

Introducing Intermediate Domains for Effective Self-Training during Test-Time

Robert A. Marsden*
University of Stuttgart

Mario Döbler*
University of Stuttgart

Bin Yang
University of Stuttgart

Abstract

Experiencing domain shifts during test-time is nearly inevitable in practice and likely results in a severe performance degradation. To overcome this issue, test-time adaptation continues to update the initial source model during deployment. A promising direction are methods based on self-training which have been shown to be well suited for gradual domain adaptation, since reliable pseudo-labels can be provided. In this work, we address two problems that exist when applying self-training in the setting of test-time adaptation. First, adapting a model to long test sequences that contain multiple domains can lead to error accumulation. Second, naturally, not all shifts are gradual in practice. To tackle these challenges, we introduce GTTA. By creating artificial intermediate domains that divide the current domain shift into a more gradual one, effective self-training through high quality pseudo-labels can be performed. To create the intermediate domains, we propose two independent variations: mixup and light-weight style transfer. We demonstrate the effectiveness of our approach on the continual and gradual corruption benchmarks, as well as ImageNet-R. To further investigate gradual shifts in the context of urban scene segmentation, we publish a new benchmark: CarlaTTA. It enables the exploration of several non-stationary domain shifts.[†]

1 Introduction

Deep neural networks achieve remarkable performance under the assumption that training and test data originate from the same distribution. However, when a neural network is deployed in the real world, this assumption is often

*Equal contribution.

[†]Code is available at: <https://github.com/mariodoebler/test-time-adaptation>

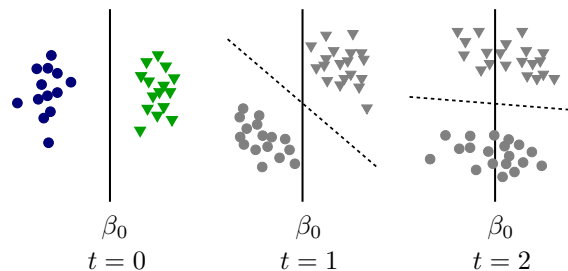


Figure 1: Illustration of a gradually evolving domain. In the beginning $t = 0$, perfect classification is possible. At time step $t = 1$, the domain has gradually changed and the original source decision boundary β_0 still separates both classes. When the domain further evolves ($t = 2$), the original decision boundary cannot separate the data anymore. When updating the model through self-training at time step $t = 1$, a good classification would still be possible at time step $t = 2$, as indicated by the dashed line.

violated. This effect is known as data shift (Quiñero-Candela et al., 2008) and leads to a potentially large drop in performance on the test data. While it is possible to improve robustness and generalization directly during training (Hendrycks et al., 2019, 2021; Muandet et al., 2013; Tobin et al., 2017; Tremblay et al., 2018), the effectiveness remains limited due to the wide range of potential data shifts (Mintun et al., 2021) that are unknown during training. Thus, another area of research, namely test-time adaptation (TTA), follows the idea to adapt the pre-trained source model during deployment, as the encountered test data provides information about the current distribution shift.

Recent work on TTA focuses on the setting where the model only has to adapt to a single test domain. Needless to say, in practice this setting is very unlikely; it is much more likely that a model encounters different domains without the knowledge when a change occurs. Q. Wang et al. (2022) denotes the setting where a model is adapted during deployment to a sequence of test domains as *continual test-time adaptation*. Due to potentially infinitely long test sequences and the encounter of different domain shifts, test-time adaptation, which is usually based on self-training and entropy minimization, is prone to error accumulation (Q. Wang et al., 2022).

Clearly, the larger a shift is, the more likely it becomes that a model introduces errors. In the case of self-training, this likely results in an unsuccessful model adaptation due to the lack of reliable pseudo-labels (Kumar et al., 2020). On the contrary, Kumar et al. (2020) showed both theoretically and empirically that a model can be adapted successfully if the experienced shifts are small enough. Hence, adaptation to large shifts can be successful if divided into smaller gradual shifts, as illustrated in Figure 1.

Looking at the nature of shifts in reality, for many applications they do not occur abruptly, but evolve gradually over time. While the change from day to night is only one example, Kumar et al. (2020) mentions, among others, evolving road conditions (Bobu et al., 2018) and sensor aging (Vergara et al., 2012). Of course, gradual shifts are not given in all settings or the gap of the experienced gradual shift is still too large for a successful model adaptation. Therefore, we propose to leverage source data to artificially create intermediate domains where, optimally, correct labels can be utilized to prevent the incorporation of additional errors. Even though requiring source data can be a limitation, we argue that having access to the initial source data is commonly the case. Now, for the creation of the intermediate domains we suggest two independent approaches: the first is based on mixup where the intermediate domains are created by linearly interpolating source and test images. The second idea uses a content-preserving light-weight style transfer model that is adapted online to new target styles. Since mixup and style transfer have their limitations and only mitigate the current domain gap, we rely on self-training to close the remaining gap. Assuming that mixup or style transfer moves the model closer to the test distribution, better self-training through more reliable pseudo-labels can be performed.

To demonstrate the effectiveness of our approach, we consider the continual and gradual corruption benchmark, as well as ImageNet-R. Due to the lack of datasets containing non-stationary domain shifts, we introduce and publish a new benchmark for the task of urban scene segmentation: CarlaTTA. It includes various non-stationary domain shifts in the setting of autonomous driving. We achieve new state-of-the-art results on all benchmarks. We summarize our main contributions as follows:

- We introduce a new framework *Gradual Test-time Adaptation* (GTTA), which conducts effective self-training by converting the current arbitrary domain shift into a gradual one. This is achieved by generating artificial intermediate domains using either mixup or light-weight style transfer.
- We publish a new benchmark for urban scene segmentation that enables the exploration of several non-stationary domain shifts during test-time in the field of autonomous driving.

