Semantic Self-adaptation: Enhancing Generalization with a Single Sample

Sherwin Bahmani*, Oliver Hahn*, Eduard Zamfir*†, Nikita Araslanov†, Daniel Cremers and Stefan Roth

Abstract—Despite years of research, out-of-domain generalization remains a critical weakness of deep networks for semantic segmentation. Previous studies relied on the assumption of a static model, *i. e.* once the training process is complete, model parameters remain fixed at test time. In this work, we challenge this premise with a *self-adaptive* approach for semantic segmentation that adjusts the inference process to each input sample. Self-adaptation operates on two levels. First, it employs a self-supervised loss that customizes the parameters of convolutional layers in the network to the input image. Second, in Batch Normalization layers, self-adaptation approximates the mean and the variance of the entire test data, which is assumed unavailable. It achieves this by interpolating between the training and the reference distribution derived from a single test sample. To empirically analyze our self-adaptive inference strategy, we develop and follow a rigorous evaluation protocol that addresses serious limitations of previous work. Our extensive analysis leads to a surprising conclusion: Using a standard training procedure, self-adaptation significantly outperforms strong baselines and sets new state-of-the-art accuracy on multi-domain benchmarks. Our study suggests that self-adaptive inference may complement the established practice of model regularization at training time for improving deep network generalization to out-of-domain data.

1 Introduction

The current state of the art for semantic segmentation [11], [37] lacks direly in out-of-distribution robustness, *i. e.* when the training and testing distributions are different. Numerous studies have investigated this issue, with a primary focus on image classification [2], [5], [34], [59], [63]. However, a recent study of existing domain generalization methods [22] comes to a sobering conclusion: Empirical Risk Minimization (ERM), which makes the *i. i. d.* assumption of the training and testing samples, is still highly competitive. This is in stark contrast to the evident advances in the area of domain adaptation, both for image classification [4], [20], [38], [72] and semantic segmentation [1], [64], [73]. This setup, however, assumes access to an unlabelled test distribution at training time, whereas in the generalization setting considered here, *only one test sample is accessible at inference time*, and *no knowledge between the subsequent test samples must be shared* [55].

In this work, we study the generalization problem of semantic segmentation from synthetic data [50], [51] through the lens of adaptation. In sharp contrast to previous work that modified the model architecture [45] or the training process [12], [13], [76], we leave the training stage unchanged, but replace the standard inference procedure with a technique inspired by domain adaptation methods [1], [35]. The technique, that we term *self-adaptation*, leverages a self-supervised loss, which allows for adapting to a single test sample with a few parameter updates. Complementary to these loss-based updates, self-adaptation integrates feature statistics of the training data with those of the test sample in the Batch Normalization layers [27], commonly employed in modern convolutional neural networks (CNNs) [24]. Expanding upon the previous conclusions in related studies [52], we find that this normalization strategy not only improves the segmentation accuracy, but also the calibration quality of the prediction confidence.

In summary, our contributions are the following:

- We formulate model generalization as a novel self-adaptive process, whereby model inference adjusts to each input sample.
- We develop a self-supervised loss that allows convolutional layers to adapt their parameters to a test image with a few update iterations.
- We analyse and expand upon previous studies of Batch Normalization in the context of out-of-distribution generalization. This allows us to devise an efficient self-adaptive normalization (SaN) strategy, which substantially improves model accuracy and calibration on the out-of-distribution data.
- We revisit the experimental protocol used in previous studies and highlight important deficiencies that may lead to distorted conclusions. Following the best practice in related domains [22], we propose a rigorous revision, which we follow in our experimentation.
- We conduct an extensive analysis of our self-adaptation process, which reveals a consistent and significant improvement in out-of-distribution accuracy over the baseline in *all benchmark scenarios*. Self-adaptation yields a new state of the art in segmentation accuracy, substantially surpassing that of previous work in virtually all considered settings.

2 Related work

Our work contributes to recent research on generalization of semantic segmentation models, and relates to the studies on feature normalization [45], [52] and online learning [55]. While the focus in previous investigations was the training

* *Equal contribution;* † *work primarily done while at TU Darmstadt.*
- *S. Bahmani, O. Hahn and S. Roth are with TU Darmstadt (Germany).*
- *E. Zamfir is with University of Würzburg (Germany).*
- *N. Araslanov and D. Cremers are with TU Munich (Germany).*
- *Code and pre-trained models: https://github.com/visinf/self-adaptive.*

strategy [76] and model design [45], we exclusively study the test-time inference process here. Yue *et al.* [76] augmented the synthetic training data by transferring style from real images. Assuming access to a classification model trained on real images, Chen *et al.* [13] regularize the training on synthetic data by ensuring feature proximity of the two models via distillation, and seek layer-specific learning rates for improved generalization. Advancing the distillation technique, Chen *et al.* [12] devise a contrastive loss that facilitates model invariance to standard image augmentations. Pan *et al.* [45] heuristically add instance normalization (IN) layers to the network. More recently, Choi *et al.* [14] and Huang *et al.* [25] extract domain-invariant feature statistics by either using an instance-selective whitening loss or frequency-based domain randomization. Kundu *et al.* [32] increase source domain diversity by augmenting single domain data to virtually simulate a multi-source scenario. Tang *et al.* [56] swap channel-wise statistics in feature normalization layers, and learn adapter functions to re-adjust the mean and the variance based on the input sample. Lee *et al.* [33] exploit additional ImageNet [17] samples to diversify style and content of the source domain. Peng *et al.* [47] align feature statistics on the category-level to encourage intra-category compactness and inter-category separability. Kim *et al.* [30] assign class-based semantic knowledge into external memory for robust semantic segmentation on unseen domains. These learning strategies nonetheless rely on the *i.i.d.* assumption violated by the out-of-distribution setting. Furthermore, these methods assume access to a *distribution* of real images during training [12], [13], [76] (as opposed to only for pre-training of the backbone), or require a modification of the network architecture [45]. Our work requires neither, hence the presented technique applies even *post-hoc* to the already (pre-) trained models to improve their generalization. Moreover, as we discuss and address in Sec. 5, the evaluation protocol used by previous studies exhibits a number of shortcomings.

Normalization. Batch Normalization (BN) [27] and other normalization techniques have been increasingly linked to model robustness [16], [26], [52], [66], [70]. The most commonly used BN, Layer Normalization (LN) [3], and Instance Normalization (IN) [60] also affect the model's expressive power, which can be further enhanced by combining these techniques in one architecture [39], [43]. In a domain adaptation setting, Li *et al.* [35] use source-domain statistics during training while replacing them with target-domain statistics during inference. More recently, Schneider *et al.* [52] combine the source and target statistics during inference, but the statistics are weighted depending on the number of samples that these statistics aggregate. Nado *et al.* [41] propose using batch statistics during inference from the target domain instead of the training statistics acquired from the source domain. Concurrent work [74] proposes a similar technique to ours, α-BN, to calibrate the batch statistics by mixing source and target information. In contrast to our work, the method was proposed without theoretical justification and proper methodology for selecting α. Following the best practice, we rely on the validation set to choose all hyperparameters, hence do not leak knowledge about the test domain. Furthermore, our self-adaptation encompasses not only the normalization layers, but all network layers. Our thorough empirical study also generalizes these results by demonstrating a consistent improvement of out-of-domain accuracy in semantic segmentation.

Adapting the model to a test sample. Several examples from previous work update the model parameters at the time of inference. In object tracking, the object detector has the need to adjust to the changing appearance model of the tracked instance [29]. Conditional generative models can learn from a single image sample for the task of super-resolution [21] and scene synthesis [53]. More recently, this principle has been used for improving the robustness of image classification models [55], [65]. The design of the self-supervised task to perform on the test sample is crucial, and the techniques developed for image classification are unsuitable for dense prediction tasks, such as semantic segmentation considered here. Nevertheless, more suitable alternatives for the self-supervised loss have been recently proposed for domain adaptation [1], and a number of other works devised domain-specific tasks for medical imaging [62] or first-person vision [8].

Setup comparison. Most of these technically related works [52], [55], [65] focus on the problem of domain adaptation in the context of image classification. They typically assume access to a number of samples (or even all test images) from the target distribution at training time. Our work instead addresses semantic segmentation in the domain generalization setting, which is fundamentally different as it only necessitates *a single datum* from the test set. In this scenario, simple objectives, such as entropy minimization employed by Tent [65], improve the baseline accuracy only moderately. By contrast, our self-adaptation with pseudo labels accounts for the inherent uncertainty in the predictions, which proves substantially more effective, as the comparison to Tent in Table 5 reveals. Our task is also different from few-shot learning (*e.g.*, [18]), where the model may adapt at test time using a small *annotated* set of image samples. No such annotation is available in our setup; our model adjusts to the test sample in a completely unsupervised fashion, requires neither proxy tasks to update the parameters [55] nor any prior knowledge of the test distribution.

3 Self-adaptation to a single sample

The traditional assumption at inference time is that the parameters of the segmentation model remain fixed. However, (self-)adaptive systems and their biological counterparts provide an example, where they learn to specialize on the particularities of their environment [57], [69]. By analogy, we here allow the segmentation model to update its parameters. Note that our setup is distinct from the domain adaptation scenario (*e.g.*, in contrast to [65]), since we discard the updated parameters when processing the next sample.

