Supplementary Material

In this supplementary material we give a detailed overview of the training and evaluation settings along with hyperparameters. We also provide a detailed description of the employed augmentation methods. Further, we show additional ablation studies and investigations. We also depict the `SegFormer` architecture in a block diagram. Finally, we discuss limitations and ethical implications for our method.

A. Detailed Description of Augmentation Methods

In the following section, we present a detailed description of the employed augmentation methods.

Random crop: Parameter ϱ defines the maximum proportion a single class can occupy in the random crop.

PhotoAug: Photometric augmentation (PhotoAug) comprises the following steps: Each transformation is applied to the image with a probability of 0.5. The position of the random contrast adjustment is in second (mode ❶) or second to last position (mode ❷). The position is randomly selected for each image.

1. random brightness
2. if ❶: random contrast
3. convert color from RGB to HSV
4. random saturation
5. random hue
6. convert color from HSV to RGB
7. if ❷: random contrast
8. randomly swap channels

Bilateral filter: A bilateral filter smoothes an image while preserving sharp edges. Its focus is on the removal of noise and textures. In general, it is a Gaussian filter that smoothes less in non-uniform regions (edge regions) and more in uniform image regions (non-edge regions). The bilateral filter at pixel index i can be described as:

$$G(i) = \frac{1}{w} \sum_{j \in \mathcal{J}} N(\Delta_{ij}; \sigma_s^2) N(||\mathbf{x}_i - \mathbf{x}_j||_2; \sigma_c^2) \mathbf{x}_j \ , \quad (3)$$

with

$$N(d; \sigma^2) = e^{-\frac{1}{2}(\frac{d}{\sigma})^2} \ , \quad (4)$$

and normalizing factor

$$w = \sum_{j \in \mathcal{J}} N(\Delta_{ij}; \sigma_s^2) N(||\mathbf{x}_i - \mathbf{x}_j||_2; \sigma_c^2). \quad (5)$$

The neighboring pixel index is denoted as j and stems from the neighborhood $\mathcal{J}$. The neighborhood is defined by the kernel size which we sample from a uniform distribution between 1 (px) and 15 (px). Distances $\Delta_{ij} = \sqrt{|h_i - h_j|^2 + |w_i - w_j|^2}$ and $||\mathbf{x}_i - \mathbf{x}_j||_2$ denote the (Euclidean) pixel distance and color difference, respectively, where h_i and w_i are the height and width position of pixel i and $\mathbf{x}_i \in \mathbb{G}^3$ denotes the vector of RGB values for pixel i (likewise for pixel j). We set the spatial distance to $\sigma_s = 75$ and the color distance to $\sigma_c = 75$. The probability for applying this filter is set to $p = 0.5$.

Listing 1. **Pseudo-code** of the **PixMix** [13] data augmenation.

```
def PixMix(x, z, K, β): # mixing image z ∈ Z
    x̃_0 = random.choice({augment(x), x})
    # random number of mixing rounds
    for k = 1:random.choice({0, 1, ..., K}):
        x_mix = random.choice({augment(x), z})
        mix_op = random.choice({add, multiply})
        x̃_k = mix_op(x̃_{k-1}, x_mix, β)
    return x̃_k
```

PixMix: The PixMix [13] augmentation method comprises multiple processing steps as shown in Listing 1. Here, $\mathbf{x}$ denotes the input image and $\mathbf{z} \in \mathcal{Z}$ denotes the mixing image from the PixMix set of fractal images $\mathcal{Z}$ [13]. We set the maximum number of mixing rounds to $K = 3$. Note that the for loop is not executed for random choice = 0. The **mix_op**$(\cdot)$ function is randomly chosen to be either addition (**add**) or multiplication (**multiply**). It gets the images $\tilde{\mathbf{x}}_{k-1}$ and $\tilde{\mathbf{x}}_{\text{mix}}$ as inputs, as well as $\beta = 3$, which is used to generate independent weighting factors for the images. The weighting factors are sampled from a Beta distribution. For the **augment**$(\cdot)$ function in the PixMix pseudocode we used only baseline augmentation methods PhotoAug and Random Flip (cf. Figure 3), that is why we denote the method as PixMix*.

B. Training/Evaluation Settings, Hyperparameters

In the following section, we will provide a detailed overview of the training and evaluation settings and hyperparameters. For the training and evaluation we employ `PyTorch` v.3.8.13 and the `MMSegmentation` toolbox v.0.11.0. Additionally, we refer to our repository, where all code for the conducted experiments is made available[3].

Training phase: In Table 9 we list all settings and hyperparameters that were used for the training process. The polynomial learning rate schedule is defined as follows:

$$\eta(\tau) = \eta_0 (1 - \frac{\tau}{\tau_{\text{max}}})^{0.9}, \quad (6)$$

with $\eta(\tau)$ being the learning rate at optimizer step (iteration) τ and η_0 being the initial learning rate. The maximum number of iterations is given by τ_{max}.
During training, the images from the source domain $\mathcal{D}^{\text{S}}$ get resized to a resolution of 720×1280.

Evaluation phase: For evaluation, we always employ the final model weights after the full training and do not perform any checkpoint selection. We resize the input images

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

Table 9. **Settings and hyperparameters** for the `SegFormer` and `DeepLabv3+` training.