2 Related Work

Unsupervised Domain Adaptation Recently, there has been a growing interest in mitigating the distributional discrepancy between two domains using unsupervised domain adaptation (UDA). Common approaches for UDA try to align either the input space (Hoffman et al., 2018; Marsden, Wiewel, et al., 2022; Z. Wu et al., 2019), the feature space (Q. Zhang et al., 2019; Marsden, Bartler, et al., 2022), the output space (Tsai et al., 2018, 2019), or several spaces in parallel (Y. Li et al., 2019; Yang & Soatto, 2020). One line of work relies on adversarial learning, where a domain classifier tries to discriminate whether some feature maps (Ganin & Lempitsky, 2015) or network outputs (Tsai et al., 2018; Vu et al., 2019) belong to the source or target domain. It is also possible to exploit adversarial learning or adaptive instance normalization (AdaIN) (Huang & Belongie, 2017) for transferring the target style to source images (Hoffman et al., 2018; Marsden, Wiewel, et al., 2022). Lately, self-training has gained a lot of attraction (Tranheden et al., 2021; Mei et al., 2020; P. Zhang et al., 2021; Hoyer et al., 2022; G. Li et al., 2020). Self-training utilizes a pre-trained (source) model to create predictions for the unlabeled target data. These predictions can then be treated as pseudo-labels to minimize, for example, the cross-entropy. Since high quality pseudo-labels are essential to this approach, most methods differ in how they select or create reliable pseudo-labels.

One-shot Unsupervised Domain Adaptation As pointed out in Luo et al. (2020), even collecting unlabeled target data can be challenging. Therefore, Luo et al. (2020) introduced one-shot UDA, where only one single target image is available during the model adaptation. To address this problem, Luo et al. (2020) extends the adaptive instance normalization framework of Huang and Belongie (2017) with a variational autoencoder. By selecting styles for which the segmentation model is uncertain, the domain gap is mitigated. Differently, X. Wu et al. (2021) uses a style mixing component within the segmentation model and further adds patch-wise prototypical matching.

Test-time Adaptation Although generalizing to any test distribution would solve many problems, the lack of information about the test environment during training imposes a great challenge. However, during model deployment, one can gain some insight into the test distribution by using the current test sample(s). This circumstance is also exploited in recent work, where Schneider et al. (2020) showed that even adapting the batch normalization (BN) statistics during test-time can significantly improve the performance on corrupted data. More sophisticated approaches perform source model optimization during test-time. For example, D. Wang et al. (2020) update the BN layers by entropy minimization. M. Zhang et al. (2021) create an ensemble prediction through test-time augmentation (Krizhevsky et al.,

2009) and then minimize the entropy with respect to all parameters. Other methods rely on self-supervised learning, using either pre-text tasks to adapt the model (Y. Sun et al., 2020; Liu et al., 2021; Bartler et al., 2022) or apply contrastive learning (D. Chen et al., 2022). Recent works make use of diversity regularizers (Liang et al., 2020; Mummadi et al., 2021) to prevent the collapse to trivial solutions potentially caused by confidence maximization.

Continual Test-time Adaptation Continual test-time adaptation considers online TTA with continually changing target domains. While some of the existing methods can be applied to the continual setting, such as the online version of TENT (D. Wang et al., 2020), they are often prone to error accumulation due to miscalibrated predictions (Q. Wang et al., 2022). CoTTA (Q. Wang et al., 2022) uses weight and augmentation-averaged predictions to reduce error accumulation and stochastic restore to circumvent catastrophic forgetting (McCloskey & Cohen, 1989).

Gradual Domain Adaptation Recent work has indicated that when the domain discrepancy is too large, adapting a model through self-training can be very challenging due to noisy pseudo-labels (Kumar et al., 2020). Therefore, numerous methods consider the setting of gradual domain adaptation (Hoffman et al., 2014; H.-Y. Chen & Chao, 2021; Y. Zhang et al., 2021), where several intermediate domains exist between source and target. While some of the proposed approaches successively adapt the model using adversarial learning (Wulfmeier et al., 2018; Bobu et al., 2018), it has been shown that self-training can be very powerful in this setting (Kumar et al., 2020).

3 Methodology

Since in many practical applications environmental conditions can change over time, a model pre-trained on source data $(\mathcal{X}^S, \mathcal{Y}^S)$ can quickly become sub-optimal for the current test data x_t^T at time step t . Online test-time adaptation counteracts the performance deterioration by updating the model based on the current test data x_t^T . As already presented by the theory for gradual domain adaptation (Kumar et al., 2020), self-training can be particularly successful when guaranteed that the domain shift is small enough. Clearly, in reality this is not always given, since the domain shift can occur at different rates and severities. Therefore, in this work, we present a framework, depicted in Figure 2, that performs TTA in two steps: First, current test images x_t^T and a batch of randomly sampled source images x^S are utilized to generate an intermediate domain. Since we rely on content-preserving methods to create intermediate domains, the transformed images $\tilde{x}(x_t^T, x^S)$ and the corresponding source labels y^S are used to minimize the cross-entropy loss $\mathcal{L}_{CE}^S(y^S, \tilde{p})$, where \tilde{p} are the softmax predictions of the transformed images \tilde{x} . In a second step, reliable self-training can now be carried out to close the re-

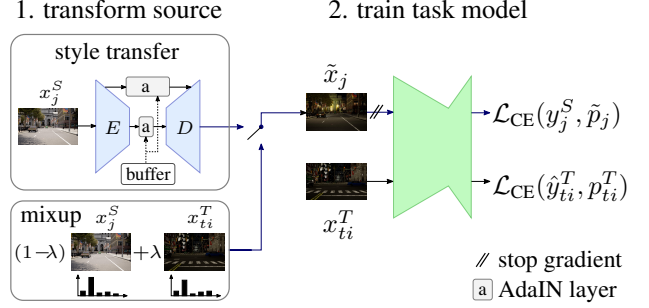


Figure 2: Our framework GTTA performs test-time adaptation in two steps. First, utilizing the current test samples, we create intermediate domains by transforming a batch of source samples. This is accomplished by relying either on mixup or style transfer based on AdaIN layers. The transformed samples and the corresponding source labels are subsequently used to minimize a cross-entropy loss $\mathcal{L}_{CE}^S(y^S, \tilde{p}^S)$. Second, self-training is performed with filtered pseudo-labels by minimizing the cross-entropy $\mathcal{L}_{CE}^T(\hat{y}_t^T, p_t^T)$.

maining domain gap. This is accomplished by minimizing the cross-entropy $\mathcal{L}_{CE}^T(\hat{y}_t^T, p_t^T)$, using sharpened softmax outputs \hat{y}_t^T from the current test images x_t^T and the corresponding softmax predictions p_t^T .

3.1 Generating intermediate domains

To generate intermediate domains, we propose two ideas: mixup known for improving robustness (H. Zhang et al., 2017) and content-preserving light-weight style transfer. Either mixup or style transfer can be chosen depending on the type of domain shifts and computational requirements.

