S}}$). We employ various real-world datasets as target domains ($\mathcal{D}^{\mathrm{T}}$) to show the generalization capability of the proposed method.

Dataset Name	# Images in $\mathcal{D}_{\text{full}}$	$\mathcal{D}_{\text{train}}$	$\mathcal{D}_{\text{dev}}$	$\mathcal{D}_{\text{test}*}$
GTA5 [32]	24,966	12,403	6,382	-
SYNTHIA [33] (SYN)	9,400	6,580	2,820	-
Cityscapes [7] (CS)	-	-	500	500
Mapillary Vistas [26] (MV)	-	-	-	2,000
BDD100k [43] (BDD)	-	-	-	1,000
ACDC [34]	-	-	-	406
KITTI [1] (KIT)	-	-	-	200

the training, the model weights $\boldsymbol{\theta}_m$, $m \in \mathcal{M}$, can be averaged resulting in $\boldsymbol{\theta}_\diamond$. The new averaged model weights $\boldsymbol{\theta}_\diamond$ could be used during inference. Different to the method described by Sämann et al. [36], *we only re-parameterize the encoder weights*

$$\boldsymbol{\theta}^{\mathrm{E}}_\diamond = \frac{1}{M} \sum_{m \in \mathcal{M}} \boldsymbol{\theta}^{\mathrm{E}}_m \,, \tag{1}$$

resulting in our proposed `ReVT` as

$$\mathbf{y}^{\mathrm{ReVT}} = (y^{\mathrm{ReVT}}_{i,s}) = \mathbf{D}(\mathbf{E}^{\mathrm{ReVT}}(\mathbf{x}; \boldsymbol{\theta}^{\mathrm{E}}_\diamond); \boldsymbol{\theta}^{\mathrm{D}}_{m'}) \,, \tag{2}$$

with an arbitrarily chosen decoder $m' \in \mathcal{M}$.

4. Experimental Setup

In the following, we introduce the employed datasets and network architectures. Afterwards, we explain the training and evaluation settings, as well as the evaluation metrics. All architectures, procedures, and metrics are implemented using `PyTorch` [29] and the `MMSegmentation` toolbox [6].

4.1. Datasets

In our experiments we evaluate multiple established domain generalization benchmarks for semantic segmentation. The definition of the individual splits and their respective number of images is shown in Table 1. As our synthetic domains we employ GTA5 [32] and SYNTHIA [33]. We employ the three commonly used real-world datasets Cityscapes [7], BDD100k [43], and Mapillary Vistas [26] as target domains. Different to other publications, we also employ the ACDC [34] and the KITTI [1] datasets to provide more evidence of domain generalization on real domains. Particularly the ACDC dataset offers considerable benefit, since it includes images from four adverse conditions (fog, nighttime, rain, and snow), which are not present in the synthetic data. In DG benchmarks, there is no common practice

Table 2. Models and corresponding **number of parameters** for the full **segmentation networks** employed in this paper.

Segmentation Network	Encoder	$\lvert\boldsymbol{\theta}\rvert$ $(\cdot 10^6)$
`DeepLabv3+` [4]	`ResNet50`	43.7
	`ResNet101`	62.7
`SegFormer` [42]	`MiT-B2`	27.4
	`MiT-B3`	47.2
	`MiT-B5`	84.7

on choosing which part of the synthetic dataset to use for training. Some publications use the entire GTA5 or SYNTHIA dataset for training [14, 44]. Other publications use the official training split of GTA5 and define their own training split for SYNTHIA [5,20]. We follow Choi et al. [5] and employ their training and development split for SYNTHIA, and the official GTA5 training and validation set for training and development, respectively. Most DG methods base their design decisions on the official validation sets of the target domains and do not report test results. Since this approach is not rigorous, we follow an approach from domain adaptation [2] and sample 500 random images from the (unused) Cityscapes training set to be used as our development set, see Table 1. To allow comparison, we use the official validation sets of the real domains as test sets. To avoid confusion with the official (partly unpublished) test sets, we name our test sets "test$*$".

4.2. Network Architectures

For our experiments we employ two different network architectures that use an encoder-decoder structure, as illustrated in Figure 2. The employed segmentation networks and the corresponding number of parameters are listed in Table 2. First, we use a `SegFormer` [42] architecture with multiple skip connections from early layers to the decoder (`SegFormer` head). Second, a `DeepLabv3+` [4] with only one skip connection from an early layer to the decoder is investigated. To ensure comparability with other reference methods, we will also perform experiments with several encoder sizes. If only `SegFormer` is mentioned and no additional information is given, this shall refer to the use of an `MiT-B5` encoder. If only `DeepLabv3+` is mentioned and no additional information is given, this shall refer to the use of a `ResNet-101` encoder. For the re-parameterization of the models several encoders are required. As can be seen in Figure 1, the M models that are used in this process will be referred to as *base models* (not to be confused with baseline models, which are simply the standard `SegFormer` or `DeepLabv3+` models), each with a potentially different image augmentation. The different image augmentations a_m for each base model m are

Table 3. Performance (mIoU (%)), when **different network parts** are used in the **re-parameterization**. **Training** was performed on the **GTA5** ($\mathcal{D}^{\text{S}} = \mathcal{D}^{\text{GTA5}}_{\text{train}}$) training set. **Evaluation** is performed on the **Cityscapes development set** ($\mathcal{D}^{\text{T}} = \mathcal{D}^{\text{CS}}_{\text{dev}}$). Reported is the mean mIoU of {①,①,①} models. For the re-parameterization, the mean is computed with one averaged encoder and the three associated decoders $m \in \{1, 2, 3\}$. Best results in bold face, second-best underlined.