Setting / Hyperparameter	`SegFormer`	`DeepLabv3+`
Optimizer	AdamW [24]	SGD
# of training iterations ($\tau_{\max}$)	40,000	60,000
Momentum values (AdamW) (β_1, β_2)	0.9, 0.999	-
Momentum (β)	-	0.9
Warm-up iterations	1500	-
Warm-up ratio	$1{\cdot}10^{-6}$	-
Initial LR (η_0)	$6{\cdot}10^{-5}$	$1{\cdot}10^{-3}$
Weight decay ω	0.01	0.0005
Learning rate (LR) schedule ($\eta(\tau)$)	Polynomial (6)	Polynomial (6)
Batch size	2	2
Random decoder init	Kaiming initialization	Kaiming initialization
Resized input resolution $\mathcal{D}_{\text{train}}^{\text{GTA5}}$	720×1280	720×1280

Table 8. **Image and label resolution** [px × px] for the employed **evaluation** datasets. °In both GTA and KITTI, there are images that differ by a few pixels from their normal resolution. *The crowd-sourced Mapillary Vistas dataset does not have a fixed resolution, but a highly variable one.

Dataset name	**Resolution ($H\times W$) of ...**	
	resized images	**labels**
GTA5 [32]	$512\times\ 932°$	$1052\times1914°$
SYNTHIA [33] (SYN)	$512\times\ 862$	760×1280
Cityscapes [7] (CS)	512×1024	1024×2048
Mapillary Vistas [26] (MV)	various*	various*
BDD100k [43] (BDD)	$512\times\ 910$	720×1280
ACDC [34]	$512\times\ 910$	1080×1920
KITTI [1] (KIT)	$309\times1024°$	$375\times1242°$

during evaluation in a way that the image will be rescaled as large as possible within a pre-defined scale (512×1024), while still keeping their aspect ratios. The network output is then resized to the original image resolution. The mIoU is calculated on the original resolution, also referred to as label resolution. The resized image and the original label resolutions for all employed datasets are listed in Table 8.

The frame rate computations were performed on the rescaled Cityscapes dataset. We used 200 images for inference and computed the mean frame rate after a warmup phase of five images to account for any delays due to image reading operations.

C. Additional Details on the Choice of Optimizer

In our experiments in Table 4 we show that the choice of optimizer has a strong effect on the baseline performance of the models, as well as on the performance after re-parameterization. In Figure 5, we compare the `SegFormer` architecture with its standard optimizer setup (left) and the `DeepLabv3+` architecture with its standard optimizer setup (right). We show the mean cosine similarity between three baseline (①) models for the encoder only ($\boldsymbol{\theta}^{\text{E}}$, upper plots) and the full model ($\boldsymbol{\theta}$, center plots), during the training process. Note that the standard number of iterations differs for both models and is 40,000 for the `SegFormer` and 60,000 for the `DeepLabv3+`.

It can be seen that the mean cosine similarity for the network parts that are re-parameterized in our method (encoder only) have a similar mean cosine similarity after the training for both networks (0.995). In the bottom plots of the figure, we report the mIoU values for both in-domain (GTA5, green) and out-of-domain (OOD) data (Cityscapes, red) for the baseline (dashed lines) and re-parameterized (solid lines) models. It can be seen that the performance of the re-parameterized models is higher for the `SegFormer` for any training iteration. For the `DeepLabv3+`, however, the baseline performance is always higher for in-domain data (green) and fluctuates for OOD data (red), but ultimately the baseline performance is also higher for OOD data in the last iterations.

Since the mean cosine similarity did not provide any insights into the causes for the poor performance of the re-parameterized `DeepLabv3+`, we further investigated the mean cosine similarity for individual layers ℓ of the networks as shown in Figure 6. We show the layer-wise mean cosine similarity for the encoder network ($\boldsymbol{\theta}^{\text{E}}_{\ell}$), where ℓ indicates the layer index. For the purpose of clarity, we only mark the first layer of each of the major network blocks. For the `SegFormer`, we indicate the transformer blocks by $\mathsf{B}b$ with b being the block index (cf. Fig-