Our approach, visualized in Fig. 1, uses data augmentation as a method to create mini-batches of images for each test sample. Based on the original test image, we first create a set of N augmented images by multi-scaling, horizontal flipping, and grayscaling. These augmented images are used to form a mini-batch, which is fed through the CNN. We transform the produced softmax probabilities from the model back to the original pixels using the inverse affine transformations, and denote the result as $m_{i,:,:,:}$ for every sample i in the mini-batch. This allows the model to have multiple predictions

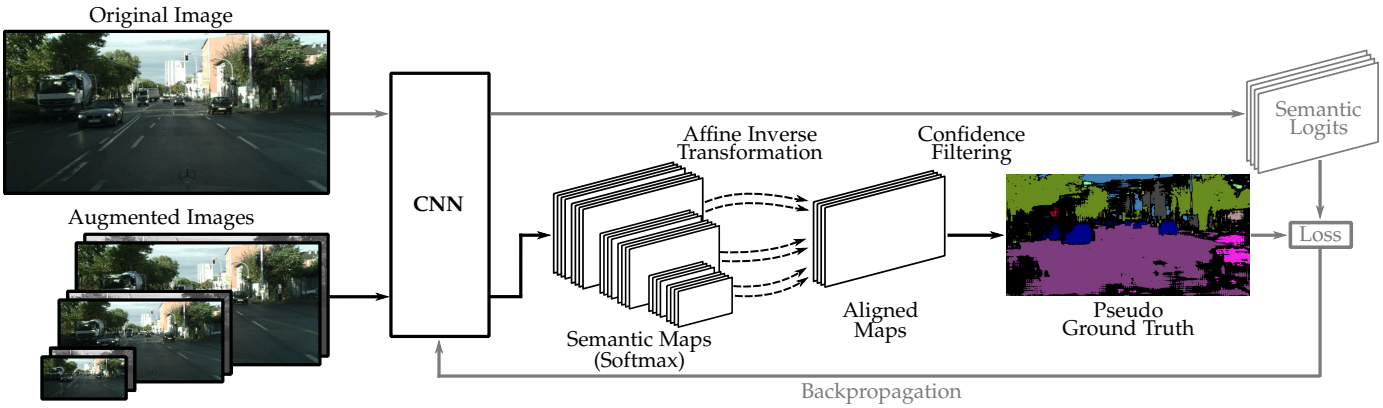

Fig. 1. *Overview of the one-sample adaptation process.* Based on a single test sample, we create a batch of images by augmenting the original input with multi-scale versions. We further add a horizontally flipped and grayscaled version for every scale. In order to transform the resulting output from every version back to the original image, we apply the respective inverse affine transformation to every prediction. Afterwards, we average the predicted softmax probabilities and create a pseudo label using a class-dependent confidence threshold. We update the model parameters by minimizing the cross-entropy loss *w. r. t.* the pseudo label and repeat this process for a small number of iterations, N_t, before producing the final prediction. The updated model is then discarded.

for one pixel. We then compute the mean $\bar{m}$ of these softmax probabilities along the mini-batch dimension i for class c and pixel (j, k) on the spatial grid, as

$$\bar{m}_{c,j,k} = \frac{1}{N} \sum_i m_{i,c,j,k}. \tag{1}$$

Using hyperparameter $\psi \in (0, 1)$, we compute a threshold value t_c from the maximum probability of every class to yield a class-dependent threshold t_c:

$$t_c = \psi \cdot \max(\bar{m}_{c,:,:}). \tag{2}$$

We finally extract the class $c^*_{j,k}$ with the highest probability for every pixel by

$$c^*_{j,k} = \arg\max(\bar{m}_{:,j,k}). \tag{3}$$

We ignore low-confidence predictions using our class-dependent threshold t_c. Specifically, all pixels with a softmax probability below the threshold are set to an ignore label, while the remaining pixels use the dominant class $c^*_{j,k}$ as the pseudo label $u_{j,k}$,

$$u_{j,k} = \begin{cases} c^*_{j,k}, & \text{if } \max(\bar{m}_{:,j,k}) \geq t_{c^*_{j,k}} \\ \text{ignore}, & \text{else.} \end{cases} \tag{4}$$

The pseudo ground truth u for the test image is used to fine-tune the model for N_t iterations with gradient descent using the cross-entropy loss. We determine all hyperparameters, *i. e.* resolution of the scales, threshold ψ, number of iterations N_t, and learning rate η, based on a validation dataset, which is distinct from the test domain (see Sec. 5 and Sec. 6.3dubbed for further discussion). After the self-adaptation process, we produce a single final prediction using the updated model weights. To process the next test sample, we reset these weights to their initial value, hence the model obtains no knowledge about the complete target data distribution.

One natural consideration in this process is the quality of the confidence calibration in the semantic maps, since the `argmax` operation in Eq. (3) and applying the threshold in Eq. (4) aim to select only the most confident pixel predictions. If the confidence values are miscalibrated (*e. g.*, due to the domain shift), a significant fraction of incorrect pixel labels will end up in the pseudo mask. Our self-adaptive normalization, which we detail next, mitigates this issue.

4 Self-Adaptive Normalization

Batch Normalization (BN) [27] has become an inextricable component of modern CNNs [24]. Although BN was originally designed for improving training convergence, there is now substantial evidence that it plays an important role in model robustness [41], including domain generalization [45]. Let $z \in \mathbb{R}^{B,H,W}$ denote a spatial tensor $H \times W$ for an arbitrary feature channel and batch size B, produced by a convolutional layer. We omit the layer and channel indexing, as the following presentation applies to all layers and feature channels on which BN operates. At training time, BN first computes the mean and the standard deviation across the batch and spatial dimensions, *i. e.*

$$\mu = \tfrac{1}{BHW} \sum_{i,j,k} z_{i,j,k}\,, \qquad \sigma^2 = \tfrac{1}{BHW} \sum_{i,j,k} (z_{i,j,k} - \mu)^2. \tag{5}$$

The normalized features $\bar{z}$ follow from applying these statistics,

$$\bar{z}_{i,j,k} = {}^{(z_{i,j,k}-\mu)}\big/_{\sqrt{\sigma^2+\epsilon}}, \tag{6}$$

where ϵ is a small positive constant for numerical stability. Notably, this process differs from the normalization used at inference time. At training time, every BN layer maintains a running estimate of μ and σ across the training batches, which we denote here as $\hat{\mu}$ and $\hat{\sigma}$. At test time, which normally assumes one input sample at a time, feature values are normalized with these running estimates $\hat{\mu}$ and $\hat{\sigma}$ in Eq. (6), since the batch statistics are not available (the batch size is 1). Henceforth, we refer to this scheme as *train BN* (*t*-BN) following previous nomenclature [41].

BN and generalization. In the context of out-of-distribution generalization, the running statistics $\hat{\mu}$ and $\hat{\sigma}$ derived from the source data and can be substantially different had they been computed using the target images. This discrepancy is generally known as the *covariate shift* problem. Domain adaptation methods, which assume access to the (unlabelled) target distribution, often alleviate this issue with a technique known as Adaptive Batch Normalization (AdaBN) [35]. The

key idea behind the method is to simply replace the source running statistics with those computed from the complete target set. However, follow-up work [41], [52] showed that the statistics computed even from a small batch of test images can already tangibly boost the accuracy of image classification models. Dubbed *prediction-time BN* (*p*-BN), the method allows the BN layers to re-use the training-time normalization strategy at test time — by normalizing the features with the statistics μ and σ of the current *test* batch, now comprising multiple images.

In contrast to AdaBN and *p*-BN, which utilize either the whole target distribution or multiple samples, our study of model generalization prescribes access to only a single target example (*i. e.* the one that our model receives as the input at inference time). One natural extension of AdaBN and *p*-BN to this setting is to simply compute the statistics per sample, which amounts to replacing BN layers with Instance Normalization (IN) layers [60] after model training. However, this may cause another extreme scenario for covariate shift. Firstly, the sample statistics may serve only as an approximation to the complete test distribution. Secondly, such a replacement may significantly interfere with the statistics of the activations in the intermediate layers with which the network was trained. It is therefore unsurprising that naively replacing BN with IN layers was found to only hurt the discriminative power of the model in practice [45].

Here, we take a more principled approach and propose a theoretically founded model of domain shift that is simple to implement and works surprisingly well in practice.

4.1 Modeling the domain shift

The domain shift scenario posits that the distribution of the training and test data is different. We model this change with a *continuous non-stationary* stochastic process X_α. In our application to CNNs, this variable corresponds to feature values evolving over "time" $\alpha \in [0, 1]$, reflecting the change from the source ($\alpha = 0$) to a *reference* ($\alpha = 1$) image domain. For example, this definition implies that X_0 can be seen as a normally distributed random variable representing activations in a given feature channel produced by a sample from the *source* data (similarly, X_1 for the reference domain). Let us define the corresponding cumulative distribution function (c.d.f.) $F_{X,\alpha}(X_\alpha)$ and assume $F_{X,0}(X_0)$ and $F_{X,1}(X_1)$ to be known. In the same fashion, we denote the corresponding probability density functions (p.d.f.s) as f_0, f_α, and f_1. Note that $F_{X,0}(X_0) \neq F_{X,1}(X_1)$ by the definition of a non-stationary stochastic process. In our context of model generalization, this formally asserts that the source and the reference domains are different. To construct a new c.d.f. and the corresponding p.d.f. for a given $\alpha \in (0, 1)$, we find realizations of X_0 and X_1 for every $u \in (0, 1]$, denoted as $x_{0,u}$ and $x_{1,u}$, such that $F_{X,0}(x_{0,u}) = F_{X,1}(x_{1,u}) = u$. We define the new c.d.f., parameterized by α, to satisfy $F_{X,\alpha}(x_{\alpha,u}) = u$ iff $x_{\alpha,u} = (1 - \alpha)x_{0,u} + \alpha x_{1,u}$. It can be shown [49] that the corresponding p.d.f. f_α can be expressed in terms of f_0 and f_1 as:

$$f_\alpha(x_{\alpha,u}) = \frac{f_0(x_{0,u}) f_1(x_{1,u})}{(1 - \alpha) f_0(x_{0,u}) + \alpha f_1(x_{1,u})}. \tag{7}$$

Note that for $\alpha = 0$ and $\alpha = 1$ we obtain the original p.d.f.s, but in contrast to mixture models, the new p.d.f. preserves the number of modes in the distribution, *i. e.* if f_0 and f_1 are unimodal, f_α remains unimodal for any $\alpha \in (0, 1)$. Moreover, for normal p.d.f.s f_0 and f_1 (which are of special interest in BN layers of CNNs), computing the first and the second moment of X_α takes a simple and elegant form [49]:

$$\begin{aligned} \mathrm{E}(X_\alpha) &= (1 - \alpha)\mathrm{E}(X_0) + \alpha \mathrm{E}(X_1), \\ \mathrm{Var}(X_\alpha) &= (1 - \alpha)\mathrm{Var}(X_0) + \alpha \mathrm{Var}(X_1). \end{aligned} \tag{8}$$

A generalization of this technique, conceptually illustrated in Fig. 2, is known as *displacement interpolation* [6].