Segmentation Network	Method: Re-Parameterization ...	mIoU (%) on $\mathcal{D}^{\text{CS}}_{\text{dev}}$
SegFormer (MiT-B5)	... not done (Baseline)	<u>44.3</u>
	... in encoder only	**47.5**
	... in decoder only	31.5
	... in full network	34.2
DeepLabv3+ (ResNet-101)	... not done (Baseline)	**34.7**
	... in encoder only	<u>31.9</u>
	... in decoder only	1.9
	... in full network	1.9

identified by ①, ②, etc. If the same image augmentations is used multiple times, e.g., $a_1 = a_2 = a_3 =$ ①, then the $M = 3$ base models were just trained with a different random seed. We then denote the used augmentations by {①,①,①}.

4.3. Training, Evaluation, and Metrics

The hyperparameters for the image augmentation, training and evaluation (inference) procedures are provided in Supplement Section B.

Unlike other methods [14, 44], we do not use the test∗ sets (official validation sets) of the individual target domains for hyperparameter tuning or selection of training checkpoints. We train all our models for a fixed number of iterations and evaluate the checkpoint from the last iteration. We want to emphasize that we firmly believe that this is closer to a realistic deployment if a domain generalization method. To evaluate the methods, we employ the standard mean intersection over union (mIoU) of 19 segmentation classes [7, 32, 34]. Hyperparameter tuning is only based on our (self-defined) development sets $\mathcal{D}_{\text{dev}}$, see Table 1. Specifically, we employ the mIoU mean on our out-of-domain (OOD) development sets for our design decisions on the proposed ReVT. To compare our method to other reference methods, we also report an mIoU over multiple domains. We follow Lee et al. [20] and Choi et al. [5] and evaluate the benchmark (BM) mean mIoU over the following benchmark set of data splits: $\{\mathcal{D}^{\text{GTA5}}_{\text{dev}}, \mathcal{D}^{\text{SYN}}_{\text{dev}}, \mathcal{D}^{\text{CS}}_{\text{test}*}, \mathcal{D}^{\text{BDD}}_{\text{test}*}, \mathcal{D}^{\text{MV}}_{\text{test}*}\}$. We report the model size $|\boldsymbol{\theta}|$ and the frame rate in frames per second (fps), as measured on an NVIDIA A100 GPU.

Table 4. Performance (mIoU (%)) for different optimizer setups, i.e., optimizer, learning rate, weight decay, etc. We investigate the effect of the standard SegFormer optimizer setup (gray rows) and Deeplabv3+ optimizer setup (yellow rows) as shown in the Supplement Section B, Table 9. **Training** was performed on the full synthetic **GTA5** ($\mathcal{D}^{\text{S}} = \mathcal{D}^{\text{GTA5}}_{\text{train}}$) dataset. Evaluation is performed on the **Cityscapes development set** ($\mathcal{D}^{\text{T}} = \mathcal{D}^{\text{CS}}_{\text{dev}}$). Reported is the mean mIoU of {①,①,①} models. Best results in bold face.

Network	Optimizer setup following ... (cf. Table 9)	mIoU (%) on $\mathcal{D}^{\text{CS}}_{\text{dev}}$	
		Baseline	re-parameterized
SegFormer	SegFormer	44.3	**47.5**
	DeepLabv3+	**46.2**	46.9
DeepLabv3+	SegFormer	**35.3**	**38.5**
	DeepLabv3+	34.7	31.9

5. Evaluation and Discussion

In this section, we will first investigate the basics of re-parameterization w.r.t. re-parameterized network parts, layers, and the number of base models. Afterwards, we evaluate different base model augmentations and optimizer methods during training to design our final ReVT. Finally, we compare our models to prior art DG methods.

5.1. Basic Investigations on Re-Parameterization

For the following experiments on basics of network re-parameterization, we only employ base models that were trained with the baseline image augmentation ①. If not stated otherwise, the experiments are performed with $M = 3$ base models ({①,①,①}). Reported is always the mIoU on the Cityscapes development set ($\mathcal{D}^{\text{T}} = \mathcal{D}^{\text{CS}}_{\text{dev}}$).