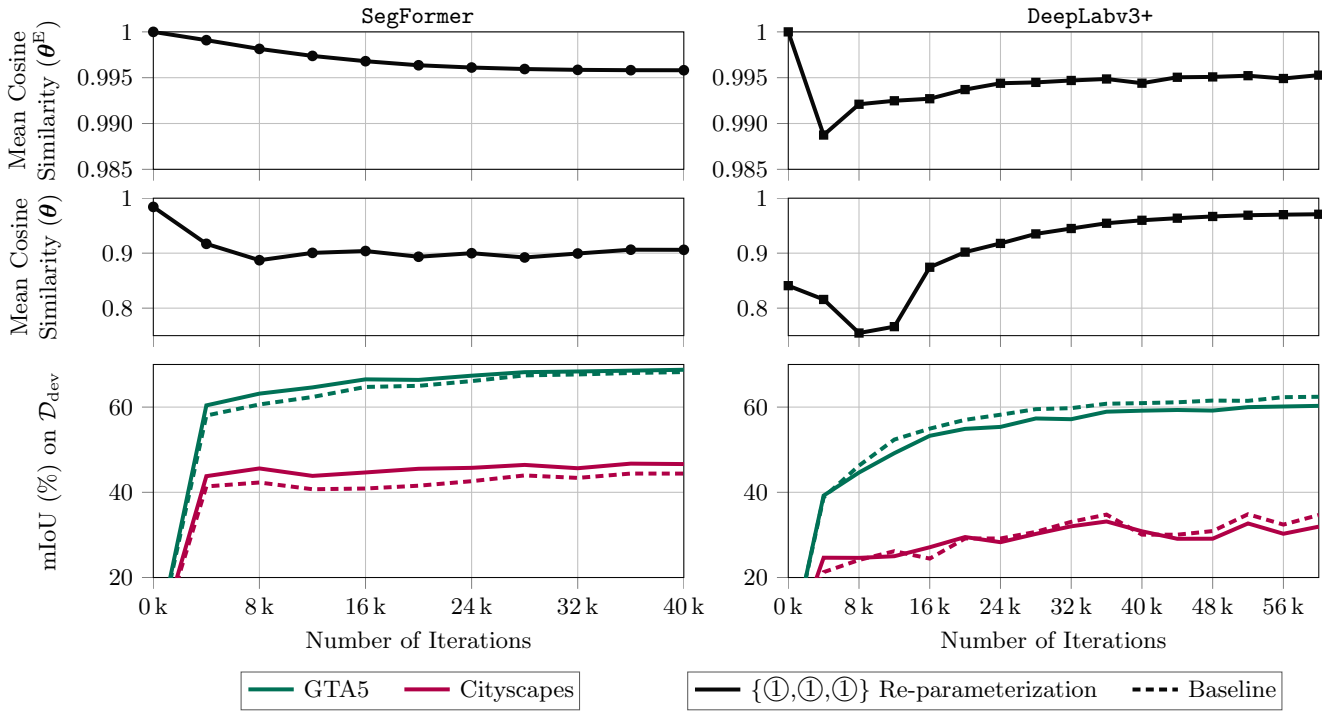

Figure 5. Comparison of **mean cosine similarity vs. mIoU performance** for `SegFormer` and `DeepLabv3+` **during training**. The first row shows the mean cosine similarity for the encoder only ($\boldsymbol{\theta}^{\mathrm{E}}$), the second row for the full network ($\boldsymbol{\theta}^{\mathrm{E}}$). The mean cosine similarity is computed between three models. In the bottom row, the mIoU is given for the dev sets of GTA5 (green) and Cityscapes (red).

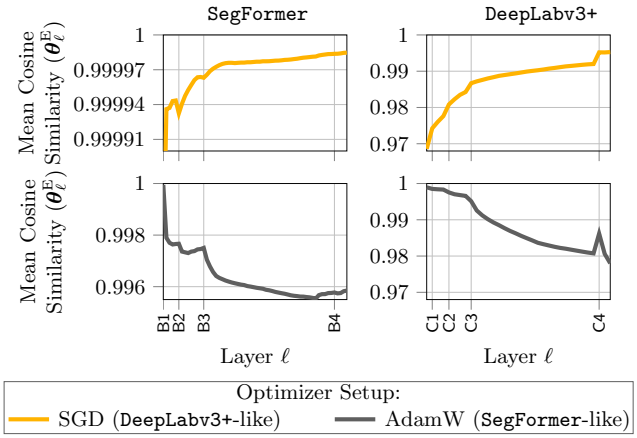

Figure 6. Comparison of the **layer-wise mean cosine similarity** for the encoder only ($\boldsymbol{\theta}_\ell^{\mathrm{E}}$). Shown are results for the `SegFormer` (left) and `DeepLabv3+` (right) architectures, which were trained with the standard `DeepLabv3+` or `SegFormer` optimizer setup, shown in gray and yellow, respectively. The mean cosine similarity is computed between three models. For the purpose of clarity, we only indicate the individual network blocks (Bb for `SegFormer`, Cc for `DeepLabv3+`) on the x axis.

ure 8). For the `DeepLabv3+`, we indicate the convolutional blocks, as defined by He et al. [48], by Cc with c being the block index. *It can be seen that the choice of optimizer has a significant effect on the layer-wise cosine similarity.* For the standard `DeepLabv3+` optimizer setup with SGD, the cosine similarity for both network architectures is higher in deeper layers and lower in earlier layers. In contrast, when the standard `SegFormer` optimizer setup with AdamW is employed, the cosine similarity is highest for earlier layers and drops for deeper layers. This specific property might be important for a well-performing encoder re-parameterization. As already shown in Table 4, this optimizer setup (AdamW [24]) also allows the `DeepLabv3+` to improve over its baseline performance. Accordingly, we used the AdamW optimizer for our `ReVT` method in the main paper.

D. Additional Ablation Studies

In this section, we will investigate the weighting of the base models and compare the `ReVT` 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.