Mixup The original idea of mixup is that linear interpolations in the input space should lead to linear interpolations in the output space. Since we do not want to introduce additional label-noise during test-time, we adapt the original idea of mixup in the sense that we do not interpolate labels. Instead we only rely on our noise-free source labels and linearly interpolate between source and test samples to close the gap between these two domains:

$$\tilde{x}_j = (1 - \lambda_{\text{mix}})x_j^S + \lambda_{\text{mix}}x_{ti}^T. \quad (1)$$

To reduce the mixup of samples belonging to different classes, we interpolate source sample x_j^S with the test sample x_{ti}^T from the current test batch x_t^T that has the highest similarity in terms of the largest dot product in the output softmax probability space. Investigations about the mixup strength λ_{mix} are presented in Appendix A.4.

Style transfer Another possibility to create intermediate domains is to leverage style transfer. However, performing style transfer during test-time imposes some challenges. It should be of light weight to enable real-time process-

ing and the network should be easily trainable during test-time, even when only having one test sample at a time. While Kim et al. (2021) introduces a method for photo-realistic style transfer during test-time, it is not suitable for our setting since it takes tens of seconds to transfer one single image-pair. This is similar to style transfer based on adversarial learning (Isola et al., 2017; Zhu et al., 2017), which can be unstable during training. Therefore, we follow Huang and Belongie (2017) and use a VGG19 based network that performs style transfer through an adaptive instance normalization (AdaIN) layer. This layer assumes that the style is mostly contained in the first two moments. In our case, the AdaIN layer re-normalizes a content feature map z_j^S belonging to source image x_j^S to have the same channel-wise mean μ and standard deviation σ as a style feature map z_{ti}^T extracted from the i -th test image x_{ti}^T :

$$\tilde{z}_j = \sigma(z_{ti}^T) \frac{z_j^S - \mu(z_j^S)}{\sigma(z_j^S)} + \mu(z_{ti}^T). \quad (2)$$

Since z_j^S and z_{ti}^T are extracted with an ImageNet pre-trained and frozen VGG19 encoder, the network only needs to be trained with respect to the decoder’s parameters. Now, let E , D , and $\tilde{x}_j = D(\tilde{z}_j)$ be the encoder, the decoder, and the transferred source image, respectively, then the loss minimized by the decoder can be written as:

$$\mathcal{L} = \lambda_s \sum_{l=1}^L \left[\text{MSE}(\mu(E_l(\tilde{x}_j)), \mu(E_l(x_{ti}^T))) \right. \quad (3) \\ \left. + \text{MSE}(\sigma(E_l(\tilde{x}_j)), \sigma(E_l(x_{ti}^T))) \right] + \text{MSE}(E(\tilde{x}_j), \tilde{z}_j)$$

where MSE is the mean-squared-error, $E_l()$ represents the output of the l -th layer of the encoder, and λ_s is a weighting term, set to $\lambda_s = 0.1$. Since images for the task of urban scene segmentation usually contain multiple classes, which may also have different styles, we follow Marsden, Wiewel, et al. (2022) in this case and use class-specific moments to calculate Eq. 2 and Eq. 3. These moments are extracted using a resized version of the source segmentation mask for the content feature map and pseudo-labels for the style feature map. The target moments are stored in style memory Q , allowing to transfer source images into previous styles.

3.2 Self-training

Self-training first converts the N_t softmax outputs $\{p_{ti}^T\}_{i=1}^{N_t}$ of the model for the current test images $\{x_{ti}^T\}_{i=1}^{N_t}$ at time step t into pseudo-labels $\hat{y}_{ti}^T = \text{argmax}(p_{ti}^T)$. These pseudo-labels are subsequently used to minimize the following cross-entropy loss

$$\mathcal{L}_{\text{CE}}^T(\hat{y}_t^T, p_t^T) = -\frac{1}{N_t} \sum_{i=1}^{N_t} \sum_{c=1}^C \hat{y}_{ti}^T \log(p_{tic}^T), \quad (4)$$

where C denotes the total number of classes. Clearly, most problems in reality are not as simple as depicted in Figure

1, and there can already be erroneous predictions within the training domain. Since the amount of incorrect predictions can increase, especially when a domain change occurs, it is important to prevent the accumulation of the initial and subsequent errors. A factor that amplifies error accumulation when conducting self-training with pseudo-labels is that the cross-entropy loss has large gradient magnitudes for uncertain predictions (Mummadi et al., 2021). Since it is mostly the uncertain predictions that tend to be incorrect, their incorporation into the training process will prevent a successful adaption to the current target domain. This problem can be mitigated by using a threshold which filters out all pseudo-labels below a certain (softmax) confidence level. Although defining a fixed threshold can work well when adapting the model to a single target domain, it is insufficient for a test sequence containing multiple domains. In addition, different models and problems tend to have different confidences: Over-confident networks naturally have high confidences, while datasets with many classes tend to be less confident. Therefore, we introduce an adaptively smoothed threshold with momentum $\alpha_{\text{th}} = 0.1$ that leverages the current softmax probabilities as follows:

$$\gamma_t = (1 - \alpha_{\text{th}}) \gamma_{t-1} + \alpha_{\text{th}} \sqrt{\frac{1}{N_t} \sum_{i=1}^{N_t} \max_c(p_{tic}^T)}. \quad (5)$$

4 Dataset: Gradual Domain Changes for Urban Scene Segmentation

Currently, there are not many datasets that are suited for investigating gradual test-time adaptation. Even though there already exist various real-world and synthetic driving datasets that contain different domains, such as Cityscapes (Cordts et al., 2016), ACDC (Sakaridis et al., 2021), Waymo (P. Sun et al., 2020), BDD100K (Yu et al., 2020), SYNTHIA (Ros et al., 2016), and GTA5 (Richter et al., 2016), they all involve only stationary domains and no sequences with gradual changes. To close this gap we introduce CarlaTTA: a dataset that enables the exploration of gradual test-time adaptation for urban scene segmentation. It is based on CARLA (Dosovitskiy et al., 2017), an open-source simulator for autonomous driving research. We create five gradual test-sequences, all evolving from the stationary source domain *clear* which is recorded at noon in clear weather. *day2night* depicts one complete day-night cycle by varying the sun altitude and sun azimuth angle. Different weather changes are addressed by the sequences *clear2fog* and *clear2rain*, where *clear2fog* changes cloudiness and fog density, while *clear2rain* varies cloudiness, precipitation, puddles, and wetness. *dynamic* combines the domain changes *day2night*, *clear2fog*, and *clear2rain*. Not only does it result in overlapping domain shifts, but also introduces new shifts, such as, reflecting lights during a rainy night. To also investigate long-term behavior, *dy-*



Figure 3: CarlaTTA: A synthetic driving dataset to explore gradual domain shifts in urban scenes.

dynamic contains multiple day-night and weather cycles resulting in a five times longer sequence. *highway* builds on top of the dynamic weather setting. In contrast to the previous datasets which mainly introduce covariate shifts, *highway* also introduces label distribution shifts, since the vehicle drives from the city onto the highway. Example images are shown in Figure 3. Further visualizations and insights into our dataset, including a detailed illustration of the weather parameters, are presented in Appendix C.