Re-parameterized network parts: In Table 3 we investigate the effect of the re-parameterization, when applied to different network parts. We compare baseline models (no re-parameterization) and re-parameterization of the encoder only, the decoder only, and the full network. We show results for the SegFormer as well as for DeepLabv3+. It can be seen, that the encoder-only re-parameterization is the only setup which improves 3.2% absolute (abs.) over the baseline results from 44.3% to 47.5%. We also see that the DeepLabv3+ does not profit at all from any form of re-parameterization, actually, the performance even degrades from 34.7% to 31.9%. Therefore, in the remainder of the paper, we will use the re-parameterization for the vision transformer SegFormer to obtain the ReVT. We will also refer to the encoder-only re-parameterization simply as re-parameterization. In the following experiment we will further investigate why the DeepLabv3+ did not profit from

Table 5. Performance (mIoU (%)) of the SegFormer, when certain enocoder **block or layer types** are used in the **re-parameterization**. **Training** was performed on the full synthetic **GTA5** ($\mathcal{D}^{\text{S}} = \mathcal{D}^{\text{GTA5}}_{\text{train}}$) dataset. **Evaluation** is performed on the **Cityscapes development set** ($\mathcal{D}^{\text{T}} = \mathcal{D}^{\text{CS}}_{\text{dev}}$). Reported is the mean mIoU $\pm$ the standard deviation of {①,①,①} models. For the re-parameterization, the mean $\pm$ standard deviation is computed with one averaged encoder and the three associated decoders $m \in \{1, 2, 3\}$. Best results in bold face, second-best underlined.

Method: Re-Parameterization ...	**mIoU (%) on $\mathcal{D}^{\text{CS}}_{\text{dev}}$**
... not done (Baseline SegFormer)	44.3 ± 1.9
... in all blocks/layers	$\mathbf{47.5} \pm 0.1$
... in patch embedding blocks only	44.7 ± 1.7
... in attention blocks only	45.4 ± 1.1
... in Mix-FFN blocks only	$\underline{47.0} \pm 0.6$
... in convolutional layers only	45.1 ± 1.5
... in fully connected layers only	46.8 ± 0.6

re-parameterization and how this effect can be avoided.

Optimizer choice: In Table 4 we investigate the performance differences of baseline models and re-parameterized models when trained with different optimizer setups. The optimizer setup comprises all settings regarding the training process. We give a detailed list in Supplement Section B in Table 9. We test the effect of the standard optimizer setup for the SegFormer (AdamW, gray rows) and DeepLabv3+ (SGD, yellow rows). It can be seen that the SegFormer baseline is stronger when trained with the DeepLabv3+ setup (46.2% vs. 44.3%), but the gain from re-parameterization becomes significantly smaller (0.7% abs. improvement vs. 3.2% abs. improvement). For the DeepLabv3+, the SegFormer optimizer setup is the much better choice, because on the one hand the baseline has a better performance (35.3% vs. 34.7%), and on the other hand, it shows significant improvement (3.2% abs.) instead of deterioration (−2.8% abs.). For more analysis, see Supplement Section C.

Re-parameterized blocks/layer types: In Table 5 we investigate the performance of re-parameterization of different block and layer types within the SegFormer encoder. For each row, only the stated blocks or layers are re-parameterized, the rest of the models is kept the same for all $m \in \mathcal{M}$. The location of the specific blocks and layers is depicted in Supplement Section F. It can be seen in Table 5 that the method works best when all parameters of the encoder are used in the re-parameterization. The selection of specific blocks or layers does not bring any advantage. However, all independently evaluated layer / block types yield an improvement over the baseline.

Number of base models: In Figure 4 we show the performance of the re-parameterization vs. various ensembling techniques for a different number M of models. For the ReVT (green), the mIoU is computed with an averaged encoder and all associated M decoders. For the encoder ensemble (blue), the feature maps $\mathbf{z}$ from the encoders are averaged and then processed by all associated M decoders. For the network ensemble, the M output posteriors are averaged (orange) or multiplied (red). For $M > 2$, all methods consistently outperform the baseline (dashed line). In our case, three base models ($M = 3$) provide the best results for the re-parameterization as well as for the network ensemble. The encoder ensemble on the other hand profits from a larger number of base models and yields the best performance for $M = 7$. The re-parameterization outperforms all ensembling techniques for all values of M by at least 1.5% abs. and for $M = 3$ by at least 2.0% abs. It also comes with an M-fold lower computational complexity in inference.

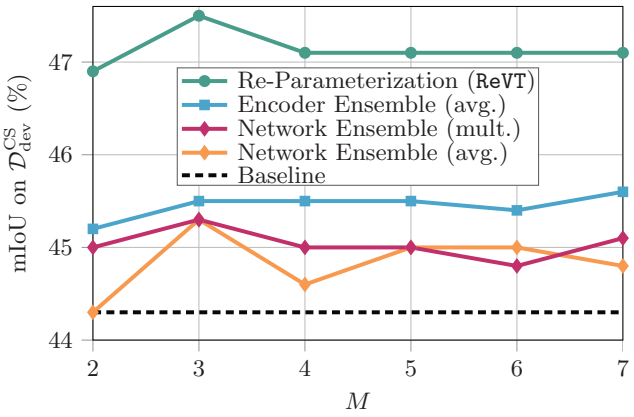

Figure 4. Performance (mIoU (%)) of the **re-parameterization** vs. **network/encoder ensembles** for different numbers M of base models. The **training** of the base models (SegFormer) was performed on the **GTA5** ($\mathcal{D}^{\text{S}} = \mathcal{D}^{\text{GTA5}}_{\text{train}}$) dataset. The evaluation is performed on the **Cityscapes development set** ($\mathcal{D}^{\text{T}} = \mathcal{D}^{\text{CS}}_{\text{dev}}$). The baseline mean is calculated from eight different models ①, and the re-parameterization from M models ①.