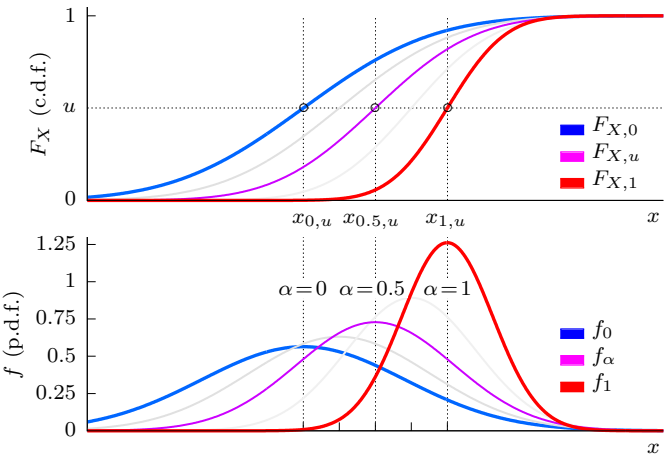

Fig. 2. *Conceptual illustration of displacement interpolation of a 1-D Gaussian.* Using two Gaussian distributions parameterized by α and defined at $\alpha \in \{0, 1\}$, we construct a new Gaussian for an arbitrary $\alpha \in (0, 1)$ (*e.g.* $\alpha = 0.5$ shown here). Given such α, the c.d.f. of the new distribution (the plot above) for any $u \in (0, 1]$ satisfies $F_{X,\alpha}(x_{\alpha,u}) = u$, where $F_{X,0}(x_{0,u}) = F_{X,1}(x_{1,u}) = u$ and $x_{\alpha,u} := (1 - \alpha)x_{0,u} + \alpha x_{1,u}$. The first and the second moments defining the corresponding p.d.f. f_α (the plot below) can be easily obtained using linear interpolation from Eq. (8).

Self-adaptive normalization (SaN). We integrate this domain shift model to improve the test-time behavior of BN layers. Consider a 1-D setting where f_0 and f_1 are the p.d.f.s of the normal distribution*, as modelled by BN layers in CNNs. Let the *source* mean $\hat{\mu}_s$ and the variance $\hat{\sigma}_s^2$ denote the running average of the sample statistics at training time, which also define the density f_0 (*i. e.* at $\alpha = 0$). If we had the *target* domain knowledge expressed by the sufficient statistics $\hat{\mu}_t$ and $\hat{\sigma}_t^2$, we could use those in place of $\hat{\mu}_s$ and $\hat{\sigma}_s^2$ in the BN layers to compensate for the covariate shift. However, at test time we only have access to the sample estimates, μ_t and σ_t^2, provided by a single datum from the target distribution:

$$\mu_t = \tfrac{1}{HW} \sum_{j,k} z_{0,j,k}\,, \qquad \sigma_t^2 = \tfrac{1}{HW} \sum_{j,k} \left(z_{0,j,k} - \mu_t\right)^2, \tag{9}$$

where $z_0 \in \mathbb{R}^{H,W}$ is a spatial feature channel in a CNN of the target sample. The leading index of 0 emphasizes that only one image sample is available in the test batch (*cf.* Eq. (5)).

Our domain shift model assumes that the test sample estimates in Eq. (9), μ_t and σ_t^2, define the *reference* p.d.f. f_1, while f_α constructed in Eq. (7) yields a closer approximation to the true density function of the target domain as a whole for some unknown $\alpha \in (0, 1)$. We demonstrate strong

*Note that the p.d.f. of the normal distribution is strictly positive for any $x_{0,u}, x_{1,u} \in \mathbb{R}$, hence the denominator in Eq. (7) is non-zero and the new p.d.f. is well-defined.

empirical support for this assumption, both in terms of the model accuracy and calibration, in Sec. 6.4.

Using Eq. (8), at inference time we compute the new mean and variance, $\hat{\mu}_t$ and $\hat{\sigma}_t$, as follows:

$$\hat{\mu}_t := (1-\alpha)\hat{\mu}_s + \alpha\mu_t\,, \qquad \hat{\sigma}_t^2 := (1-\alpha)\hat{\sigma}_s^2 + \alpha\sigma_t^2. \quad (10)$$

To normalize features of the test sample, in Eq. (6) we swap μ and σ^2 for the computed $\hat{\mu}_t$ and $\hat{\sigma}_t^2$. Notably, this does not affect the behavior of the BN layers at training time and applies only at test time. Since this approach combines the inductive bias coming in the form of the running statistics from the source domain with statistics extracted from a single test instance, which is an unsupervised process, we refer to it as *Self-adaptive Normalization* (SaN).

Comparison to previous work. Setting $\alpha = 0$ in Eq. (10) defaults to the established procedure, *t*-BN, and uses only the running statistics from training on the source domain at inference time. Conversely, $\alpha = 1$ corresponds to Instance Normalization [60] or, equivalently, to the *p*-BN strategy [41] with a batch size of 1. While μ_t and σ_t are averaged over a *batch* of target images in [41], we rely on a single test sample in this work. Our experiments in Sec. 6.4 expand upon previous analyses [52] with an extensive emperical study of this normalization strategy for semantic segmentation. Batch Instance Normalization [43] used a similar weighting approach for adaptive stylization. However, α was a *training* parameter, whereas we use α only for model selection using a validation set.

5 Designing a principled evaluation

Previous studies [12], [13], [45], [76] on domain generalization for semantic segmentation used divergent evaluation methodologies, which exacerbates the comparison and reproducibility in follow-up research. Encouraged by similar observations [22], we first set out and then follow a number of principles reflecting the best-practice experimentation to enable a fair and reproducible evaluation.

Our first principle is:

A. The test set must comprise multiple domains.

A few previous works [12], [13], [45] used only a single target domain, Cityscapes [15], for testing. However, like other research datasets, Cityscapes [15] is carefully curated (*e. g.*, the same camera hardware and country was used for image capture), hence only partially represents the visual diversity of the world. A more comprehensive approach by [76] considered a number of target domains. However, a separate model was selected for every target domain (based on a *different* validation set). As a result, each selected model may be biased toward the chosen target domain, hence may not be indicative of strong generalization. This leads us to the next principle:

B. A single model must be used for all test domains.

Further, many works do not clearly define the strategy of model selection, such as the used validation set. In fact, recent work [25] even uses the test domains for hyperparameter tuning, for the lack of an established protocol. This is especially undesirable when studying model generalization. Therefore, we stress that:

C. Model selection must not use images from the test set;

TABLE 1. *Comparison to state-of-the-art domain generalization methods for reported source domains and target domains.*

Method	*Source Domains*		*Target Domains*			
	GTA	SYNTHIA	CS	Mapillary	BDD	IDD
DRPC [76]	✓	✓	✓	✓	✓	-
ASG [13]	✓	-	✓	-	-	-
CSG [12]	✓	-	✓	-	-	-
RobustNet [14]	✓	-	✓	✓	✓	-
FSDR [25]	✓	✓	✓	✓	✓	-
WildNet [33]	✓	-	✓	✓	✓	-
SAN-SAW [47]	✓	✓	✓	✓	✓	-
PIN [30]	✓	✓	✓	✓	✓	-
This work	✓	✓	✓	✓	✓	✓

D. The validation set must be clearly specified.

These principles follow naturally from the requirements of domain generalization, with principles *C* and *D*, in particular, being widely accepted in machine learning research. To our surprise, we found that *no previous work on domain generalization for semantic segmentation has yet fulfilled all of these principles.* To encourage and facilitate good evaluation practice, we therefore revise the experimental protocol used so far. We consider a practical scenario in which a supplier prepares a model for a consumer without *a-priori* knowledge on where this model may be deployed, much akin to [32]. On the supplier's side, we assume access to two data distributions for model training and validation, the *source data* and the *validation set*. After training the model on the source data and choosing its hyperparameters on the validation set, we assess its generalization ability on three qualitatively distinct *target sets*. The average accuracy across these sets provides an estimate of the expected model accuracy for its out-of-distribution deployment on the consumer's side. Next, we concretize the datasets used in this study, which focus on traffic scenes for compatibility with previous work [12], [45], [76]. Notably, the scale of our study, summarised in Table 1, exceeds that of previous work.

Source data. We train our model on the training split of two synthetic datasets (mutually exclusive) with low-cost ground truth annotation: GTA [50] and SYNTHIA [51]. Importantly, these datasets exhibit visual discrepancy (*i. e.* domain shift) *w. r. t.* the real imagery, to which our model needs to generalize.

Validation set. For model selection and hyperparameter tuning, we use the validation set of WildDash [77]. In our scenario, the validation set is understood to be of limited quantity, owing to its more costly annotation compared to the source data. In contrast to the training set, however, it bears closer visual resemblance to the potential target domains.

Multi-target evaluation. Following model selection, we evaluate the single model on three target domains comprising the validation sets from Cityscapes [15], BDD [75], and IDD [61]. The choice of these test domains stems from a number of considerations, such as the geographic origin of the scenes (Cityscapes, BDD, and IDD were collected in Germany, North America, and India, respectively). Geographic distinction as well as substantial differences in data acquisition (*e. g.*, camera properties) of these datasets bring together an assortment of challenges for the segmentation model at test

time. Since the deployment site of our model is unknown, we assume a uniform prior over the target domains as our test distribution. Given comparable sample sizes in each target domain, the average of the mean accuracy across them would approximate the expected model accuracy.

To compare to previous works, we also evaluate on Mapillary [44]. Mapillary does not publicly disclose the geographic origins of individual samples, hence is unsuitable to identify a potential location bias acquired by the model from the training data. This is possible in our proposed evaluation protocol, since the geographic locations from Cityscapes, BDD, and IDD do not overlap.

6 Experiments

6.1 Overview

Let us begin by highlighting the main results from the detailed analysis that will follow:

SaN. First, we empirically verify that SaN yields a consistent boost of the segmentation accuracy in the out-of-distribution scenario, also in comparison to alternative normalization techniques from the literature. Crucial to the efficacy of our self-adaptation strategy from Sec. 3, we also find significant improvements of model calibration in terms of the expected calibration error, ECE [42].

State of the art. Self-adaptation achieves new state-of-the-art accuracy in virtually all benchmark scenarios. This is despite following our more stringent evaluation principles *A–D* from Sec. 5, *which actually put self-adaptation at a disadvantage* when comparing to previous work. In particular, we evaluate a single model on multiple domains and do not access the test images for model selection. Moreover, we use a weaker backbone architecture (DeepLabv1) than the previous state of the art. In fact, the accuracy of self-adaptation improves further with more advanced architectures (*cf.* Sec. 6.7).

Runtime. Through a detailed analysis, we find that self-adaptation, while not real-time yet, is nevertheless more cost-effective than the widely adopted model ensembles. We identify and study adjustable factors, such as the number of self-adaptation iterations, that provide flexible leverage over the accuracy-runtime trade-off, lacking in previous works.

6.2 Dataset details

GTA [50] is a street view dataset generated semi-automatically from the computer game Grand Theft Auto V. The dataset consists of 12,403 training images, 6,382 validation images, and 6,181 testing images of resolution 1914 × 1052 with 19 different semantic classes.

SYNTHIA. We use the SYNTHIA-RAND-CITYSCAPES subset of the synthetic dataset SYNTHIA [51], which contains 9,400 images, and has 16 semantic classes in common with GTA. Images have a resolution of 1280 × 760 pixels.