5 Experiments

Baselines Since BN has proven to be very effective during test-time (Schneider et al., 2020), we consider several variations that can be derived from the following equations:

$$\begin{aligned}\mu_{tm} &= (1 - \alpha)\hat{\mu}_m^S + \alpha\mu_{tm}^T \\ \sigma_{tm} &= (1 - \alpha)\hat{\sigma}_m^S + \alpha\sigma_{tm}^T.\end{aligned}\quad (6)$$

While $(\hat{\mu}_m^S, \hat{\sigma}_m^S)$ denote the running mean and standard deviation of channel m estimated during source training, $(\mu_{tm}^T, \sigma_{tm}^T)$ are the corresponding moments extracted from the current test batch at time step t . By using Eq. 6, the notation of BN related baselines can be harmonized: $\alpha = 0$ refers to the commonly known *source* baseline (BN-0), $\alpha = 1$ only exploits the current test statistics (BN-1), and $\alpha = 0.1$ leverages the source statistics as a prior (BN-0.1). However, none of them exploits gradual domain shifts, since Eq. 6 is instantaneous at time step t . Therefore, we introduce BN-EMA, which incorporates previous domain shifts by performing an exponential moving average using the running statistics from time step $(t - 1)$:

$$\begin{aligned}\hat{\mu}_{tm} &= (1 - \alpha)\hat{\mu}_{(t-1)m} + \alpha\mu_{tm}^T \\ \hat{\sigma}_{tm} &= (1 - \alpha)\hat{\sigma}_{(t-1)m} + \alpha\sigma_{tm}^T.\end{aligned}\quad (7)$$

To further evaluate our method, we compare to several approaches from related fields: TENT (D. Wang et al., 2020)

uses BN-1 in combination with an entropy minimization strategy with respect to the BN parameters. CoTTA (Q. Wang et al., 2022) utilizes BN-1 and a mean teacher with test-time augmentation to perform entropy minimization. Further it introduces stochastic restore, where source pre-trained weights are restored with a certain probability. AdaContrast (D. Chen et al., 2022) uses pseudo-label refinement for self-training and contrastive learning.

For segmentation, we additionally consider MEMO (M. Zhang et al., 2021), which combines test-time augmentation and entropy minimization and two methods from one-shot UDA. While ASM (Luo et al., 2020) uses an AdaIN based style transfer model to explore the style space, SM-PPM (X. Wu et al., 2021) integrates style mixing into the segmentation network and combines it with patch-wise prototypical matching.

5.1 Adapting to shifts caused by corruptions

Corruption benchmarks CIFAR10C, CIFAR100C, and ImageNet-C were originally published to evaluate robustness of neural networks (Hendrycks & Dietterich, 2019). The benchmark comprises of 15 corruptions with 5 severity levels, which were applied to the validation images of ImageNet (Deng et al., 2009) and the test images of CIFAR10 and CIFAR100 (Krizhevsky et al., 2009), respectively. In accordance with the RobustBench benchmark (Croce et al., 2020), a pre-trained WideResNet-28 is used for CIFAR10-to-CIFAR10C, ResNeXt-29 for CIFAR100-to-CIFAR100C, and ResNet-50 for ImageNet-to-ImageNet-C. Following the implementation and hyperparameters of Q. Wang et al. (2022), a batch size of 200 is utilized for CIFAR and a batch size of 64 for ImageNet. Note that we also investigate single sample test-time adaptation in Appendix A.5. We use Adam (Kingma & Ba, 2014) as an optimizer with a fixed learning rate of $1e-5$ for all experiments. Due to the low-resolution images of CIFAR, we only consider the mixup variant GTTA-MIX for CIFAR10C and CI-

Table 1: Classification error rate (%) on the corruption benchmark for the online continual test-time adaptation task on the highest corruption severity level 5. We report the performance of our method averaged over 5 runs.