5.2. ReVT Method Design

In Table 6 we evaluate various augmentation methods @ (see also Figure 3 and Supplement Section A) to identify strong base models. In the lower part of the table we report some ($M = 3$) combinations $\{@_1, @_2, @_3\}$ of these base models by our re-parameterization. The gray columns indicate our development sets ($\mathcal{D}_{\text{dev}}$), where the light grey column is $\mathcal{D}^{\text{GTA5}}_{\text{dev}}$. Since we train on $\mathcal{D}^{\text{GTA5}}_{\text{train}}$, we select our models on the (dark gray) OOD mean mIoU of $\mathcal{D}^{\text{SYN}}_{\text{dev}}$ and $\mathcal{D}^{\text{CS}}_{\text{dev}}$. We select those base models for further evaluation in the ReVT that performed best, or second-, or third-ranked on the out-of-domain $\mathcal{D}^{\text{SYN}}_{\text{dev}}$ and $\mathcal{D}^{\text{CS}}_{\text{dev}}$ development sets (OOD mean). It can be seen that base models ④, ⑥, and ⑤ yield the best-, second-, third-ranked performance (41.0%, 40.7%, and 40.0%) on $\mathcal{D}^{\text{CS}}_{\text{dev}}$ as well as $\mathcal{D}^{\text{SYN}}_{\text{dev}}$ (out-of-domain data),

Table 6. Performance (mIoU (%)) of the SegFormer model (with an MiT-B5 encoder) using different domain generalization methods. **Training** was performed on the synthetic **GTA5** ($\mathcal{D}^{\mathrm{S}} = \mathcal{D}^{\mathrm{GTA5}}_{\mathrm{train}}$) dataset. **Evaluation** is performed on the **Cityscapes**, **GTA5**, and **SYNTHIA development sets** (gray columns) and on the **test∗ data** of various real-world target datasets ($\mathcal{D}^{\mathrm{T}} = \mathcal{D}_{\mathrm{test}*}$). Reported is the mean mIoU ± the standard deviation of $M = 3$ models with various image augmentations. For the ReVT, the mean ± standard deviation is computed with one averaged encoder and the three associated decoders $m \in \{1, 2, 3\}$. Best results in bold face, second-best underlined.

	Method performed:	**mIoU (%) on**							
		$\mathcal{D}^{\mathrm{GTA5}}_{\mathrm{dev}}$	$\mathcal{D}^{\mathrm{SYN}}_{\mathrm{dev}}$	$\mathcal{D}^{\mathrm{CS}}_{\mathrm{dev}}$	**OOD mean**	$\mathcal{D}^{\mathrm{CS}}_{\mathrm{test}*}$	$\mathcal{D}^{\mathrm{BDD}}_{\mathrm{test}*}$	$\mathcal{D}^{\mathrm{MV}}_{\mathrm{test}*}$	**test∗ mean**
... during training	Baseline ①	68.2±0.0	33.8±0.7	44.3±1.9	39.1±5.4	45.3±1.9	43.3±1.3	46.8±0.9	45.2±2.0
	–PhotoAug ②	68.5±0.1	32.3±0.4	42.0±1.1	37.2±4.9	42.5±1.5	42.3±0.7	45.5±0.9	43.4±1.8
	–PhotoAug, –Rand. Flip ③	**69.0**±0.3	33.0±0.5	42.8±0.8	37.9±4.9	42.3±1.3	41.1±0.8	46.4±0.9	43.3±2.5
	+PixMix* [13] ④	65.1±0.2	**35.4**±1.1	**46.5**±0.3	**41.0**±5.6	46.9±0.8	46.1±1.0	51.2±0.3	48.1±2.4
	+Bilateral Filter (BF) [39]⑤	68.0±0.1	34.3±0.6	45.7±0.3	40.0±5.7	46.8±0.5	44.2±1.1	49.4±1.0	46.8±2.3
	+PixMix* [13] +BF [39] ⑥	64.3±0.1	35.2±0.3	46.2±0.7	40.7±5.5	**47.5**±0.8	**46.7**±0.1	**51.5**±0.4	**48.6**±2.2
...after training	ReVT {①,①,①}	68.6±0.2	35.5±0.4	47.5±0.1	41.5±6.0	49.3±0.1	45.3±0.5	49.3±0.2	48.0±1.9
	ReVT {②,②,②}	69.1±0.1	34.1±0.3	44.4±0.5	39.3±5.2	44.9±0.5	44.1±0.4	47.6±0.4	45.5±1.6
	ReVT {③,③,③}	**69.7**±0.1	35.0±0.4	46.0±0.3	40.5±5.5	45.7±0.2	43.5±0.3	48.8±0.2	46.0±2.2
	ReVT {④,④,④}	65.7±0.1	36.2±0.0	**48.6**±0.3	42.4±6.2	48.8±0.3	47.5±0.2	53.2±0.3	49.8±2.5
	ReVT {⑤,⑤,⑤}	68.5±0.1	35.9±0.0	47.3±0.2	41.6±5.7	48.6±0.3	45.9±0.2	51.2±0.1	48.6±2.2
	ReVT {⑥,⑥,⑥}	64.9±0.0	36.1±0.2	48.0±0.2	42.1±6.0	49.7±0.2	**48.5**±0.4	**53.5**±0.1	**50.5**±2.1
	ReVT {④,⑤,⑥}	66.4±0.7	36.9±0.2	47.9±0.5	42.4±5.5	49.5±0.4	48.1±0.2	53.1±0.2	50.2±2.1
	ReVT {①,④,⑥}	66.4±0.7	**37.3**±0.9	**48.6**±0.8	**42.9**±5.7	**50.0**±0.5	48.0±0.3	52.8±0.2	50.2±2.0