WildDash. The WildDash benchmark [77] was developed to evaluate models *w. r. t.* their robustness for driving scenarios under real-world conditions. It comprises 4256 images of real-world scenes with a resolution of 1920 × 1080 pixels.

Cityscapes [15] is an ego-centric street-scene dataset and contains 5,000 high-resolution images with 2048 × 1024 pixels. It is split into 2,975 train, 500 val, and 1,525 test images with 19 semantic classes being annotated.

BDD [75] is a driving video dataset, which also contains semantic labelings with the identical 19 classes as in the other datasets. Images have a resolution of 1280 × 720 pixels. The training, validation, and test sets contain 7,000, 1,000, and 2,000 images, respectively.

IDD [61] is a dataset for road scene understanding in unstructured environments. It contains 10,003 images annotated with 34 classes. We only evaluate on the 19 classes overlapping with the other datasets. IDD is split into 6,993 training images, 981 validation images, and 2,029 test images.

Mapillary [44] contains 66 object classes; analogously to IDD we only evaluate on the 19 classes overlapping with the other datasets. The dataset is split into a training set with 18,000 images and a validation set with 2,000 images with a minimum resolution of 1920 × 1080 pixels.

6.3 Caveats of baseline implementation

To legitimately study the out-of-distribution model accuracy, it is essential to establish its upper bound attainable by the standard training procedures [40]. *Indeed, we empirically found poorly tuned baselines to be the easiest to improve upon, both in terms of the absolute and relative accuracy margin.* We followed the best practice in the literature and found a number of training details to be crucial for obtaining a highly competitive baseline, *i. e.* a model that does not use our self-adaptive inference. Among them is using heavy data augmentation. Concretely, we use random horizontal flipping, multi-scale cropping with a scale range of $[0.08, 1.0]$, as well as photometric image perturbations:[†] color jitter, random blur, and grayscaling. Color jitter, applied with probability 0.5, perturbs image brightness, contrast, and saturation using a factor sampled uniformly from the range $[0.7, 1.3]$. We use a different range of $[0.9, 1.1]$ for the hue factor. We randomly blur the image using a Gaussian kernel with the standard deviation sampled from $[0.1, 2.0]$. Additionally, we convert the image to grayscale with a probability of 0.1. Furthermore, we also found that the polynomial decay schedule we used for the learning rate, as well training for at least 50 epochs (for both GTA and SYNTHIA) are essential to achieve a high baseline accuracy. Note that we only used WildDash as the validation set to tune these training details. We also experimented with higher input resolution and a larger batch size, but did not observe a significant improvement, yet a drastic increase in the computational overhead.

We implement our framework in PyTorch [46]. Following [45], our baseline model is DeepLabv1 [9] without CRF post-processing, but the reported results also generalize to more advanced architectures, including Transformers (see Sec. 6.7). We mainly use ResNet-50 and ResNet-101 [24] pre-trained on ImageNet [17] as backbone. We minimize the cross-entropy loss with an SGD optimizer and a learning rate of 0.005, decayed polynomially with the power set to 0.9. All models are trained on the source domains for 50 epochs with batch size, momentum, and weight decay set to 4, 0.9, and 0.0001, respectively. For data augmentation, we compute crops of random size (0.08 to 1.0) of the original image size, apply a random aspect ratio (3/4 to 4/3) to the crop, and then resize the result to 512 × 512 pixels. Furthermore, we use

[†]We use Pillow library (https://pillow.readthedocs.io) to implement photometric augmentation.

TABLE 2. *Segmentation accuracy and model calibration using SaN. (a)* The mean IoU (%) on three target domains (Cityscapes, BDD, IDD) across both backbones, trained on GTA and SYNTHIA. *t*-BN denotes train BN [27], while *p*-BN refers to prediction-time BN [41]. On average, SaN significantly outperforms both baselines. *(b)* The *ECE* (%) on three target domains (Cityscapes, BDD, IDD) across both backbones trained on GTA and SYNTHIA. SaN improves model calibration of the baseline and even outperforms MC-Dropout [19]. Remarkably, both techniques prove to be mutually complementary. The lower bounds (in-domain) on ECE when directly trained on Cityscapes/BDD/IDD are 7.46%/14.99%/9.48% for ResNet-50, and 6.52%/14.57%/9.00% for ResNet-101.

(a)

Method	*IoU (%, ↑), source: GTA / SYNTHIA*					
	CS		BDD		IDD	
ResNet-50						
w/ *t*-BN	30.95	31.83	28.52	24.30	32.78	24.73
w/ *p*-BN	**37.71**	33.83	31.67	23.36	30.85	23.39
w/ SaN (*Ours*)	37.54	**36.14**	**32.79**	**26.66**	**34.21**	**26.37**
ResNet-101						
w/ *t*-BN	32.90	37.25	32.54	**29.32**	30.36	27.19
w/ *p*-BN	39.88	34.58	34.30	24.24	33.05	22.32
w/ SaN (*Ours*)	**42.17**	**38.01**	**35.40**	28.66	**33.52**	**27.28**

(b)

Method	*ECE (%, ↓), source: GTA / SYNTHIA*					
	CS		BDD		IDD	
ResNet-50	37.28	37.50	35.61	43.19	27.73	40.11
w/ SaN (*Ours*)	30.57	30.96	30.94	33.27	26.90	36.31
w/ MC-Dropout	30.29	34.82	29.80	37.30	24.17	36.63
w/ both (*Ours*)	**25.50**	**30.66**	**27.36**	**33.06**	**22.62**	**35.60**
ResNet-101	35.24	31.39	33.74	33.77	27.28	36.56
w/ SaN (*Ours*)	26.12	30.33	28.89	31.83	23.98	36.26
w/ MC-Dropout	31.30	32.73	29.95	32.76	25.15	34.07
w/ both (*Ours*)	**24.44**	**27.71**	**28.68**	**30.48**	**23.32**	**32.67**

random horizontal flipping, color jitter, random blur, and grayscaling. We train our models with SyncBN [46] on two NVIDIA GeForce RTX 2080 GPUs.

For self-adaptation, we use multiple scales with factors of (0.25, 0.5, 0.75) *w. r. t.* the original image resolution, as well as horizontal flipping and grayscaling for each scale. We study the relative importance of these augmentation types in Sec. 6.7. Based on the validation set WildDash, we set threshold $\psi = 0.7$, use $N_t = 10$ iterations and a learning rate $\eta = 0.05$. We only train the layers conv4_x, conv5_x, and the classification head as we did not observe any benefits from updating all model parameters.[‡]. Furthermore, this reduces runtime due to not backpropagating through the whole network. We investigate this choice as part of the runtime-accuracy analysis in Sec. 6.6.

On importance of the baseline. We note that the reported accuracy from previous work, as we will see in Table 3, is not entirely consistent *w. r. t.* the choice of the model architecture. In particular, DRPC [76] uses an FCN, yet outperforms other domain generalization approaches with DeepLabv1 [45] and DeepLabv2 [12], [13] considerably. This is a regrettable consequence of inconsistent training schedules used in previous works that proved difficult to reproduce. For example, at the time of submission, the implementation by Yue *et al.* [76], which reports excellent segmentation accuracy for the baseline, has not been made available;[§] parts of the code implementing the semantic segmentation architecture introduced in [45] are also not publicly available.[¶] These circumstances make reporting the accuracy of the implementation-specific baselines indispensable, which has thus become the standard practice in more recent previous [12], [13] and related works [22]. To facilitate reproducibility of our results, we publicly release our implementation under Apache License 2.0.

[‡]Due to the higher computational cost of the ResNet-101 backbone, we only adapt the layers `conv5_x` and the classification head.

[§]https://github.com/xyyue/DRPC/issues

[¶]https://github.com/XingangPan/IBN-Net

6.4 Improving accuracy and calibration with SaN

For both source domains (GTA, SYNTHIA) in combination with all main target domains (Cityscapes, BDD, IDD), we investigate the effect of SaN on out-of-domain segmentation accuracy and calibration. We first select α on the validation set (see Sec. 6.7 for the methodology) and report the results in Table 2. In Table 2a, we compare the SaN accuracy on the target domains with *t*-BN and *p*-BN. Remarkably, SaN improves the mean IoU not only of the *t*-BN baseline (*e. g.*, by 4.1% IoU with ResNet-50 trained on GTA, on average), which represents an established evaluation mode, but also over the more recent *p*-BN [41]. When trained on SYNTHIA, SaN provides benefits for the expected segmentation accuracy even if *p*-BN fails to improve over the *t*-BN baseline, as is the case with ResNet-50; the results remain on par with the *t*-BN baseline even when *p*-BN is significantly worse than *t*-BN. Furthermore, we found that the calibration of our models, in terms of the expected calibration error (ECE) [42], also improves. As shown in Table 2b, not only does SaN substantially enhance the baseline, but it even tends to outperform the commonly used MC-Dropout method [19]. Rather surprisingly, SaN exhibits a complementary effect with MC-Dropout: the calibration of the predictions improves even further when both methods are used jointly. This observation holds even for the segmentation accuracy. For example, our model trained on GTA and tested with SaN *and* MC-Dropout (*i. e.*, by averaging the predictions) improves the IoU of the SaN-only inference on Cityscapes, BDD, IDD by 1.3%, 2.34%, 1.29%, whereas MC-Dropout alone does not have such an effect.

Overall, the combined results from Table 2 demonstrate that SaN improves both the model accuracy and the calibration quality of the predictions in the out-of-distribution setting irrespective of the backbone network and specifics of the source data.

Comparison to other normalization strategies. We additionally compare SaN to alternative techniques proposed in the literature: Batch-Instance Normalization (BIN) [43] and Switchable Normalization (SN) [39]. Although both of these techniques share technical similarities with our SaN,

TABLE 3. *Mean IoU (%) comparison to state-of-the-art domain generalization methods* for both source domains (GTA, SYNTHIA) as well as three target domains (Cityscapes, Mapillary, BDD). In-domain training to obtain the upper bounds uses our baseline DeepLabv1 following the same schedule as with the synthetic datasets. (‡), (†) and (††) denote the use of FCN [37], DeepLabv2 [10], and DeepLabv3+ [11], respectively.