Weighting of networks: In Figure 7 we depict multiple possible weighting combinations for three models with the best combination (marked with a blue circle) achieving an mIoU of 47.68%, while the uniform re-parameterization

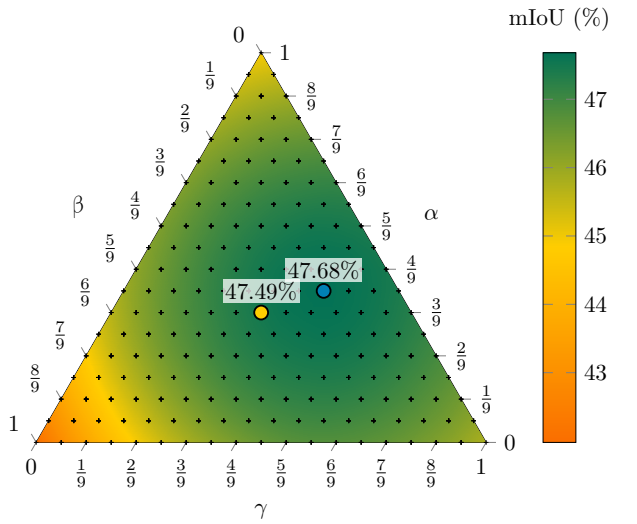

Figure 7. Ternay plot showing the performance (mIoU (%)) of three baseline models {①,①,①} and multiple weight combinations (α, β, γ). The mIoU is calculated by a re-parameterization, where the parameters of the three models are weighted by the values of α, β, and γ, respectively. The **training** of the base models (`SegFormer`) was performed on the synthetic **GTA5** training set ($\mathcal{D}^{\text{S}} = \mathcal{D}^{\text{GTA5}}_{\text{train}}$). The **evaluation** is performed on the **Cityscapes development set** ($\mathcal{D}^{\text{T}} = \mathcal{D}^{\text{CS}}_{\text{dev}}$). The center of the plot ($\alpha = \beta = \gamma = \frac{1}{3}$) corresponds to a uniform encoder re-parameterization.

(all models weighted with $\frac{1}{3}$, marked with a yellow circle) achieves an mIoU of 47.49%. It can be seen that the weighting of the individual models is actually quite insensitive, which is why we decide to use the simple variant of the uniform encoder re-parameterization (i.e., weights $\frac{1}{3}$).

`ReVT` vs. ensembles: In Table 10 we compare the `ReVT` and a network ensemble for different combinations of training settings. It can be seen that the `ReVT` method outperforms the network ensemble not only for the combination of three baseline models {①,①,①}, as already shown in Figure 4, but also for all other tested base model combinations {ⓐ₁, ⓐ₂, ⓐ₃} on the real test∗ sets. However, the network ensemble is slightly better in the synthetic source domain ($\mathcal{D}^{\text{GTA5}}_{\text{dev}}$), which has little relevance for real-world applications. The same applies to the base model combination {⑤,⑤,⑤}, where the ensemble performs slightly better on the SYNTHIA dataset ($\mathcal{D}^{\text{SYN}}_{\text{dev}}$). The overall best performance for each dataset is highlighted in light green. It can be seen that the `ReVT` method achieves top performance for each dataset and is even on par with the ensemble on the source dataset ($\mathcal{D}^{\text{GTA5}}_{\text{dev}}$). In general, once again, the `ReVT` {①,④,⑥} seems to be a strong `ReVT`, generalizing well to unseen datasets.

Table 10. Performance (mIoU (%)) of a **network ensemble** vs. the **`ReVT`** for different base model combinations. 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 **GTA5** and **SYNTHIA development sets** and on the **test∗ data** of various real-world target datasets ($\mathcal{D}^{\text{T}} = \mathcal{D}_{\text{test}*}$). Reported is the mean value over the respective number of employed base models. Best result for each each base model combination in bold face, overall best performance per dataset is highlighted in light green.

Base Models	Method	mIoU (%) on				
		$\mathcal{D}^{\text{GTA5}}_{\text{dev}}$	$\mathcal{D}^{\text{SYN}}_{\text{dev}}$	$\mathcal{D}^{\text{CS}}_{\text{test}*}$	$\mathcal{D}^{\text{BDD}}_{\text{test}*}$	$\mathcal{D}^{\text{MV}}_{\text{test}*}$
{①,①,①}	Ensemble	**68.8**	34.6	46.9	44.8	48.0
	`ReVT`	68.6	**35.5**	**49.3**	**45.3**	**49.3**
{②,②,②}	Ensemble	**69.2**	33.1	43.8	43.9	46.6
	`ReVT`	69.1	**34.1**	**44.9**	**44.1**	**47.6**
{③,③,③}	Ensemble	**69.7**	33.8	43.8	42.6	47.3
	`ReVT`	**69.7**	**35.0**	**45.7**	**43.5**	**48.8**
{④,④,④}	Ensemble	**65.9**	**36.8**	47.5	47.4	52.6
	`ReVT`	65.7	36.2	**48.8**	**47.5**	**53.2**
{⑤,⑤,⑤}	Ensemble	**68.6**	35.1	47.5	45.4	50.4
	`ReVT`	**68.5**	**35.9**	**48.6**	**45.9**	**51.2**
{⑥,⑥,⑥}	Ensemble	**65.0**	**37.0**	48.0	47.9	52.8
	`ReVT`	64.9	36.1	**49.7**	**48.5**	**53.5**
{④,⑤,⑥}	Ensemble	**67.2**	35.1	48.1	47.3	51.6
	`ReVT`	**66.4**	**36.9**	**49.5**	**48.1**	**53.1**
{①,④,⑥}	Ensemble	**68.5**	34.2	47.0	45.9	50.1
	`ReVT`	66.4	**37.3**	**50.0**	**48.0**	**52.8**