whereas the base models ③ and ② yield the best and second-ranked performance (69.0% and 68.5%) on $\mathcal{D}^{\mathrm{GTA5}}_{\mathrm{dev}}$ (in-domain data). This is to be expected, as the image augmentation makes it harder to learn in the source domain, but forces the base models to generalize slightly, as can be seen in the improved OOD performance of the base models ④, ⑤, and ⑥. In the following, we will report the performance for three different ReVTs. First, the ReVT {①,④,⑥} combines the baseline base model with the two best performing augmentation methods, which leads to the best mean OOD performance (42.9%). Second, the ReVT {④,⑤,⑥} combines the best-, second-, and third-ranked augmentation methods. Third, the ReVT {④,④,④} combines three base models with the single best augmentation method. The later two achieve the second-ranked performance on the OOD data (42.4%).

In the lower part of Table 6 it can be seen that the best test∗ performance can be achieved with a ReVT {⑥,⑥,⑥} leading to a test∗ mean performance of 50.5%. Our best dev set ReVT {①,④,⑥} achieves 50.2% as test∗ mean.

5.3. Comparison to Prior Art DG Methods

In Table 7 we compare our method (ReVT) to prior art methods for domain generalization. We sort methods with respect to their encoder model (Enc.) and give the number of parameters of the full network in the third column. We also indicate whether the methods are trained with only one source domain, or if real auxiliary domains are employed and also report the inference frame rate. Methods are grouped to emphasize that w.r.t. the number of parameters $|\theta|$ and w.r.t. the frame rate, the MiT-B2-based ReVT is competitive to ResNet-50-based methods (group 1), the MiT-B3-based ReVT is competitive to ResNet-101-based methods (group 2), while the MiT-B5-based ReVT builds an own group. Not all methods report the mIoU values for all datasets. In addition to the commonly used datasets, we also evaluate on ACDC [34] and KITTI [1] to provide more evidence of domain generalization on real domains.

It can be seen that *we exceed the performance of prior work that is comparable in network size*. In group 1, our ReVT with the MiT-B2 encoder achieves a benchmark mIoU (BM mean) of 47.29%, excelling the best prior work (WildNet with ResNet-50, 46.33%), while having fewer parameters (27.4 M vs. 43.7 M parameters) and a higher framerate (12 fps vs. 7.9 fps). In group 2, the ReVT with the MiT-B3 achieves a BM mean performance of 50.12%, which is 2.31% abs. higher than the best prior work (WildNet with ResNet-101, 47.81%), while having fewer parameters (47.2 M vs. 62.7 M) and a higher frame rate (10.7

Table 7. Performance (mIoU (%)) of various domain generalization methods employing different segmentation networks, sorted into three performance groups. **Training** was performed on the synthetic **GTA5** ($\mathcal{D}^{\mathrm{S}} = \mathcal{D}^{\mathrm{GTA5}}_{\mathrm{train}}$) dataset. The results marked with $^\circ$ are cited from [20] and with * are cited from the respective paper. All results without any identifier are simulated. **Evaluation** is performed on the **SYNTHIA** and **GTA5 development sets** and on the **test$*$ data** of various real-world target datasets ($\mathcal{D}^{\mathrm{T}} = \mathcal{D}_{\mathrm{test}*}$). BM means benchmark. For our simulations we report mean values over three runs with different seeding. Best performance per group in bold face, second best underlined.