	Method	*Backbone: ResNet-50*						*Backbone: ResNet-101*					
		CS		Mapillary		BDD		CS		Mapillary		BDD	
	In-domain Bound	71.23		58.39		58.53		73.84		62.81		61.19	
GTA	No Adapt	32.45	*↑4.97*	25.66	*↑8.46*	26.73	*↑5.41*	33.56	*↑8.97*	28.33	*↑9.72*	27.76	*↑10.96*
	DRPC‡ [76]	37.42		34.12		32.14		42.53		38.05		38.72	
	No Adapt	25.88	*↑3.77*		–		–	29.63	*↑3.16*		–		–
	ASG† [13]	29.65						32.79					
	No Adapt	25.88	*↑9.39*		–		–	29.63	*↑9.25*		–		–
	CSG† [12]	35.27						38.88					
	No Adapt	28.95	*↑7.63*	28.18	*↑12.15*	25.14	*↑10.06*		–		–		–
	RobustNet†† [14]	36.58		40.33		35.20							
	No Adapt		–		–		–	33.4	*↑11.4*	27.9	*↑15.5*	27.3	*↑13.9*
	FSDR‡ [25]							44.8		43.4		41.20	
	No Adapt	35.16	*↑9.46*	31.29	*↑14.77*	29.71	*↑8.71*	35.73	*↑10.06*	33.42	*↑13.66*	34.06	*↑7.67*
	WildNet†† [33]	44.62		46.09		38.42		45.79		47.08		**41.73**	
	No Adapt	29.32	*↑10.43*	28.33	*↑13.53*	25.71	*↑11.63*	30.64	*↑14.69*	28.65	*↑12.12*	27.82	*↑13.36*
	SAN-SAW [47]	39.75		41.86		37.34		45.33		40.77		41.18	
	No Adapt	31.60	*↑9.40*	29.00	*↑8.40*	25.10	*↑9.50*	–		–		–	
	PIN [30]	41.00††		37.40††		34.60††		44.90†		39.71†		41.31†	
	No Adapt	30.95	*↑14.18*	34.56	*↑12.93*	28.52	*↑11.09*	32.90	*↑14.09*	36.00	*↑11.49*	32.54	*↑7.67*
	Self-adaptation *(Ours)*	**45.13**		**47.49**		**39.61**		**46.99**		**47.49**		40.21	
SYNTHIA	No Adapt	28.36	*↑7.29*	27.24	*↑5.50*	25.16	*↑6.37*	29.67	*↑7.91*	28.73	*↑5.39*	25.64	*↑8.70*
	DRPC‡ [76]	35.65		32.74		31.53		37.58		34.12		34.34	
	No Adapt		–		–		–	-		-		-	
	FSDR‡ [25]							40.8		39.6		**37.4**	
	No Adapt	23.18	*↑15.74*	21.79	*↑12.73*	24.50	*↑10.74*	23.85	*↑17.02*	21.84	*↑15.42*	25.01	*↑10.97*
	SAN-SAW [47]	38.92		34.52		**35.24**		40.87		37.26		35.98	
	No Adapt	31.83	*↑9.77*	33.41	*↑7.80*	24.30	*↑9.05*	37.25	*↑5.07*	36.84	*↑4.36*	29.32	*↑3.95*
	Self-adaptation *(Ours)*	**41.60**		**41.21**		33.35		**42.32**		**41.20**		33.27	

TABLE 4. *Mean IoU (%) comparison of SaN to SN [39] and BIN [43].*

Method	CS	BDD	IDD	Mean
SN	31.75	**33.60**	31.60	32.32
BIN	34.57	32.68	30.22	32.49
SaN *(Ours)*	**37.54**	32.79	**34.21**	**34.85**

these approaches were developed for different purposes. SN was only shown to improve in-domain accuracy, while BIN tackles domain adaptation for image classification, not domain generalization studied in this work. Furthermore, both methods modify the model architecture before training, while SaN works with any pretrained semantic segmentation model. Nevertheless, we implemented both SN and BIN in our segmentation model based on the ResNet-50 backbone. We trained these approaches on GTA in an identical setup as SaN. From results in Table 4, we find that SaN outperforms both BIN and SN by a significant margin in terms of mean IoU of the target domains.

6.5 Evaluating self-adaptation: New state of the art

We compare self-adaptation with state-of-the-art domain generalization methods in Table 3. While most of the other methods report their results on weakly tuned baselines, we show consistent improvements even over a carefully tuned baseline, regardless of backbone architecture or source data. Our single model with self-adaptation even outperforms DRPC [76] and FSDR [25] on most benchmarks (*e.g.*, by $4.2 - 9.4\%$ on Mapillary with ResNet-101). These methods train individual models for each target domain; FSDR [25] even uses the target domains for hyperparameter tuning, hence violates our out-of-distribution evaluation protocol. Recall that ASG [13] and CSG [12] (as well as DRPC [76]) require access to a distribution of real images for training, while IBN-Net [45] modifies the model architecture. Our approach requires neither, alters only the inference procedure, yet outperforms these methods in all benchmark scenarios substantially. WildNet [33] appears to be more accurate than self-adaptation on BDD with ResNet-101. However, this is not a fair comparison, since it uses a more advanced architecture (DeepLabv3+ *vs.* DeepLabv1). As we will see in Sec. 6.7, self-adaptation, in fact, outperforms WildNet when using the same architecture by 2.79% IoU. Similarly, SAN-SAW [47] reaches higher accuracy on BDD, if trained on SYNTHIA, presumably due to the ASPP module [10] that we do not use. Self-adaptation considerably outperforms SAN-SAW in all other scenarios. Overall, despite adhering to a

TABLE 5. *Mean IoU (%) comparison to Tent [65]. (a)* Using the same HRNet-W18 backbone [67] as in [65], our self-adaptation outperforms Tent [65] substantially, even when using SaN alone. *(b)* Our self-adaptation loss defined in Sec. 3 yields significantly higher segmentation accuracy compared to the entropy minimization used by Tent [65].

(a) HRNet-W18 backbone

Method (*Source: GTA*)	CS
Tent [65]	36.4
SaN *(Ours)*	40.0
Self-adaptation *(Ours)*	**44.1**

(b) Comparison to entropy minimization (DeepLabv1)

Method (*Source: GTA*)	CS	BDD	IDD	Mean
Entropy min. [65]	40.20	35.40	33.97	36.52
Self-adaptation *(Ours)* (ψ=0.0)	44.00	38.55	39.09	40.55
Self-adaptation *(Ours)* (ψ=0.7)	**45.13**	**39.61**	**40.32**	**41.69**

TABLE 6. *Mean IoU (%) with TTA [54] and our self-adaptation* reported across both source domains (GTA, SYNTHIA) and three target domains (Cityscapes, BDD, IDD).

Method (w/ SaN)	*Source: GTA / SYNTHIA*					
	CS		BDD		IDD	
ResNet-50	37.54	36.14	32.79	26.66	34.21	26.37
TTA	42.56	39.67	37.72	32.10	37.98	30.46
Self-adapt. *(Ours)*	**45.13**	**41.60**	**39.61**	**33.35**	**40.32**	**31.22**
ResNet-101	42.17	38.01	35.40	28.66	33.52	27.28
TTA	44.37	39.91	38.49	32.68	38.35	30.04
Self-adapt. *(Ours)*	**46.99**	**42.32**	**40.21**	**33.27**	**40.56**	**31.40**

TABLE 7. *Runtime (ms) with TTA [54] and self-adaptation.* We report the runtime for both ResNet-50 and ResNet-101 on three resolutions of 2048 × 1024, 1280 × 720, and 1920 × 1080, common to the target domains Cityscapes, BDD, and IDD, respectively. All methods use SaN with a negligible computational cost.

Method (w/ SaN)	*Mean runtime per sample (ms)*			
	CS	BDD	IDD	Mean
ResNet-50	**314**	**136**	**214**	**221**
TTA	1308	713	766	929
Self-adaptation (*Ours*)	7862	3742	5307	5637
ResNet-101	**458**	**239**	**252**	**316**
TTA	1519	766	860	1048
Self-adaptation (*Ours*)	9060	4241	6142	6481

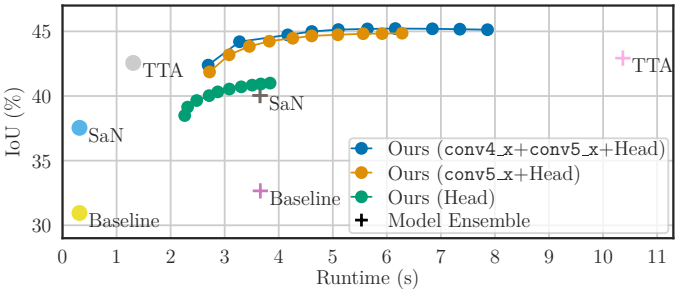

Fig. 3. *Runtime-accuracy comparison on GTA → Cityscapes generalization using one NVIDIA GeForce RTX 2080 GPU.* The curves trace self-adaptation iterations, *i. e.* the first point corresponds to $N_t = 1$, while the last shows $N_t = 10$. Self-adaptation proves more cost-effective than model ensembles of 10 networks and can trade off the accuracy against the inference time by varying the number of iterations and the choice of the adapted layers. The same conclusion holds for BDD and IDD.

stricter evaluation practice and a simpler model architecture, self-adaptation overwhelmingly exceeds the segmentation accuracy of previous work.

Comparison to Tent [65]. Like self-adaptation, Tent [65] also updates model parameters at test time. However, different from constructing the pseudo labels based on well-calibrated predictions in our self-adaptation, Tent simply minimizes the entropy of a single-scale prediction. Tent also limits the adaptation to updating only the BN parameters, whereas our self-adaptation generalizes this process to convolutional layers. To demonstrate these advantages, we compare to Tent [65] in Table 5a. We train HRNet-W18 [67] on GTA and compare the IoU on Cityscapes to the equivalent configuration of Tent. *Under a comparable computational budget of 10 model update iterations*, self-adaptation substantially outperforms Tent, by a remarkable 7.7% IoU. SaN alone already outperforms Tent significantly with a single forward pass by 3.6%. Further, our self-adaptation loss is also considerably more effective than the entropy minimization employed by Tent, as shown in Table 5b. We improve by 4% IoU on average even without tuning threshold ψ, and by 5.2% when it is tuned.

Comparison to test-time augmentation. We also compare our self-adaptation to the standard non-adaptive inference, as well as test our models against Test-Time Augmentation (TTA) [54] as a stronger baseline. TTA augments the test samples with their flipped and grayscaled version on multiple scales and averages the predictions as the final result. This augmentation strategy yielded highest accuracy benefits on our validation set. In Table 6, we show IoU scores for both source domains (GTA, SYNTHIA) and three target domains (Cityscapes, BDD, IDD) across both backbones. Even though TTA improves the baseline (*e. g.*, by 3.37% IoU with ResNet-101 using GTA, on average), our proposed self-adaptation surpasses that by a clear and consistent margin of 2.19% IoU on average. This observation aligns well with our reported ECE scores in Table 2b and our comparison to Tent [65] above: it demonstrates that self-adaptation exploits the calibrated confidence of our predictions to yield reliable pseudo labels for adapting the model for a particular test sample.