E. `ReVT` with SYNTHIA as Source

In this section, we provide additional results for our method, when trained with the SYNTHIA dataset as source domain: $\mathcal{D}^{\text{S}} = \mathcal{D}^{\text{SYN}}_{\text{train}}$. In Table 11 we evaluate the various augmentation methods ⓐ that were already evaluated for models trained on GTA5 ($\mathcal{D}^{\text{S}} = \mathcal{D}^{\text{SYN}}_{\text{train}}$) in Section 5. 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 $\mathcal{D}^{\text{SYN}}_{\text{dev}}$. The OOD mean mIoU of $\mathcal{D}^{\text{GTA5}}_{\text{dev}}$ and $\mathcal{D}^{\text{CS}}_{\text{dev}}$ is shown in the dark grey columns. It can be seen in the upper part of Table 11 that the augmentation methods do not improve the performance as much as for models trained on GTA5. The best OOD mean performance is achieved with the baseline model ①. On the test∗ mean the PixMix* augmentation works best, followed by the combination of PixMix* and the bilateral filter, and baseline model.

Although the individual augmentations do not perform as well for these models, the `ReVT` {①,④,⑥}*, which we*

Table 11. Performance (mIoU (%)) of the SegFormer model (with an MiT-B5 encoder) using different domain generalization methods. **Training** was performed on the synthetic **SYNTHIA** ($\mathcal{D}^{\mathrm{S}} = \mathcal{D}^{\mathrm{SYN}}_{\mathrm{train}}$) 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 $m \in \{1, 2, 3\}$. For models trained on SYNTHIA, we evaluate over 16 classes, as is common practice [19]. Best results in bold face, second-best underlined.

	Method performed:	**mIoU (%) on**							
		$\mathcal{D}^{\mathrm{SYN}}_{\mathrm{dev}}$	$\mathcal{D}^{\mathrm{GTA5}}_{\mathrm{dev}}$	$\mathcal{D}^{\mathrm{CS}}_{\mathrm{dev}}$	**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	76.5±0.1	42.8±0.4	**44.3**±1.3	**43.5**±1.2	**45.1**±1.6	35.2±1.4	**42.5**±0.9	40.9±4.4
	−PhotoAug ②	77.3±0.1	39.8±0.6	41.4±0.5	40.6±0.9	41.6±0.8	33.7±1.0	40.8±0.6	38.7±3.6
	−PhotoAug, −Rand. Flip ③	**78.3**±0.0	40.7±0.7	41.3±0.3	41.0±0.6	41.8±0.2	34.3±1.1	41.3±0.3	39.1±3.5
	+PixMix* [13] ④	73.8±0.1	**43.1**±1.0	42.6±1.7	42.8±1.4	42.6±2.2	**38.4**±0.8	**42.5**±0.6	**41.2**±2.4
	+Bilateral Filter (BF) [39]⑤	76.0±0.1	42.9±0.9	43.2±0.7	43.1±0.8	43.1±0.9	36.0±0.7	41.7±0.4	40.3±3.1
	+PixMix* [13] +BF [39] ⑥	72.3±0.1	41.2±0.6	43.3±0.4	42.2±1.2	43.0±0.5	38.0±0.6	41.6±0.3	40.9±2.2
...after training	ReVT {①,①,①}	76.2±0.1	42.3±0.1	44.9±0.4	43.6±1.3	45.8±0.4	35.8±0.2	43.6±0.2	41.7±4.3
	ReVT {②,②,②}	76.9±0.0	40.4±0.2	42.2±0.2	41.3±1.0	42.6±0.1	34.7±0.3	42.0±0.3	39.8±3.6
	ReVT {③,③,③}	**78.0**±0.1	41.3±0.6	41.9±0.3	41.6±0.6	42.5±0.4	35.2±0.3	42.7±0.3	40.1±3.5
	ReVT {④,④,④}	73.7±0.1	43.4±0.1	44.1±0.3	43.7±0.4	44.3±0.3	39.5±0.3	44.0±0.3	42.6±2.2
	ReVT {⑤,⑤,⑤}	75.7±0.0	**44.1**±0.7	44.4±0.3	**44.2**±0.6	44.5±0.3	37.1±0.3	43.0±0.2	41.5±3.2
	ReVT {⑥,⑥,⑥}	72.2±0.1	42.4±0.2	44.4±0.2	43.4±1.0	44.3±0.2	38.9±0.2	42.9±0.0	42.0±2.3
	ReVT {④,⑤,⑥}	74.0±0.6	43.2±0.4	45.0±0.4	44.1±0.9	45.1±0.4	39.6±0.4	44.0±0.2	42.9±2.4
	ReVT {①,④,⑥}	74.1±0.6	42.7±0.3	**45.7**±0.5	**44.2**±1.5	**46.3**±0.3	**40.3**±0.5	**44.8**±0.1	**43.8**±2.6

already identified as our best ReVT *in Section 5, provides both top OOD mean (44.2%) and test∗ mean (43.8%) performance.* Again, second-best results are achieved with the ReVT {④,⑤,⑥}. Additionally, the ReVT {⑤,⑤,⑤} is on par with the ReVT {①,④,⑥} for the OOD mean performance when trained on SYNTHIA.