	Time		$t \longrightarrow$																	
	Method	source-free	updates	<i>Gaussian</i>	<i>shot</i>	<i>impulse</i>	<i>defocus</i>	<i>glass</i>	<i>motion</i>	<i>zoom</i>	<i>snow</i>	<i>frost</i>	<i>fog</i>	<i>brightness</i>	<i>contrast</i>	<i>elastic-trans</i>	<i>pixelate</i>	<i>jpeg</i>	Mean	
CIFAR10C	BN-0 (src.)	✓	-	72.3	65.7	72.9	46.9	54.3	34.8	42.0	25.1	41.3	26.0	9.3	46.7	26.6	58.5	30.3	43.5	
	BN-1	✓	-	28.1	26.1	36.3	12.8	35.3	14.2	12.1	17.3	17.4	15.3	8.4	12.6	23.8	19.7	27.3	20.4	
	TENT-cont.	✓	1	24.8	20.6	28.6	14.4	31.1	16.5	14.1	19.1	18.6	18.6	12.2	20.3	25.7	20.8	24.9	20.7	
	AdaContrast	✓	1	29.1	22.5	30.0	14.0	32.7	14.1	12.0	16.6	14.9	14.4	8.1	10.0	21.9	17.7	20.0	18.5	
	CoTTA	✓	1	24.3	21.3	26.6	11.6	27.6	12.2	10.3	14.8	14.1	12.4	7.5	10.6	18.3	13.4	17.3	16.2	
	GTТА-MIX	✗	1	26.0	21.5	29.7	11.1	30.0	12.2	10.5	15.1	14.1	12.3	7.5	10.0	20.4	15.8	21.4	17.2±0.06	
	GTТА-MIX	✗	4	23.4	18.3	25.5	10.1	27.3	11.6	10.1	14.1	13.0	10.9	7.4	9.0	19.4	14.5	19.8	15.6±0.04	
CIFAR100C	BN-0 (src.)	✓	-	73.0	68.0	39.4	29.3	54.1	30.8	28.8	39.5	45.8	50.3	29.5	55.1	37.2	74.7	41.2	46.4	
	BN-1	✓	-	42.1	40.7	42.7	27.6	41.9	29.7	27.9	34.9	35.0	41.5	26.5	30.3	35.7	32.9	41.2	35.4	
	TENT-cont.	✓	1	37.2	35.8	41.7	37.9	51.2	48.3	48.5	58.4	63.7	71.1	70.4	82.3	88.0	88.5	90.4	60.9	
	AdaContrast	✓	1	42.3	36.8	38.6	27.7	40.1	29.1	27.5	32.9	30.7	38.2	25.9	28.3	33.9	33.3	36.2	33.4	
	CoTTA	✓	1	40.1	37.7	39.7	26.9	38.0	27.9	26.4	32.8	31.8	40.3	24.7	26.9	32.5	28.3	33.5	32.5	
	GTТА-MIX	✗	1	39.4	34.4	36.6	24.7	36.8	26.6	24.3	30.1	28.9	34.6	22.8	25.1	30.7	26.9	34.7	30.4±0.01	
	GTТА-MIX	✗	4	36.4	32.1	34.0	24.4	35.2	25.9	23.9	28.9	27.5	30.9	22.6	23.4	29.4	25.5	33.3	28.9±0.02	
ImageNetC	BN-0 (src.)	✓	-	97.8	97.1	98.2	81.7	89.8	85.2	78.0	83.5	77.1	75.9	41.3	94.5	82.5	79.3	68.6	82.0	
	BN-1	✓	-	85.0	83.7	85.0	84.7	84.3	73.7	61.2	66.0	68.2	52.1	34.9	82.7	55.9	51.3	59.8	68.6	
	TENT-cont.	✓	1	81.6	74.6	72.7	77.6	73.8	65.5	55.3	61.6	63.0	51.7	38.2	72.1	50.8	47.4	53.3	62.6	
	AdaContrast	✓	1	82.9	80.9	78.4	81.4	78.7	72.9	64.0	63.5	64.5	53.5	38.4	66.7	54.6	49.4	53.0	65.5	
	CoTTA	✓	1	84.7	82.1	80.6	81.3	79.0	68.6	57.5	60.3	60.5	48.3	36.6	66.1	47.2	41.2	46.0	62.7	
	GTТА-MIX	✗	1	80.5	74.7	72.4	77.8	75.7	64.3	54.0	57.0	58.6	44.6	33.9	67.5	49.4	44.7	49.3	60.3±0.07	
	GTТА-MIX	✗	4	75.2	67.4	64.6	73.3	72.5	61.8	52.7	53.0	54.9	42.6	33.8	63.9	48.9	44.4	47.0	57.1±0.14	
	GTТА-ST	✗	1	80.6	74.1	74.3	76.8	74.9	62.3	53.9	56.4	58.0	44.1	33.4	62.2	48.6	44.9	50.4	59.7±0.08	
	GTТА-ST	✗	4	76.7	69.0	69.3	73.1	72.1	59.6	51.6	53.4	56.7	42.9	33.7	57.2	47.9	43.4	47.9	57.0±0.06	

FAR100C. A mixup strength of $\lambda_{\text{mix}} = \frac{1}{3}$ is used for all experiments. For ImageNet-C, we additionally compare to the style transfer variant GTТА-ST. The style transfer network consists of the same VGG19 based encoder-decoder architecture as used in (Marsden, Wiewel, et al., 2022) and is pre-trained for 20k iterations on the source domain using Adam with learning rate 1×10^{-4} .

Continual corruption benchmarks We first consider the continual TТА setting, as proposed in Q. Wang et al. (2022). Starting with a network pre-trained on source data, the model is adapted during test-time in an online fashion. Unlike the standard setting where the model is reset before being adapted to a new corruption type, the continual setting does not assume to have any knowledge about the current domain or shift. Test-time adaptation is performed under the highest corruption severity level 5.

The results are reported in Table 1. Simply evaluating the pre-trained source model yields an average error of 43.5% for CIFAR10C, 46.4% for CIFAR100C, and 82.0%

for ImageNet-C. Using the current test batch to adapt the batch statistics (BN-1) already drastically decreases the error for all datasets. As already pointed out by Q. Wang et al. (2022), TENT-continual outperforms BN-1 in early stages, but quickly deteriorates after a few corruptions. This becomes particularly evident for CIFAR100C, where TENT achieves an error of 90.4% for the last corruption. To avoid error accumulation, one can use TENT-episodic instead. However, in the episodic setup, knowledge from previous examples cannot be leveraged, resulting in a performance on par with BN-1. Another option to stabilize the training which we investigate in Appendix A.1 is source replay. This has the restriction of requiring access to source data, but stabilizes self-training and improves the performance, e.g., for TENT on all datasets. CoTTA shows its strong suits for CIFAR10C outperforming BN-1 by 4.2% and performs comparably to TENT on ImageNet-C. AdaContrast outperforms BN-1, but lacks behind CoTTA on all datasets. Our method GTТА successfully shows on all datasets that generating intermediate domains by mixup or

Table 2: Average classification error rate (%) for the gradual corruption benchmark across all 15 corruptions. We separately report the performance averaged over all severity levels (level 1–5) and averaged only over the highest severity level 5 (level 5). The number in brackets denotes the difference compared to the continual benchmark.

		BN-0	BN-1	TENT-cont.	AdaCont.	CoTTA	GTТА-MIX	GTТА-MIX	GTТА-ST	GTТА-ST
	src.-free updates	✓	✓	✓	✓	✓	✗	✗	✗	✗
		-	-	1	1	1	1	4	1	4
CIFAR10C	level 1–5	24.7	13.7	20.4	12.1	10.9	10.5	9.5	-	-
	level 5	43.5	20.4	25.1 (+4.4)	15.8 (-2.7)	14.2 (-2.0)	15.0 (-2.2)	13.0 (-2.6)	-	-
CIFAR100C	level 1–5	33.6	29.9	74.8	33.0	26.3	24.3	23.9	-	-
	level 5	46.4	35.4	75.9 (+15.0)	35.9 (+2.5)	28.3 (-4.2)	27.6 (-2.8)	26.1 (-2.8)	-	-
ImageNetC	level 1–5	58.4	48.3	46.4	66.3	38.8	39.3	37.7	39.8	38.7
	level 5	82.0	68.6	58.9 (-3.7)	72.6 (+7.1)	43.1 (-19.6)	51.8 (-8.5)	47.7 (-9.4)	51.9 (-7.8)	48.3 (-8.7)

style transfer allows a better adaptation via self-training. Performing a single update per test batch leads to state-of-the-art results on CIFAR100C and ImageNet-C. Performing four updates results in a further improvement on all datasets and also sets state-of-the-art results on CIFAR10C. Note that utilizing more update steps for source-free methods like CoTTA does not improve the performance, on the contrary. A more in-depth analysis about when multiple update steps are beneficial is discussed in Appendix A.1.