6.6 Runtime analysis: Self-adaptation is cost-effective

We profile the runtime of the baseline, test-time augmentation (TTA) and self-adaptation across a range of input resolutions available in the target domains using a single NVIDIA GeForce RTX 2080 GPU. As a widely adopted baseline, we also consider a model ensemble comprising 10 networks with a DeepLabv1 architecture (as in self-adaptation), each one trained with a unique random seed [23]. Note that TTA, self-adaptation, and the ensemble use SaN for a fair comparison, and we also test the ensemble with TTA.

Table 7 reports the runtime estimates per sample, averaged over the complete image set (to account for small deviations in the input resolution). Since our self-adaptation uses 10 update iterations, which requires backpropagation, the linear increase in the inference time *w. r. t.* TTA and the baseline is expected. Such runtime cost may at first

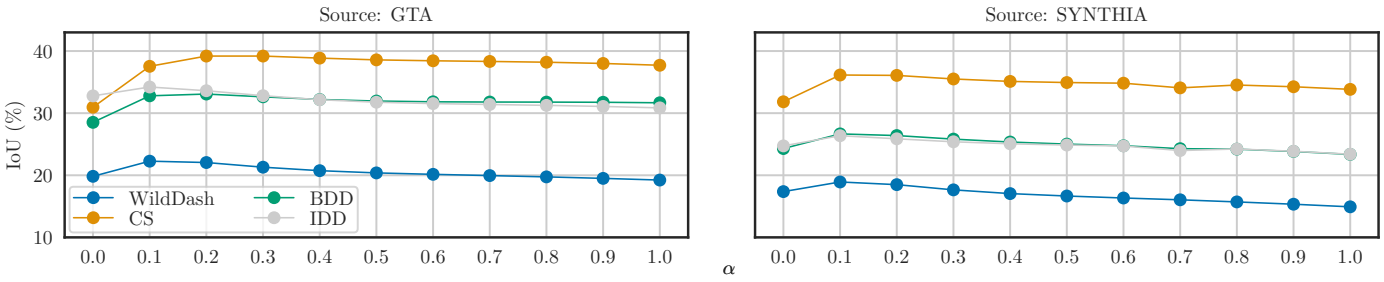

Fig. 4. *Mean IoU (%, ↑) on the target and validation sets by varying α in SaN.* Using ResNet-50 backbone, we report the accuracy on the target domains (Cityscapes, BDD, IDD) for different values of α in steps of 0.1 after training on GTA *(left)* and SYNTHIA *(right)*. We choose one α using the validation set (WildDash), which remains constant during the inference on all target domains.

seem prohibitive. However, the runtime *vs*. accuracy plot in Fig. 3 provides a strong justification: it clearly shows that self-adaptation is more efficient and more accurate than model ensembles. Self-adaption further offers an accuracy trade-off by means of varying the number of iterations required to adapt to a single sample, or updating fewer network layers. For example, we may skip `conv4_x` from self-adaptation to significantly reduce the computational footprint at an almost negligible detriment to the accuracy. Alternatively, self-adaption can amortize the runtime costs by using fewer update iterations, while still providing a clear accuracy improvement. For example, using $N_t = 3$ iterations decreases the runtime of self-adaptation almost twofold while preserving around 95% of the model accuracy attainable with more iterations. Further, we investigated the influence of using the automatic mixed precision module in PyTorch and its influence on the inference runtime. While maintaining an identical IoU on Cityscapes using the ResNet-50 backbone, mixed precision with $N_t = 10$ reduced the runtime by almost 30%: the inference takes only 5746 ms compared to 7862 ms using single precision.

While our implementation of self-adaptation is not real-time yet, it may already serve as a useful reference for less accurate real-time systems by providing periodic feedback. For real-time needs, standalone SaN can significantly boost the baseline accuracy without any computational overhead.

Overall, we find that *(i)* self-adaptation provides clear advantages in segmentation accuracy over baselines at a reasonable increase of the inference time; *(ii)* it is both more accurate and more efficient than model ensembles; and *(iii)* it provides a flexible runtime-accuracy trade-off by means of varying the number of update iterations and the number of the layers to adjust.

6.7 Further empirical analysis

Selecting α. Setting an optimal α for every target domain is infeasible in domain generalization as the target domain during inference is unknown. Using our revised protocol, we choose the optimal α in steps of 0.1 based on the IoU on the validation set of WildDash instead. Fig. 4 shows a detailed plot of the influence of α on the segmentation accuracy, both on the validation set of WildDash and on the target domains. We observe that the maximum accuracy on the validation set is attained with $\alpha = 0.1$, and that self-adaptation is quite robust to this inevitably suboptimal choice in terms of the test accuracy. Clearly, there cannot be any guarantee that value 0.1 is the optimal one for all target domains. However, choosing α based on the validation set is in line with the established practice in machine learning, as our principle *C* from Sec. 5 also asserts — tuning model hyperparameters is not allowed on the test sets (Cityscapes, BDD, IDD), but is only possible on the validation set (WildDash). Despite $\alpha = 0.1$ being *suboptimal w.r.t.* the test domains, hence disadvantaging self-adaptation in comparison to previous work that used those test domains for hyperparameter tuning and model selection, we nevertheless reach state-of-the-art results (*cf.* Table 3).

TABLE 8. *Mean IoU (%) using our self-adaptation integrated with DeepLabv3+ [11] based on a ResNet-50 and ResNet-101 backbone as well as with HRNet-W18, HRNet-W48 [67] and UPerNet [71] with a Swin-T backbone [36].* We observe substantial improvements of the segmentation accuracy on all three target domains (Cityscapes, BDD, and IDD) after training on GTA.

Method	*Target domains*			
	CS	BDD	IDD	Mean
DeepLabv3+ ResNet-50	37.51	35.45	37.50	36.82
Self-adaptation (*Ours*)	**46.56**	**43.17**	**44.07**	**44.60**
DeepLabv3+ ResNet-101	38.19	37.05	38.22	37.82
Self-adaptation (*Ours*)	**48.14**	**44.52**	**45.72**	**46.13**
HRNet-W18	33.08	29.40	32.97	31.82
Self-adaptation (*Ours*)	**44.05**	**38.29**	**43.78**	**42.04**
HRNet-W48	34.66	30.85	34.64	33.38
Self-adaptation (*Ours*)	**48.82**	**42.79**	**43.74**	**45.12**
UPerNet Swin-T	39.67	36.04	39.74	38.48
Self-adaptation (*Ours*)	**45.04**	**39.77**	**44.33**	**43.05**

Self-adaptation with advanced architectures. Our conclusions also hold for more advanced and recent architectures. We trained five state-of-the-art segmentation models on GTA: DeepLabv3+ [11] with both a ResNet-50 and ResNet-101 backbone, HRNet-W18 and HRNet-W48 [67] as well as UPerNet [71] with a Swin-T Transformer backbone [36]. Table 8 reports consistent and substantial improvements of the mean IoU over the baseline across all these architectures and the target domains.

Choosing test-time augmentation strategies. We verify the influence of the augmentation type used by self-adaptation. Recall from Sec. 6.3 that we use multiple scales with horizontal flipping and grayscaling to augment one image sample. We compare a flipping-only, scaling-only, and grayscaling-only version of our self-adaptation to the combination of flipping, grayscaling, and the spatial scaling, which we used

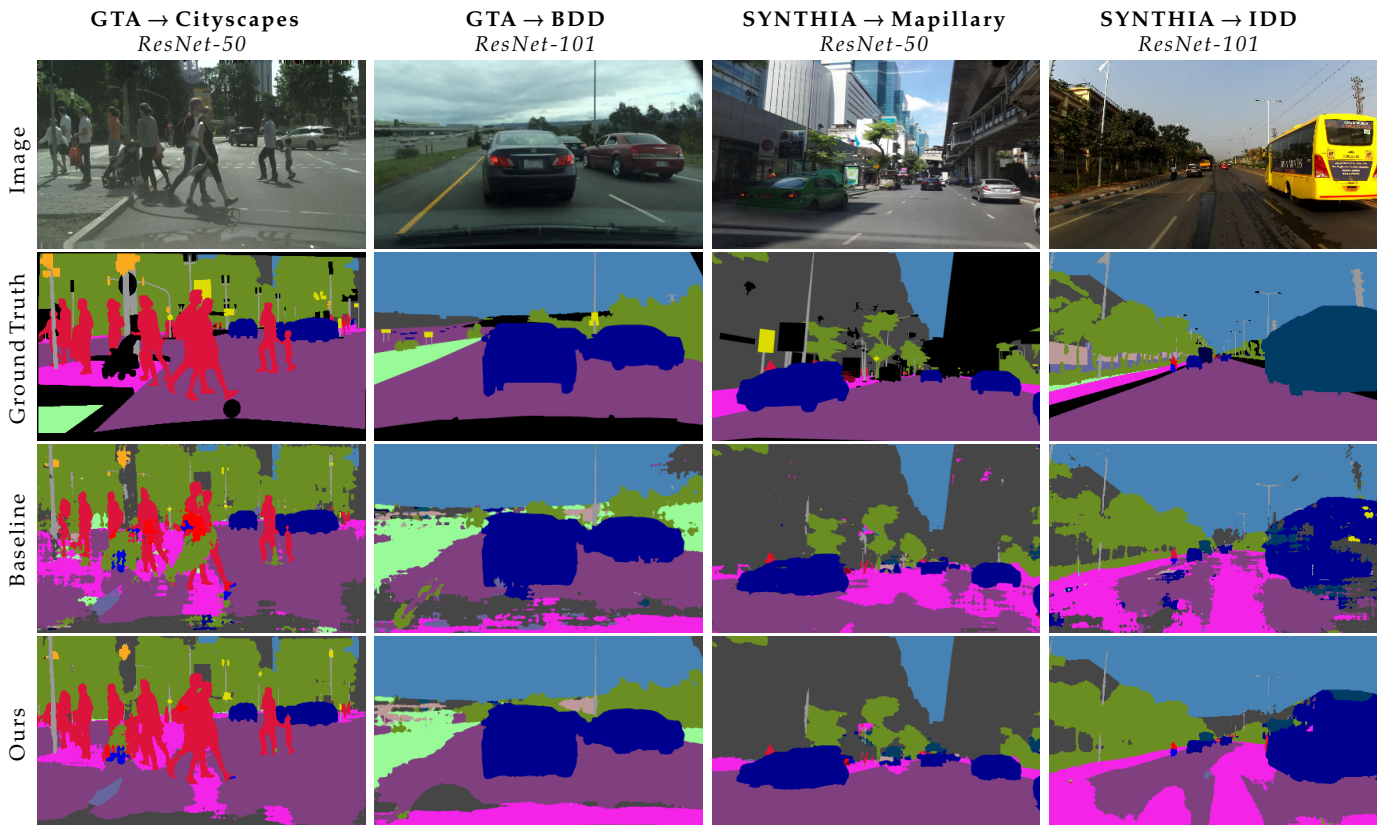

Fig. 5. *Qualitative semantic segmentation results for the generalization from GTA to Cityscapes and GTA to BDD, SYNTHIA to Mapillary, and SYNTHIA to IDD* for the ResNet-50 or ResNet-101 backbone. We show the input image (top row), ground truth and the predictions of the baseline model and of our proposed self-adaptation (bottom row).