In the following, we compare also against prior art that have also been evaluated with SYNTHIA as source domain. Again, we choose the ReVT {①,④,⑥} and ReVT {④,⑤,⑥} for the comparison with prior art. The results are shown in Table 12. In contrast to the models trained on GTA5, for models trained on SYNTHIA, the ReVT {①,④,⑥} does not always reach the top BM mean performance. Similar to the GTA5-trained models, the performance of both ReVT variants remains slightly behind that of the baseline for synthetic source domain data ($\mathcal{D}^{\mathrm{SYN}}_{\mathrm{dev}}$), which, however, has little relevance for practical real-world applications.

For the small (group 1) and midsized (group 2) models, the ReVT {④,⑤,⑥} yields a slightly better performance of 45.79% vs. 45.44% (baseline: 44.09%) and 48.99% vs. 48.84% (baseline: 46.72%), respectively. For the large models (group 3), the ReVT {①,④,⑥} yields the best performance with a BM mIoU of 49.64% (baseline: 48.42%). In summary, on the benchmark (BM) data, our proposed SYNTHIA-trained ReVT models achieve an mIoU improvement of +1.2% absolute (large models) to +1.7% absolute (small models).

It should be noted that no prior work reported on all datasets necessary for the benchmark (BM) mean when trained with SYNTHIA as source domain. All of our ReVTs improve on the prior art for the reported domains. Only for the smallest models in group 1 the SAN+SAW method [30] achieves a higher mIoU on the BDD dataset (best prior art: 35.42% vs. ours: 35.18%). For the midsized models we already improve on this domain (best prior art: 37.40% vs. ours: 38.73%), and interestingly significantly excel the SAN+SAW method (ours: 38.73% vs. SAN+SAW: 35.98%).

F. SegFormer Block Diagrams

In Section 5 we investigated the effect of the reparameterization on different network parts (cf. Table 3) and block or layer types (cf. Table 5). To give the reader a better idea of how the network is structured and where the individual block and layer types are located in the network, an hierarchically illustrated overview of the SegFormer architecture with an MiTB5 encoder is given in Figures 8, 9,

Table 12. Performance (mIoU (%)) of various domain generalization methods employing different segmentation networks, sorted into three performance groups. **Training** was performed on the synthetic **SYNTHIA** ($\mathcal{D}^{\mathrm{S}} = \mathcal{D}^{\mathrm{SYN}}_{\mathrm{train}}$) 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. For models trained on SYNTHIA, we evaluate over 16 classes, as is common practice [19]. 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$^{\bullet}$	49.6	✓	7.9	28.36	25.16	27.24	-	-	-	-	-
		DRPC$^{\bullet}$ [44]	49.6	✗	8.3	35.65	31.53	32.74	-	-	-	-	-
		SAN+SAW* [30]	25.6	✓	8.1	38.92	**35.42**	34.52	-	29.16	-	-	-
	MiT-B2	Baseline	27.4	✓	**12.0**	39.71	29.76	38.37	**74.78**	37.83	26.16	35.18	44.09
		Ours: ReVT {④,⑤,⑥}	27.4	✓	**12.0**	**41.09**	35.18	40.21	71.59	**40.88**	**30.39**	34.64	**45.79**
		Ours: ReVT {①,④,⑥}	27.4	✓	**12.0**	40.91	34.53	**40.44**	71.45	39.87	30.13	**35.29**	45.44
Group 2	ResNet-101	Baseline$^{\bullet}$	68.6	✓	7.9	29.67	25.64	28.73	-	-	-	-	-
		DRPC$^{\bullet}$ [44]	68.6	✗	5.3	37.58	34.34	34.12	-	-	-	-	-
		FSDR* [14]	68.6	✗	5.3	40.80	37.40	39.60	-	-	-	-	-
		SAN+SAW* [30]	44.6	✓	5.3	40.87	35.98	37.26	-	30.79	-	-	-
	MiT-B3	Baseline	47.2	✓	**10.7**	42.43	33.33	40.47	**75.82**	41.53	29.73	35.91	46.72
		Ours: ReVT {④,⑤,⑥}	47.2	✓	**10.7**	**45.26**	**38.73**	42.86	73.12	**44.99**	**35.27**	**36.42**	**48.99**
		Ours: ReVT {①,④,⑥}	47.2	✓	**10.7**	44.97	38.65	**43.00**	73.16	44.42	35.16	36.31	48.84
Group 3	MiT-B5	Baseline	84.7	✓	**9.7**	45.07	35.19	42.51	**76.49**	42.82	30.81	37.02	48.42
		Ours: ReVT {④,⑤,⑥}	84.7	✓	**9.7**	45.08	39.62	43.99	73.96	**43.25**	35.12	37.20	49.18
		Ours: ReVT {①,④,⑥}	84.7	✓	**9.7**	**46.28**	**40.30**	**44.76**	74.11	42.74	**35.75**	**37.86**	**49.64**

10, 11, and 12.