Gradual corruption benchmarks We now investigate a setting, where the test domain changes gradually. Starting from the lowest severity level 1, the severity level is incremented as follows: $1 \rightarrow 2 \rightarrow \dots \rightarrow 5 \rightarrow \dots \rightarrow 2 \rightarrow 1$. After a cycle, we switch to the next corruption type and sequentially repeat the same procedure for all corruptions.

The results for the gradual setting are reported in Table 2. For a direct comparison between the results of the continual and gradual setting, we report the average error at the highest severity level 5 in addition to the average over all severities. The effect that TENT-continual’s performance deteriorates over time is even more prominent in the gradual setup due to longer sequences. TENT and AdaContrast show both for some of the datasets a worse performance at level 5 compared to the continual setup. CoTTA and GTТА both benefit from the gradually changing domains. GTТА-MIX shows the best performance on all datasets, except for ImageNet-C at level 5 where CoTTA takes the lead. GTТА-ST performs slightly worse than GTТА-MIX.

5.2 Adapting to real-world distribution shifts

ImageNet-R To investigate the performance in the presence of distribution shifts not caused by corruptions, we also analyze ImageNet-R (Hendrycks et al., 2021) using the same setting as for ImageNet-C. ImageNet-R consists of 30,000 samples depicting several renditions of 200 ImageNet classes. The results are shown in Table 3. While GTТА-MIX again outperforms previous methods on this benchmark, GTТА-ST shows a tremendous improvement.

Table 3: Classification error rate (%) for ImageNet-R averaged over 5 runs using a ResNet-50.

Method	source-free	updates	error rate
BN-0 (source)	✓	-	63.8
BN-1	✓	-	60.4
TENT-cont.	✓	1	57.6
AdaContrast	✓	1	59.1
CoTTA	✓	1	57.4
GTТА-MIX	✗	1	56.4±0.23
GTТА-MIX	✗	4	56.6±0.76
GTТА-ST	✗	1	53.8±0.19
GTТА-ST	✗	4	52.5±0.24

Comparing GTТА-MIX and GTТА-ST We find that mixup is especially suited for compensating domain gaps covered by the corruption benchmark and not necessarily for real-world distribution shifts where style transfer demonstrates its advantages. Since mixup in our case is a linear combination of source and test images, it is intuitive, that GTТА-MIX particularly performs well on corruptions that are additive. Examples are, Gaussian noise, snow, frost, and fog. When it comes to natural distribution shifts, such as introduced by ImageNet-R, mixup has its limitations. In contrast, style transfer based on adaptive instance normalization can perform arbitrary style transfer, as shown by the original work (Huang & Belongie, 2017). Even though GTТА-ST can cope with various domain shifts, as established by the results on ImageNet-C and ImageNet-R, it has a slight memory and computational overhead due to the additional style transfer network.

5.3 Experiments on CarlaTTA

Setup To demonstrate the effectiveness of our approach for natural shifts in the context of autonomous driving, we consider the CarlaTTA benchmark below. We report the mean intersection-over-union (mIoU) over the entire test sequence. All methods use the same pre-trained source

Table 4: mIoU (%) for CarlaTTA. The source model achieves 78.4% mIoU on the *clear* test split.

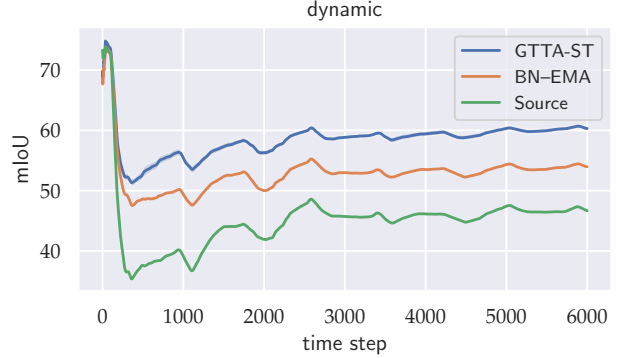
Method	src.-free	day2night	clear2fog	clear2train	dynamic	highway
BN=0 (source)	✓	58.4	52.8	71.8	46.6	28.7
BN=0.1	✓	62.7	56.5	72.8	52.1	37.2
BN=1	✓	62.0	56.8	71.4	52.6	32.8
BN-EMA	✓	63.4	58.3	73.4	53.9	31.9
MEMO	✓	61.0	55.1	71.6	50.3	35.2
TENT-cont.	✓	61.5	56.0	70.9	50.3	32.0
TENT-ep.	✓	61.9	56.8	71.4	52.6	32.8
CoTTA	✓	61.4	56.8	70.7	46.4	33.8
ASM	✗	58.5	53.0	69.2	50.2	39.4
SM-PPM	✗	63.1	56.7	72.7	53.2	33.4
self-training	✗	63.2	54.1	74.4	50.3	33.2
style-transfer	✗	66.0	62.2	74.6	59.1	41.9
GTТА-ST	✗	66.7	61.6	74.7	60.3	44.8

model trained for 100k iterations on the stationary source domain *clear*. To prevent overfitting to the source domain, we apply random horizontal flipping, Gaussian blur, color jittering, as well as random scaling in the range $[0.75, 2]$ before the image is cropped to a size of 1024×512 . Following the standard framework in UDA for semantic segmentation (Tsai et al., 2018), we use the DeepLab-V2 (L.-C. Chen et al., 2017) architecture with a ResNet-101 backbone. The style transfer network consists of the same architecture as described in Section 5.1, with the only difference that now class-conditional AdaIN layers are used.

Implementation details The segmentation model is trained with SGD using a constant learning of 2.5×10^{-4} , momentum of 0.9, and weight decay of 5×10^{-4} . During test-time adaptation, we use batches consisting of two source samples and two crops of the current test sample. While one of the source samples is transferred into the current test style, the other is transferred into a previously seen style, as domain shifts may reoccur. During the online adaptation, both networks are updated once for each new test sample.

5.3.1 Results for CarlaTTA

Our results are summarized in Table 4. As expected, BN=0 (source) performs the worst by a large margin. While BN-EMA outperforms for all scenarios except *highway*, BN=0.1 is absolutely 4.4% better than the second best (BN=1) on the *highway* split. Regarding TENT, we find that the episodic setting performs better than the continual setting. Nevertheless, both variants cannot surpass BN=1. Following X. Wu et al. (2021), we evaluate ASM without the attention module and use 4 updates per test sample. For SM-PPM, we get the best results using 8 adaptation steps.