TABLE 9. *The role of the augmentation type in self-adaptation.* We report mean IoU (%, ↑) and runtime (ms, ↓) for TTA [54] and our self-adaptation for the GTA source domain and the Cityscapes target domain for the ResNet-50 backbone. We investigate multiple scales (MS), horizontal flipping (HF), and grayscaling (G).

Method	*TTA*		*Ours*	
	IoU	Runtime	IoU	Runtime
Baseline	30.95	**314**	31.47	**7135**
SaN (Ours)	37.54	**314**	39.04	**7135**
MS	42.27	749	44.92	7272
HF	38.01	986	39.33	7593
G	37.96	908	39.65	7193
MS + HF	42.48	1316	44.94	7890
MS + G	42.28	1202	45.06	7486
MS + HF + G	**42.56**	1308	**45.13**	7862

the experiments above. We used a ResNet-50 backbone trained on GTA and report the accuracy on Cityscapes in Table 9. We observe a significant boost in accuracy in comparison to a strong baseline that uses our SaN with no augmentations. Furthermore, we show that using multiple scales is more important than flipping for self-adaptation. Note that the augmentations used do not impact runtime in a significant way, since the batch sizes between these setups vary insignificantly; it is the backpropagation that dominates the main computational footprint. Varying the number of iterations, as studied in Sec. 6.6, provides a more flexible mechanism for accuracy-runtime trade-off.

6.8 Qualitative examples

In Fig. 5 we visualize qualitative segmentation results produced by self-adaptation for generalization across different choices of the source and the target domain, as well as the backbone architecture. We observe a clearly perceivable improvement over the baseline, especially in terms of image boundary consistency. Our approach exhibits more homogeneous semantic masks with visibly fewer irregular-shaped artifacts than the baseline (*e.g.*, "sidewalk" false positives in SYNTHIA → Mapillary). Self-adapted models may still struggle with cases of mislabeling regions with incorrect, but semantically related classes. For example, the model after self-adaptation may assign "sidewalk" to the road pixels, as visible in GTA → BDD and SYNTHIA → IDD examples. This is not surprising if the erroneous labels are already contained in the pseudo labels of the initial prediction (manifested by the baseline model), on which self-adaptation relies. These failure instances seem to occur more frequently if the domain shift between the train and the test distributions is more significant, as this can lead to poorly calibrated predictions. In many cases, this can be alleviated with SaN, as we have seen from the improved model calibration in Table 2b. As a result, self-adaptation can cope sufficiently well with milder domain shift scenarios. As examples on GTA → Cityscapes and SYNTHIA → Mapillary in Fig. 5 show, despite the baseline model exhibiting some degree of this failure mode, self-adaptive inference visibly rectifies these errors.

We additionally ran our inference on video sequences and include the results as part of our supplementary material.

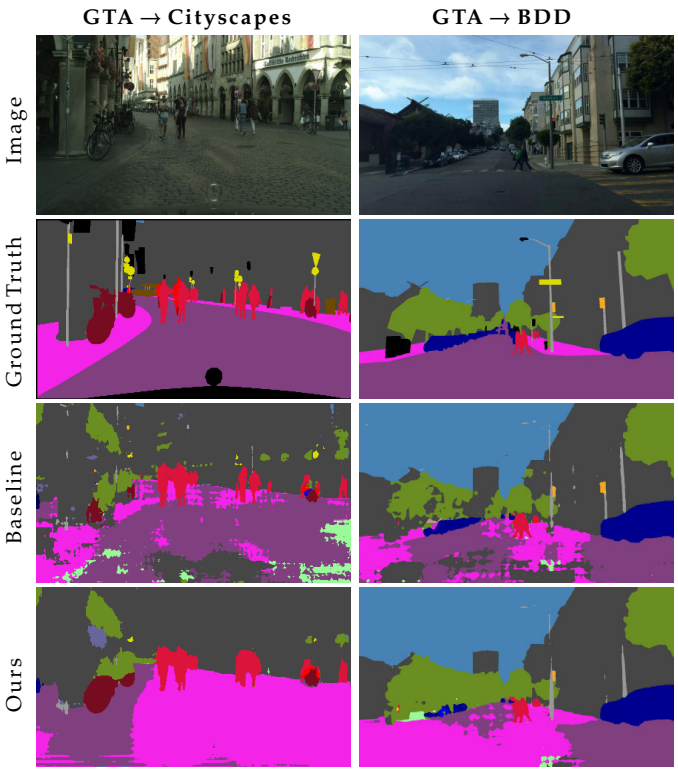

Fig. 6. *Failure cases of semantic segmentation for generalization results from GTA to Cityscapes and BDD* for ResNet-50. Self-adaptation may compound the errors in the baseline prediction. However, the frequency of this failure mode is low (see Fig. 7 and Sec. 6.9 for more detail).

Confirming our analysis of the qualitative results above, we observe that self-adaptation significantly improves the segmentation quality and removes some of the most pathological failure modes of the baseline (*e. g.*, the lower middle part of the frame area). Surprisingly, we also note an improvement in the temporal coherence between consecutive frames despite self-adaptation lacking any temporal regularisation.

6.9 Analyzing failure cases

Conceptually, if the initial semantic prediction is incorrect and confident, which may happen due to imperfect model calibration, such an error is likely to end up in the pseudo labels and lead astray the self-adaptation process. We already remarked on this limitation in the previous section; Fig. 6 illustrates further examples of this failure mode. Misguided by incorrect pseudo labels, self-adaptation may compound the initial error and "propagate" the incorrect label to the areas sharing the same appearance.

To estimate the impact of this issue on the empirical results, we analyze it statistically. The histograms in Fig. 7 show relative improvement of the self-adaptive inference strategy *w. r. t.* the baseline in terms of IoU on four validation sets. We observe that a decrease in segmentation accuracy is actually quite rare: less than 10% of the images, on average, exhibit lower accuracy. In most such cases the accuracy reduces only marginally (by less than 5%), and only a fraction of images (less than 1% on average) deteriorate in accuracy by at most 10%. The overwhelming majority of the image samples benefit from self-adaptive process, which increases their accuracy by up to 35% IoU compared to the baseline.

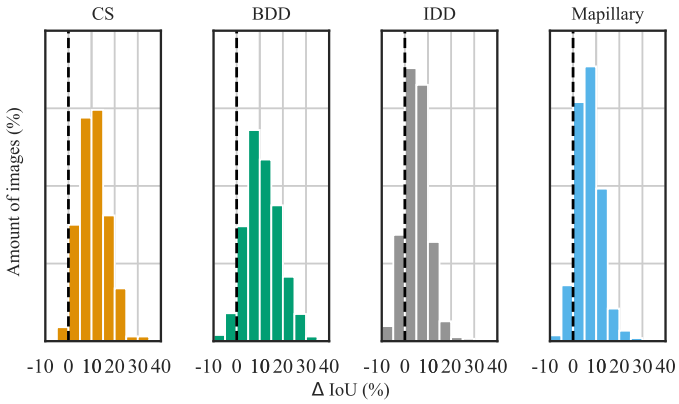

Fig. 7. *Empirical distribution of accuracy change on individual images for generalization from source domain (GTA) to target domains (Cityscapes, BDD, IDD, and Mapillary)* for the ResNet-50 backbone. We visualize the relative improvement of our self-adaptive inference strategy w.r.t. the baseline in terms of accuracy with Δ IoU (%).

7 Conclusion and Outlook

The *i.i.d.* assumption underlying the traditional learning principle, ERM, implies that a training process relying on it is unlikely to produce models robust to an arbitrary domain shift, unless we make further assumptions about the test distribution. In the out-of-distribution scenario, the test domain is unknown, hence formulating such assumptions is difficult. To by-pass this issue, we presented and studied a self-adaptive *inference* process. We also highlighted a number of shortcomings in the experimental protocol used in previous work. By following the best practice in machine learning research, we formulated four principles defining a rigorous evaluation process in domain generalization. We implemented and followed these principles in our experiments. Our analysis clearly demonstrates that a single sample from the test domain can already suffice to substantially improve model predictions. The accuracy improvement shown by our experiments is remarkably substantial, despite no changes to the training process or the model architecture, unlike in previous works [13], [76]. We hope that these encouraging results will incentivize our research community to study self-adaptive techniques in other application domains, such as panoptic segmentation, or monocular depth prediction.

The particular instantiation of self-adaptation that we presented in this work is not yet real-time. We extensively analyzed the existing trade-off with the segmentation accuracy in Sec. 6.6 and found self-adaptation to be nonetheless more cost-effective than model ensembles — a widely adopted framework in machine learning. However, our scope of this work was to rigorously demonstrate concrete accuracy benefits, which will make improving model efficiency a worthwhile goal to pursue in future work. Indeed, decreasing the latency of self-adaptive inference is an intriguing avenue for research. Using adaptive step sizes, higher-order optimization [7], [31] or implicit gradients [48], we may lower the number of required update steps. An orthogonal line of research may consider low-precision computations [68], low-rank decomposition [28], or alternating update strategies [58] to offer reduced computational footprint for each update iteration. We are excited to explore these directions and hope the reader may find this work equally inspiring.