G. Discussion of Limitations

Although modern methods for domain generalization provide good performance on completely unseen real data (after training on synthetic data), the performance still remains behind that of modern methods for unsupervised domain adaptation (UDA) [47,49]. Such a comparison, however, is not entirely fair, since UDA methods employ unlabeled data from a target domain (typically Cityscapes) during the training process, which we intentionally avoid in domain generalization. Nevertheless, it should be noted that better performance on a specific target domain can be achieved, if samples from this domain are available during training.

Our proposed method cannot be applied advantageously to any already trained model, since the optimizer choice has a significant impact on the performance. To be fair, however, this is the case with all prior art methods as well. Most of them additionally extend the training process considerably, far beyond the choice of the optimizer [5, 14, 20, 30, 44].

H. Discussion of Ethical Implications

Although well generalizing semantic segmentation has many civilian applications that provide great value to society, e.g., automated driving, robotics, and medical applications, this technology can also be used for military and surveillance applications. Research on better generalizing methods may also indirectly contribute to the improvement of these applications.

Another aspect to consider are biases in the employed datasets. Three of the five real datasets (Cityscapes, ACDC, KITTI) were captured in Central Europe, one in the USA (BDD100k), and only one contains data from all over the world (Mapillary Vistas). This may lead to biases regarding different ethnicities in the data, which were not investigated further in this paper. For the reported results on improved generalization from synthetic to real data, the biases may be negligible, but should be considered for possible real-world applications.

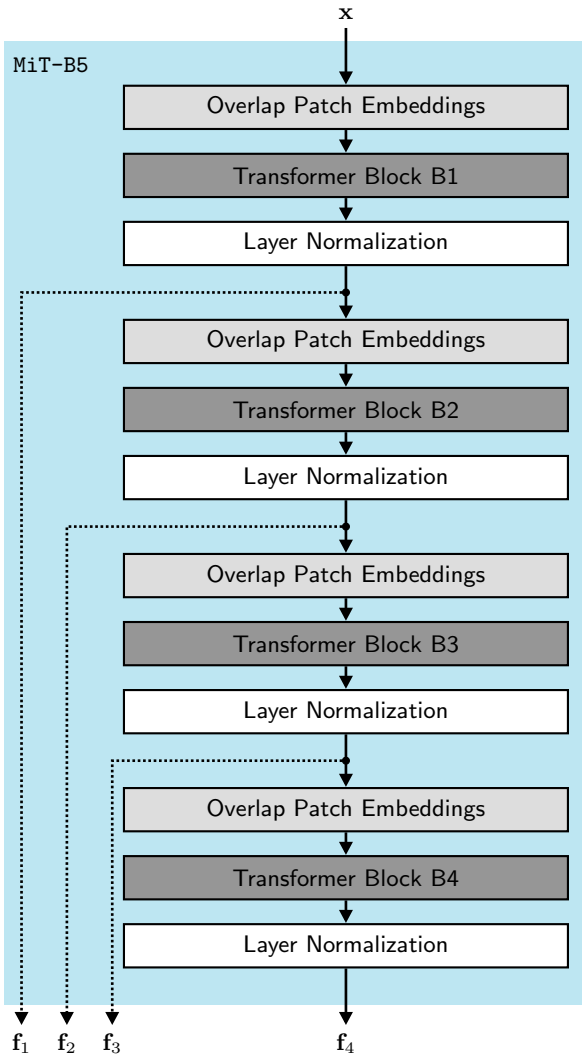

Figure 8. Overview of the `MiT-B5` encoder that is employed by the largest `SegFormer` model. This is the standard encoder employed in the segmenation model (cf. Figure 2).

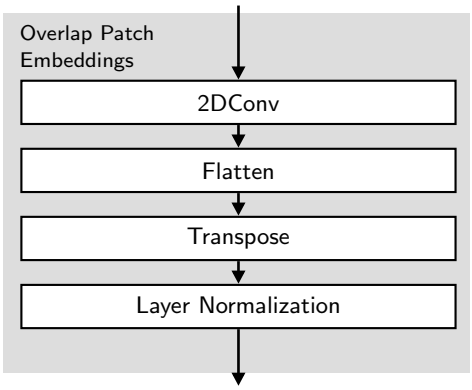

Figure 9. Overview of the overlap patch embeddings block that is employed in the `MiT` encoder (cf. Figure 8 for `MiT-B5`).