Figure 4: mIoU up to time step t for the *dynamic* sequence.

SM-PPM performs better than TENT or ASM, however, it is still slightly worse on average compared to BN-EMA. CoTTA does not perform better than BN=1 and performance significantly drops for the longer *dynamic* sequence. We attribute this to the circumstances that the mean teacher always lags behind the current test domain.

In contrast, our approach GTТА-ST substantially outperforms all baselines by a large margin. Compared to the source model, the mIoU increases by more than 8% in four out of five cases. While self-training alone only provides a clear advantage in two cases, it cannot effectively exploit the gradual domain shift in this setting and even suffers from error accumulation. Style transfer, on the other hand, has a clear advantage in all evaluation settings, since it does not introduce any error accumulation due to label-noise. The combination of both methods now increases the performance on *day2night*, *dynamic*, and *highway* as through the intermediate domain introduced by style transfer, self-training benefits from more reliable pseudo-labels.

In Figure 4, we illustrate the mIoU up to time step t for the *dynamic* sequence. While the performance of the source model suffers heavily from the domain shift, GTТА-ST is able to maintain good performance throughout the test sequence. Further visualizations and ablation studies are located in Appendix B.

6 Conclusion

In this work we addressed current challenges in online continual and gradual test-time adaptation. Through the creation of intermediate domains by mixup or style transfer, successful self-training for arbitrary domain shifts can be performed. This is supported by experiments for the various gradual changes covered by CarlaTTA and the continual and gradual corruption benchmarks. On all presented benchmarks, we outperform existing methods by a large margin. We are certain that CarlaTTA will give other researchers the opportunity to further investigate the setting of gradual test-time adaptation.

7 Acknowledgments

This publication was created as part of the research project “KI Delta Learning” (project number: 19A19013R) funded by the Federal Ministry for Economic Affairs and Energy (BMWi) on the basis of a decision by the German Bundestag.

References

- Bartler, A., Bühler, A., Wiewel, F., Döbler, M., & Yang, B. (2022). Mt3: Meta test-time training for self-supervised test-time adaptation. In *International conference on artificial intelligence and statistics* (pp. 3080–3090).
- Bobu, A., Tzeng, E., Hoffman, J., & Darrell, T. (2018). Adapting to continuously shifting domains.
- Chen, D., Wang, D., Darrell, T., & Ebrahimi, S. (2022). Contrastive test-time adaptation. In *Cvpr*.
- Chen, H.-Y., & Chao, W.-L. (2021). Gradual domain adaptation without indexed intermediate domains. *Advances in Neural Information Processing Systems*, 34.
- Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4), 834–848.
- Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., ... Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3213–3223).
- Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., ... Hein, M. (2020). Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *Cvpr09*.
- Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st annual conference on robot learning* (pp. 1–16).
- Ganin, Y., & Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *International conference on machine learning* (pp. 1180–1189).
- Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., ... others (2021). The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 8340–8349).
- Hendrycks, D., & Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*.
- Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., & Lakshminarayanan, B. (2019). Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*.
- Hoffman, J., Darrell, T., & Saenko, K. (2014). Continuous manifold based adaptation for evolving visual domains. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 867–874).
- Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., ... Darrell, T. (2018). Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning* (pp. 1989–1998).
- Hoyer, L., Dai, D., & Van Gool, L. (2022). Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 9924–9935).
- Huang, X., & Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the ieee international conference on computer vision* (pp. 1501–1510).
- Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1125–1134).
- Kim, S., Kim, S., & Kim, S. (2021). Deep translation prior: Test-time training for photorealistic style transfer. *arXiv preprint arXiv:2112.06150*.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.
- Kumar, A., Ma, T., & Liang, P. (2020). Understanding self-training for gradual domain adaptation. In *International conference on machine learning* (pp. 5468–5479).
- Li, G., Kang, G., Liu, W., Wei, Y., & Yang, Y. (2020). Content-consistent matching for domain adaptive semantic segmentation. In *European conference on computer vision* (pp. 440–456).
- Li, Y., Yuan, L., & Vasconcelos, N. (2019). Bidirectional learning for domain adaptation of semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 6936–6945).
- Liang, J., Hu, D., & Feng, J. (2020). Do we really need

- to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International conference on machine learning* (pp. 6028–6039).
- Liu, Y., Kothari, P., van Delft, B., Bellot-Gurlet, B., Mor-dan, T., & Alahi, A. (2021). Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34.
- Luo, Y., Liu, P., Guan, T., Yu, J., & Yang, Y. (2020). Adversarial style mining for one-shot unsupervised domain adaptation. *Advances in Neural Information Processing Systems*, 33, 20612–20623.
- Marsden, R. A., Bartler, A., Döbler, M., & Yang, B. (2022). Contrastive learning and self-training for unsupervised domain adaptation in semantic segmentation. In *2022 international joint conference on neural networks (ijcnn)* (pp. 1–8).
- Marsden, R. A., Wiewel, F., Döbler, M., Yang, Y., & Yang, B. (2022). Continual unsupervised domain adaptation for semantic segmentation using a class-specific transfer. In *2022 international joint conference on neural networks (ijcnn)* (pp. 1–8).
- McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation* (Vol. 24, pp. 109–165). Elsevier.
- Mei, K., Zhu, C., Zou, J., & Zhang, S. (2020). Instance adaptive self-training for unsupervised domain adaptation. *arXiv preprint arXiv:2008.12197*.
- Mintun, E., Kirillov, A., & Xie, S. (2021). On interaction between augmentations and corruptions in natural corruption robustness. *Advances in Neural Information Processing Systems*, 34.
- Muandet, K., Balduzzi, D., & Schölkopf, B. (2013). Domain generalization via invariant feature representation. In *International conference on machine learning* (pp. 10–18).
- Mummadi, C. K., Hutmacher, R., Rambach, K., Levinkov, E., Brox, T., & Metzen, J. H. (2021). Test-time adaptation to distribution shift by confidence maximization and input transformation. *arXiv preprint arXiv:2106.14999*.
- Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (2008). *Dataset shift in machine learning*. Mit Press.
- Richter, S. R., Vineet, V., Roth, S., & Koltun, V. (2016). Playing for data: Ground truth from computer games. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *European conference on computer vision (eccv)* (Vol. 9906, pp. 102–118). Springer International Publishing.
- Ros, G., Sellart, L., Materzynska, J., Vazquez, D., & Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3234–3243).
- Sakaridis, C., Dai, D., & Van Gool, L. (2021). Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 10765–10775).
- Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., & Bethge, M. (2020). Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33, 11539–11551.
- Sun, P., Kretschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., ... others (2020). Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2446–2454).
- Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., & Hardt, M. (2020). Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning* (pp. 9229–9248).
- Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 23–30).
- Tranheden, W., Olsson, V., Pinto, J., & Svensson, L. (2021). Dacs: Domain adaptation via cross-domain mixed sampling. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (pp. 1379–1389).
- Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., ... Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 969–977).
- Tsai, Y.-H., Hung, W.-C., Schulter, S., Sohn, K., Yang, M.-H., & Chandraker, M. (2018). Learning to adapt structured output space for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7472–7481).
- Tsai, Y.-H., Sohn, K., Schulter, S., & Chandraker, M. (2019). Domain adaptation for structured output via discriminative patch representations. In *Proceedings of the IEEE international conference on computer vision* (pp. 1456–1465).
- Vergara, A., Vembu, S., Ayhan, T., Ryan, M. A., Homer, M. L., & Huerta, R. (2012). Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166, 320–329.
- Vu, T.-H., Jain, H., Bucher, M., Cord, M., & Pérez, P. (2019). Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In