	Enc.	Method	$\lvert\theta\rvert$ ($\cdot 10^6$)	Single Source	Frame Rate [fps]	mIoU (%) on $\mathcal{D}^{\mathrm{CS}}_{\mathrm{test}*}$	$\mathcal{D}^{\mathrm{BDD}}_{\mathrm{test}*}$	$\mathcal{D}^{\mathrm{MV}}_{\mathrm{test}*}$	$\mathcal{D}^{\mathrm{SYN}}_{\mathrm{dev}}$	$\mathcal{D}^{\mathrm{GTA5}}_{\mathrm{dev}}$	$\mathcal{D}^{\mathrm{ACDC}}_{\mathrm{test}*}$	$\mathcal{D}^{\mathrm{KIT}}_{\mathrm{test}*}$	BM mean
Group 1	ResNet-50	Baseline$^\circ$	43.7	✓	7.9	35.16	29.71	31.29	27.97	71.17	-	-	39.06
		IBN-Net$^\circ$ [28]	43.6	✓	8.4	36.52	34.18	38.74	30.41	70.78	-	-	42.12
		RobustNet$^\circ$ [5]	43.6	✓	8.5	38.78	35.64	40.38	28.97	70.16	-	-	42.78
		DRPC* [44]	49.6	✗	8.3	37.42	32.14	34.12	-	-	-	-	-
		SAN+SAW* [30]	25.6	✓	8.1	39.75	37.34	41.86	30.79	-	-	-	-
		WildNet$^\circ$ [20]	43.6	✗	7.9	44.62	38.42	46.09	31.34	**71.20**	-	-	46.33
	MiT-B2	Baseline	27.4	✓	**12.0**	41.73	38.77	44.15	31.20	65.95	30.20	44.34	44.36
		Ours: ReVT {④,④,④}	27.4	✓	**12.0**	45.06	40.44	49.46	**33.29**	62.57	**36.62**	48.94	46.16
		Ours: ReVT {④,⑤,⑥}	27.4	✓	**12.0**	45.55	**43.43**	**49.91**	33.16	63.58	36.66	49.27	47.13
		Ours: ReVT {①,④,⑥}	27.4	✓	**12.0**	**46.27**	43.29	49.84	**33.29**	63.74	36.01	**50.13**	**47.29**
Group 2	ResNet-101	Baseline$^\circ$	62.7	✓	5.1	35.73	34.06	33.42	29.06	71.79	-	-	40.81
		IBN-Net$^\circ$ [28]	62.6	✓	6.0	37.68	36.64	36.75	30.84	70.39	-	-	42.46
		RobustNet$^\circ$ [5]	62.6	✓	6.0	37.26	38.66	38.09	30.17	70.53	-	-	42.94
		DRPC* [44]	68.6	✗	5.3	42.53	38.72	38.05	-	-	-	-	-
		FSDR* [14]	68.6	✗	5.3	44.80	41.20	43.40	-	-	-	-	-
		SAN+SAW* [30]	44.6	✓	5.3	45.33	41.18	40.77	31.84	-	-	-	-
		WildNet$^\circ$ [20]	62.6	✗	5.1	45.79	41.73	47.08	32.51	**71.91**	-	-	47.81
	MiT-B3	Baseline	47.2	✓	**10.7**	43.92	42.96	46.36	32.57	67.59	34.44	45.18	46.68
		Ours: ReVT {④,④,④}	47.2	✓	**10.7**	46.19	46.04	51.39	34.31	64.00	39.16	48.23	48.39
		Ours: ReVT {④,⑤,⑥}	47.2	✓	**10.7**	47.95	**48.26**	**52.59**	**36.80**	64.70	40.96	**49.84**	50.06
		Ours: ReVT {①,④,⑥}	47.2	✓	**10.7**	**48.33**	48.17	52.28	36.67	65.14	**41.38**	49.74	**50.12**
Group 3	MiT-B5	Baseline	84.7	✓	**9.7**	45.31	43.32	46.85	33.81	**68.17**	36.22	46.16	47.49
		Ours: ReVT {④,④,④}	84.7	✓	**9.7**	48.81	47.52	**53.21**	36.18	65.67	39.19	45.86	50.28
		Ours: ReVT {④,⑤,⑥}	84.7	✓	**9.7**	49.55	**48.11**	53.06	36.86	66.38	40.36	46.88	50.79
		Ours: ReVT {①,④,⑥}	84.7	✓	**9.7**	**49.96**	48.01	52.76	**37.27**	66.40	**41.15**	**50.39**	**50.88**

fps vs. 5 fps). In both groups, our method performs slightly worse in the source domain (GTA5), which is included in the BM mean, which, however, has little relevance for practical real-world applications.

It should also be noted, that our method does not employ any real auxiliary domains for image stylization such as WildNet [20]*, DRPC* [44] *and FSDR* [14]. It can also be seen that our largest ReVT with an MiT-B5 encoder (group 3) achieves the overall highest performance of all evaluated models with a BM mean mIoU of 50.88%, still having a higher frame rate than ResNet-50-based WildNet [20], which achieves only a BM mean of 46.33%.

The higher mIoU values on the additional target domains (ACDC and KITTI) further indicate the excellent generalization capability of the ReVTs.