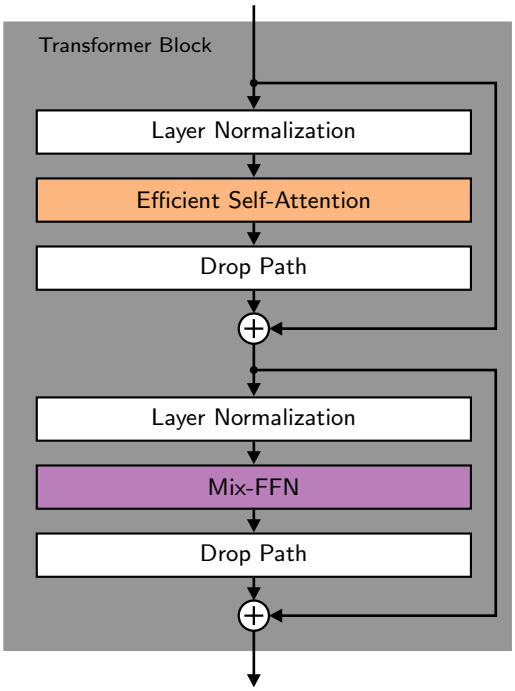

Figure 10. Overview of the transformer block that is employed in the `MiT` encoder (cf. Figure 8 for `MiT-B5`).

Additional References

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

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proc. of* CVPR, pages 770–778, Las Vegas, NV, USA, June 2016

[49] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. DAFormer: Improving Network Architectures and Training Strategies for Domain-Adaptive Semantic Segmentation. In *Proc. of* CVPR, pages 9924–9935, New Orleans, LA, USA, June 2022

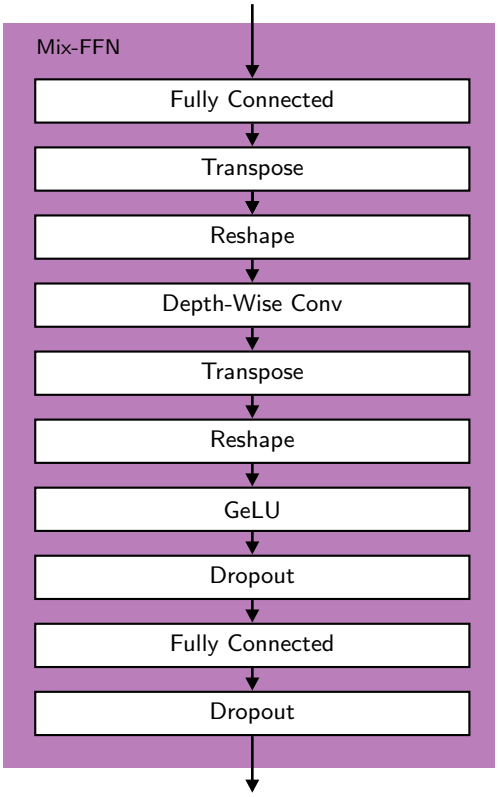

Figure 11. Overview of the Mix-FFN block that is employed in the transformer block (cf. Figure 10).

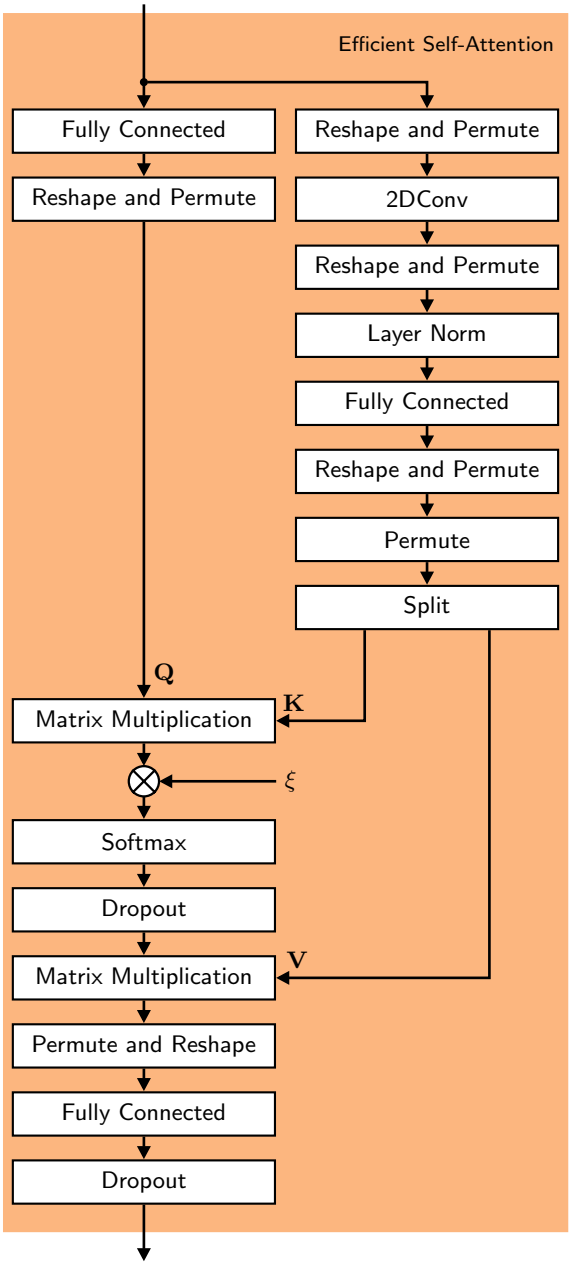

Figure 12. Overview of the efficient self-attention block that is employed in the transformer block (cf. Figure 10). The fixed scaling factor ξ is a hyperparameter and block-dependent.