REFERENCES

[1] Araslanov, N., Roth, S.: Self-supervised augmentation consistency for adapting semantic segmentation. In: CVPR. pp. 15384–15394 (2021)
[2] Arjovsky, M., Bottou, L., Gulrajani, I., Lopez-Paz, D.: Invariant risk minimization. arXiv:1907.02893 [stat.ML] (2019)
[3] Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv:1607.06450 [stat.ML] (2016)
[4] Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.W.: A theory of learning from different domains. Mach. Learn. **79**(1-2), 151–175 (2010)
[5] Bickel, S., Brückner, M., Scheffer, T.: Discriminative learning under covariate shift. J. Mach. Learn. Res. **10**, 2137–2155 (2009)
[6] Bonneel, N., van de Panne, M., Paris, S., Heidrich, W.: Displacement interpolation using Lagrangian mass transport. ACM Trans. Graph. **30**(6), 158 (2011)
[7] Botev, A., Ritter, H., Barber, D.: Practical Gauss-Newton optimisation for deep learning. In: ICML. vol. 70, pp. 557–565 (2017)
[8] Cai, M., Lu, F., Sato, Y.: Generalizing hand segmentation in egocentric videos with uncertainty-guided model adaptation. In: CVPR. pp. 14392–14401 (2020)
[9] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected CRFs. In: ICLR (2015)
[10] Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 834–848 (2018)
[11] Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: ECCV. vol. 7, pp. 833–851 (2018)
[12] Chen, W., Yu, Z., Mello, S.D., Liu, S., Alvarez, J.M., Wang, Z., Anandkumar, A.: Contrastive syn-to-real generalization. In: ICLR (2021)
[13] Chen, W., Yu, Z., Wang, Z., Anandkumar, A.: Automated synthetic-to-real generalization. In: ICML. vol. 119, pp. 1746–1756 (2020)
[14] Choi, S., Jung, S., Yun, H., Kim, J.T., Kim, S., Choo, J.: RobustNet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In: CVPR. pp. 11580–11590 (2021)
[15] Cordts, M., Omran, M., Ramos, S., Scharwächter, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The Cityscapes dataset for semantic urban scene understanding. In: CVPR. pp. 3213–3223 (2016)
[16] Deecke, L., Murray, I., Bilen, H.: Mode normalization. In: ICLR (2019)
[17] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Li, F.F.: ImageNet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009)
[18] Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML. vol. 70, pp. 1126–1135 (2017)
[19] Gal, Y., Ghahramani, Z.: Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: ICML. vol. 48, pp. 1050–1059 (2016)
[20] Ganin, Y., et al., E.U.: Domain-adversarial training of neural networks. J. Mach. Learn. Res. **17**, 59:1–59:35 (2016)
[21] Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: ICCV. pp. 349–356 (2009)
[22] Gulrajani, I., Lopez-Paz, D.: In search of lost domain generalization. In: ICLR (2021)
[23] Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Trans. Pattern Anal. Mach. Intell. **12**(10), 993–1001 (1990)
[24] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
[25] Huang, J., Guan, D., Xiao, A., Lu, S.: FSDR: Frequency space domain randomization for domain generalization. In: CVPR. pp. 6891–6902 (2021)
[26] Huang, L., Zhou, Y., Zhu, F., Liu, L., Shao, L.: Iterative normalization: Beyond standardization towards efficient whitening. In: CVPR. pp. 4874–4883 (2019)
[27] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. vol. 37, pp. 448–456 (2015)
[28] Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. In: BMVC (2014)
[29] Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1409–1422 (2012)
[30] Kim, J., Lee, J., Park, J., Min, D., Sohn, K.: Pin the memory: Learning to generalize semantic segmentation. In: CVPR. pp. 4350–4360 (2022)
[31] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
[32] Kundu, J.N., Kulkarni, A., Singh, A., Jampani, V., Babu, R.V.: Generalize then adapt: Source-free domain adaptive semantic segmentation. In: ICCV. pp. 7046–7056 (2021)
[33] Lee, S., Seong, H., Lee, S., Kim, E.: WildNet: Learning domain generalized semantic segmentation from the wild. In: CVPR. pp. 9936–9946 (2022)
[34] Li, D., Yang, Y., Song, Y.Z., Hospedales, T.M.: Deeper, broader and artier domain generalization. In: ICCV. pp. 5542–5550 (2017)
[35] Li, Y., Wang, N., Shi, J., Liu, J., Hou, X.: Revisiting batch normalization for practical domain adaptation. In: ICLR (2017)
[36] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV. pp. 9992–10002 (2021)
[37] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR. pp. 3431–3440 (2015)
[38] Long, M., Zhu, H., Wang, J., Jordan, M.I.: Unsupervised domain adaptation with residual transfer networks. In: NeurIPS. pp. 136–144 (2016)
[39] Luo, P., Ren, J., Peng, Z., Zhang, R., Li, J.: Differentiable learning-to-normalize via switchable normalization. In: ICLR (2019)
[40] Montavon, G., Orr, G.B., Müller, K. (eds.): Neural Networks: Tricks of the Trade - Second Edition, Lecture Notes in Computer Science, vol. 7700. Springer (2012)
[41] Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B., Snoek, J.: Evaluating prediction-time batch normalization for robustness under covariate shift. arXiv:2006.10963 [cs.LG] (2020)
[42] Naeini, M.P., Cooper, G.F., Hauskrecht, M.: Obtaining well calibrated probabilities using Bayesian binning. In: AAAI. pp. 2901–2907 (2015)
[43] Nam, H., Kim, H.: Batch-instance normalization for adaptively style-invariant neural networks. In: NeurIPS. pp. 2563–2572 (2018)
[44] Neuhold, G., Ollmann, T., Rota Bulò, S., Kontschieder, P.: The Mapillary Vistas dataset for semantic understanding of street scenes. In: ICCV. pp. 4990–4999 (2017)
[45] Pan, X., Luo, P., Shi, J., Tang, X.: Two at once: Enhancing learning and generalization capacities via IBN-Net. In: ECCV. vol. 4, pp. 484–500 (2018)
[46] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: PyTorch: An imperative style, high-performance deep learning library. In: NeurIPS. pp. 8024–8035 (2019)
[47] Peng, D., Lei, Y., Hayat, M., Guo, Y., Li, W.: Semantic-aware domain generalized segmentation. In: CVPR. pp. 2594–2605 (2022)
[48] Rajeswaran, A., Finn, C., Kakade, S.M., Levine, S.: Meta-learning with implicit gradients. In: NeurIPS. pp. 113–124 (2019)
[49] Read, A.: Linear interpolation of histograms. NIM-A **425**(1), 357–360 (1999)
[50] Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: ECCV. vol. 2, pp. 102–118 (2016)
[51] Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.M.: The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: CVPR. pp. 3234–3243 (2016)
[52] Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., Bethge, M.: Improving robustness against common corruptions by covariate shift adaptation. In: NeurIPS. pp. 11539–11551 (2020)
[53] Shaham, T.R., Dekel, T., Michaeli, T.: SinGAN: Learning a generative model from a single natural image. In: ICCV. pp. 4569–4579 (2019)
[54] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
[55] Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A.A., Hardt, M.: Test-time training with self-supervision for generalization under distribution shifts. In: ICML. vol. 119, pp. 9229–9248 (2020)
[56] Tang, Z., Gao, Y., Zhu, Y., Zhang, Z., Li, M., Metaxas, D.N.: CrossNorm and SelfNorm for generalization under distribution shifts. In: ICCV. pp. 52–61 (2021)
[57] Thrun, S.: Lifelong learning algorithms. In: Learning to Learn, pp. 181–209. Springer (1998)
[58] Tonioni, A., Tosi, F., Poggi, M., Mattoccia, S., di Stefano, L.: Real-time self-adaptive deep stereo. In: CVPR. pp. 195–204 (2019)

[59] Torralba, A., Efros, A.A.: Unbiased look at dataset bias. In: CVPR. pp. 1521–1528 (2011)

[60] Ulyanov, D., Vedaldi, A., Lempitsky, V.S.: Instance normalization: The missing ingredient for fast stylization. arXiv:1607.08022 [cs.CV] (2016)

[61] Varma, G., Subramanian, A., Namboodiri, A.M., Chandraker, M., Jawahar, C.V.: IDD: A dataset for exploring problems of autonomous navigation in unconstrained environments. In: WACV. pp. 1743–1751 (2019)

[62] Varsavsky, T., Orbes-Arteaga, M., Sudre, C.H., Graham, M.S., Nachev, P., Cardoso, M.J.: Test-time unsupervised domain adaptation. In: MICCAI. pp. 428–436 (2020)

[63] Volpi, R., Namkoong, H., Sener, O., Duchi, J.C., Murino, V., Savarese, S.: Generalizing to unseen domains via adversarial data augmentation. In: NeurIPS. pp. 5339–5349 (2018)

[64] Vu, T.H., Jain, H., Bucher, M., Cord, M., Pérez, P.: Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In: CVPR. pp. 2517–2526 (2019)

[65] Wang, D., Shelhamer, E., Liu, S., Olshausen, B., Darrell, T.: Tent: Fully test-time adaptation by entropy minimization. In: ICLR (2021)

[66] Wang, G., Peng, J., Luo, P., Wang, X., Lin, L.: Kalman normalization: Normalizing internal representations across network layers. In: NeurIPS. pp. 21–31 (2018)

[67] Wang, J., Sun, K., et al., T.C.: Deep high-resolution representation learning for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. **43**(10), 3349–3364 (2021)

[68] Wang, Y., Jiang, Z., Chen, X., Xu, P., Zhao, Y., Lin, Y., Wang, Z.: E2-train: Training state-of-the-art CNNs with over 80% energy savings. In: NeurIPS. pp. 5139–5151 (2019)

[69] Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Mach. Learn. **23**(1), 69–101 (1996)

[70] Wu, Y., He, K.: Group normalization. In: ECCV. vol. 8, pp. 3–19 (2018)

[71] Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: ECCV. vol. 5, pp. 418–434 (2018)

[72] Xie, S., Zheng, Z., Chen, L., Chen, C.: Learning semantic representations for unsupervised domain adaptation. In: ICML. pp. 5423–5432 (2018)

[73] Yang, Y., Soatto, S.: FDA: Fourier domain adaptation for semantic segmentation. In: CVPR. pp. 4084–4094 (2020)

[74] You, F., Li, J., Zhao, Z.: Test-time batch statistics calibration for covariate shift. arXiv:2110.04065 [cs.CV] (2021)

[75] Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., Darrell, T.: BDD100K: A diverse driving dataset for heterogeneous multitask learning. In: CVPR. pp. 2633–2642 (2020)

[76] Yue, X., Zhang, Y., Zhao, S., Sangiovanni-Vincentelli, A.L., Keutzer, K., Gong, B.: Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In: ICCV. pp. 2100–2110 (2019)

[77] Zendel, O., Honauer, K., Murschitz, M., Steininger, D., Domínguez, G.F.: WildDash – Creating hazard-aware benchmarks. In: ECCV. vol. 6, pp. 407–421 (2018)