6. Conclusions

In this work we show how to improve the domain generalization capabilities of a vision transformer for semantic segmentation with a simple but effective augmentation and re-parameterization method (ReVT). We show the effect of different image augmentations and optimizer methods on the re-parameterization. Our method is smaller and computationally more efficient than network and encoder ensembles and also achieves state-of-the-art performance in the synthetic-to-real domain generalization task for semantic segmentation, exceeding prior art. In contrast to some prior art, our ReVT does not require an additional real auxiliary domain during training. We achieve a top mean mIoU of 50.88%, when using the largest model and also improve on the best prior art by 0.96% and 2.31% absolute using models with fewer parameters and a higher frame rate.

References

[1] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes. *International Journal of Computer Vision*, 126(9):961–972, Sept. 2018. 4, 7

[2] Nikita Araslanov and Stefan Roth. Self-Supervised Augmentation Consistency for Adapting Semantic Segmentation. In *Proc. of CVPR*, pages 15384–15394, virtual, June 2021. 1, 4

[3] Jan-Aike Bolte, Markus Kamp, Antonia Breuer, Silviu Homoceanu, Peter Schlicht, Fabian Hüger, Daniel Lipinski, and Tim Fingscheidt. Unsupervised Domain Adaptation to Improve Image Segmentation Quality Both in the Source and Target Domain. In *Proc. of CVPR - Workshops*, pages 1404–1413, Long Beach, CA, USA, June 2019. 1

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-Decoder With Atrous Separable Convolution for Semantic Image Segmentation. In *Proc. of ECCV*, pages 801–818, Munich, Germany, Sept. 2018. 1, 4

[5] 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 *Proc. of CVPR*, pages 11580–11590, virtual, June 2021. 2, 4, 5, 16

[6] MMSegmentation Contributors. MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 4

[7] 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 *Proc. of CVPR*, pages 3213–3223, Las Vegas, NV, USA, June 2016. 4, 5

[8] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning Augmentation Strategies From Data. In *Proc. of CVPR*, pages 113–123, Long Beach, CA, USA, June 2019. 2

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. of CVPR*, pages 248–255, Miami, FL, USA, June 2009. 2

[10] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. RepVGG: Making VGG-Style ConvNets Great Again. In *Proc. of CVPR*, pages 13733–13742, virtual, June 2021. 3

[11] Tim Fingscheidt, Hanno Gottschalk, and Sebastian Houben, editors. *Deep Neural Networks and Data for Automated Driving: Robustness, Uncertainty Quantification, and Insights Towards Safety*. Springer Nature, Cham, 2022. 1

[12] Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty. In *Proc. of ICLR*, pages 1–15, virtual, Apr. 2020. 2

[13] Dan Hendrycks, Andy Zou, Mantas Mazeika, Leonard Tang, Bo Li, Dawn Song, and Jacob Steinhardt. PixMix: Dreamlike Pictures Comprehensively Improve Safety Measures. In *Proc. of CVPR*, pages 16783–16792, New Orleans, LA, USA, June 2022. 1, 2, 3, 11

[14] Jiaxing Huang, Dayan Guan, Aoran Xiao, and Shijian Lu. FSDR:Frequency Space Domain Randomization for Domain Generalization. In *Proc. of CVPR*, pages 6891–6902, virtual, June 2021. 2, 4, 5, 8, 16

[15] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging Weights Leads to Wider Optima and Better Generalization. In *Proc. of UAI*, pages 1–10, Monterey, CA, USA, Aug. 2018. 1, 2, 3

[16] Michael Kamp, Linara Adilova, Joachim Sicking, Fabian Hüger, Peter Schlicht, Tim Wirtz, and Stefan Wrobel. Efficient Decentralized Deep Learning by Dynamic Model Averaging. In *Proc. of ECML PKDD*, pages 7393–4090, Dublin, Ireland, Sept. 2018. 3

[17] Myeongjin Kim and Hyeran Byun. Learning Texture Invariant Representation for Domain Adaptation of Semantic Segmentation. In *Proc. of CVPR*, pages 12975–12984, Seattle, WA, USA, June 2020. 2

[18] Marvin Klingner, Mouadh Ayache, and Tim Fingscheidt. Continual BatchNorm Adaptation (CBNA) for Semantic Segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):20899–20911, 2022. 1

[19] Marvin Klingner, Jan-Aike Termöhlen, Jacob Ritterbach, and Tim Fingscheidt. Unsupervised BatchNorm Adaptation (UBNA): A Domain Adaptation Method for Semantic Segmentation Without Using Source Domain Representations. In *Proc. of WACV - Workshops*, pages 210–220, Waikoloa, HI, USA, Jan. 2022. 1, 15, 16

[20] Suhyeon Lee, Hongje Seong, Seongwon Lee, and Euntai Kim. WildNet: Learning Domain Generalized Semantic Segmentation from the Wild. In *Proc. of CVPR*, pages 9936–9946, New Orleans, LA, USA, June 2022. 2, 4, 5, 8, 16

[21] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, Broader and Artier Domain Generalization. In *Proc. of ICCV*, pages 5542–5550, Venice, Italy, Oct. 2017. 2

[22] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M. Hospedales. Episodic Training for Domain Generalization. In *Proc. of ICCV*, pages 1446–1455, Seoul, Korea, Oct. 2019. 2

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proc. of CVPR*, pages 3431–3440, Boston, MA, USA, June 2015. 1

[24] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *Proc. of ICLR*, pages 1–18, New Orleans, LA, USA, May 2019. 2, 13

[25] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain Generalization via Invariant Feature Representation. In *Proc. of ICML*, pages 10–18, Atlanta, GA, USA, June 2013. 2

[26] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In *Proc. of ICCV*, pages 4990–4999, Venice, Italy, Oct. 2017. 4

[27] Viktor Olsson, Wilhelm Tranheden, Juliano Pinto, and Lennart Svensson. ClassMix: Segmentation-Based Data Augmentation for Semi-Supervised Learning. In *Proc. of WACV*, pages 1369–1378, Jan. 2021. 2

[28] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at Once: Enhancing Learning and Generalization Capacities via IBN-Net. In *Proc. of ECCV*, pages 464–479, Munich, Germany, Sept. 2018. 2

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proc. of NeurIPS*, pages 8024–8035, Vancouver, BC, Canada, Dec. 2019. 4

[30] Duo Peng, Yinjie Lei, Munawar Hayat, Yulan Guo, and Wen Li. Semantic-Aware Domain Generalized Segmentation. In *Proc. of CVPR*, pages 2594–2605, New Orleans, LA, USA, June 2022. 2, 15, 16

[31] Fengchun Qiao, Long Zhao, and xi Peng. Learning to Learn Single Domain Generalization. In *Proc. of CVPR*, pages 12556–12565, virtual, June 2020. 2

[32] Stephan Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for Data: Ground Truth from Computer Games. In *Proc. of ECCV*, pages 102–118, Amsterdam, Netherlands, Oct. 2016. 4, 5

[33] 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 *Proc. of CVPR*, pages 3234–3243, Las Vegas, NV, USA, June 2016. 4

[34] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. ACDC: The Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding. In *Proc. of ICCV*, pages 10765–10775, virtual, Oct. 2021. 4, 5, 7

[35] Manuel Schwonberg, Joshua Niemeijer, Jan-Aike Termöhlen, Jörg P. Schäfer, Nico M. Schmidt, Hanno Gottschalk, and Tim Fingscheidt. Survey on Unsupervised Domain Adaptation for Semantic Segmentation for Visual Perception in Automated Driving. *IEEE Access*, 11:54296–54336, 2023. 1

[36] Timo Sämann, Ahmed Mostafa Hammam, Andrei Bursuc, Christoph Stiller, and Horst-Michael Groß. Improving Predictive Performance and Calibration by Weight Fusion in Semantic Segmentation. *arXiv:2207.11211*, July 2022. 1, 3, 4

[37] Antti Tarvainen and Harri Valpola. Mean Teachers are Better Role Models: Weight-Averaged Consistency Targets Improve Semi-Supervised Deep Learning Results. In *Proc. of NIPS*, pages 1–10, Long Beach, CA, USA, Dec. 2017. 3

[38] Jan-Aike Termöhlen, Marvin Klingner, Leon J. Brettin, Nico M. Schmidt, and Tim Fingscheidt. Continual Unsupervised Domain Adaptation for Semantic Segmentation by Online Frequency Domain Style Transfer. In *Proc. of ITSC*, pages 2881–2888, virtual, Sept. 2021. 1

[39] Carlo Tomasi and Roberto Manduchi. Bilateral Filtering for Fray and Color Images. In *Proc. of ICCV*, pages 839–846, Bombay, India, Jan. 1998. 1, 2, 3

[40] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors, July 2022. 1, 3

[41] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model Soups: Averaging Weights of Multiple Fine-Tuned Models Improves Accuracy Without Increasing Inference Time. In *Proc. of ICML*, pages 23965–23998, Baltimore, MD, USA, July 2022. 1, 3

[42] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. In *Proc. of NeurIPS*, pages 12077–12090, virtual, Dec. 2021. 1, 2, 4

[43] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A Diverse Driving Video Database With Scalable Annotation Tooling. *arXiv*, (1805.04687), Aug. 2018. 4

[44] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain Randomization and Pyramid Consistency: Simulation-to-Real Generalization Without Accessing Target Domain Data. In *Proc. of ICCV*, pages 2100–2110, Seoul, Korea, Oct. 2019. 2, 4, 5, 8, 16

[45] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. In *Proc. of ICCV*, pages 6023–6032, Seoul, Korea, Oct. 2019. 2

[46] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond Empirical Risk Minimization. In *Proc. of ICLR*, pages 113–123, Vancouver, Canada, Apr. 2018. 2

[47] Jian Zhang, Lei Qi, Yinghuan Shi, and Yang Gao. Generalizable Semantic Segmentation via Model-Agnostic Learning and Target-Specific Normalization. *arXiv*, (2003.12296), May 2020. 2