Proceedings of the ieee conference on computer vision and pattern recognition (pp. 2517–2526).

the ieee international conference on computer vision (pp. 2223–2232).

- Wang, D., Shelhamer, E., Liu, S., Olshausen, B., & Darrell, T. (2020). Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*.
- Wang, Q., Fink, O., Van Gool, L., & Dai, D. (2022). Continual test-time domain adaptation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 7201–7211).
- Wu, X., Wu, Z., Lu, Y., Ju, L., & Wang, S. (2021). Style mixing and patchwise prototypical matching for one-shot unsupervised domain adaptive semantic segmentation. *arXiv preprint arXiv:2112.04665*.
- Wu, Z., Wang, X., Gonzalez, J. E., Goldstein, T., & Davis, L. S. (2019). Ace: adapting to changing environments for semantic segmentation. In *Proceedings of the ieee international conference on computer vision* (pp. 2121–2130).
- Wulfmeier, M., Bewley, A., & Posner, I. (2018). Incremental adversarial domain adaptation for continually changing environments. In *2018 ieee international conference on robotics and automation (icra)* (pp. 4489–4495).
- Yang, Y., & Soatto, S. (2020). Fda: Fourier domain adaptation for semantic segmentation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 4085–4095).
- Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., ... Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 2636–2645).
- Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.
- Zhang, M., Levine, S., & Finn, C. (2021). Memo: Test time robustness via adaptation and augmentation. *arXiv preprint arXiv:2110.09506*.
- Zhang, P., Zhang, B., Zhang, T., Chen, D., Wang, Y., & Wen, F. (2021). Prototypical pseudo label denoising and target structure learning for domain adaptive semantic segmentation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 12414–12424).
- Zhang, Q., Zhang, J., Liu, W., & Tao, D. (2019). Category anchor-guided unsupervised domain adaptation for semantic segmentation. In *Advances in neural information processing systems* (pp. 435–445).
- Zhang, Y., Deng, B., Jia, K., & Zhang, L. (2021). Gradual domain adaptation via self-training of auxiliary models. *arXiv preprint arXiv:2106.09890*.
- Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of*

Supplementary Materials

A Ablation studies for the classification benchmarks

A.1 Component analysis

Influence of individual components We begin our ablation studies by examining each component on the continual TTA benchmarks. The results are summarized in Table 5. First, we investigate the effect of solely performing self-training and the advantages of filtering pseudo-labels. Compared to the BN-1 performance, self-training alone only improves the performance on CIFAR10C and CIFAR100C. For ImageNet-C and ImageNet-R, we experience error accumulation resulting in a drastic performance degradation on ImageNet-C. Filtering pseudo-labels according to our proposed threshold benefits all continual classification benchmarks surpassing the BN-1 performance. This is expected since less confident predictions are more likely to be incorrect and can significantly contribute to error accumulation due to large gradient magnitudes (Mumtaz et al., 2021). Utilizing source data in the form of source replay stabilizes online test-time adaptation in the sense that the performance is improved on all investigated datasets. We also analyze source replay for TENT (D. Wang et al., 2020) and find that it is also beneficial for methods based on entropy minimization. TENT-continual with source replay achieves an error rate of 18.1%, 31.2%, 61.3%, and 57.6% on CIFAR10C, CIFAR100C, ImageNet-C, and ImageNet-R, respectively. This corresponds to an error reduction of -2.6%, -29.7%, -1.3%, and -1.0%. Especially for CIFAR100C, this is a significant improvement, as TENT-continual without source replay suffers from heavy error accumulation. Last but not least we look into the effect of omitting self-training and solely doing mixup or style transfer. While training on the intermediate domains provided either by mixup or style transfer significantly improves the performance upon BN-1, both methods are limited in mitigating the domain gap. Adding self-training to the framework further benefits the results. This highlights that self-training can have a big performance improvement when provided with reliable pseudo-labels.

Performing multiple update steps As already shown in Table 1, GTTA-MIX and GTTA-ST benefit from doing multiple update steps. In general, performing multiple updates does not necessarily lead to a performance improvement. Assuming that a proper learning rate for a single update step was chosen, performing, e.g., four updates does not improve the performance when only doing self-training, as presented in Table 5. This especially becomes apparent for ImageNet-C and ImageNet-R, where a drastic performance degradation is experienced. This is not surprising as in the setting of online test-time adaptation only the current test batch x_t^T is commonly utilized to perform adaptation. We denote this as over-adaptation, where as a result error accumulation can be very prominent. Naturally, filtering less confident predictions, which are more likely to be false predictions, reduces error accumulation. Source replay further helps in the setting of multiple updates to stabilize the adaptation and even results in a performance improvement on CIFAR10C and CIFAR100C compared to a single update. Finally, generating intermediate domains in the form of either mixup or style transfer is the key factor to benefit from multiple updates on CIFAR10C, CIFAR100C, and ImageNet-C. Since mixup has its limitations to create intermediate domains for natural shifts, as imposed by ImageNet-R, only style transfer benefits from multiple updates on this dataset.

A.2 Trade-off between efficiency and performance

In Table 6 (a) we further explore the effect of performing different numbers of update steps for our overall approaches GTTA-MIX and GTTA-ST, since for some applications it can make sense to neglect computational efficiency in favor of a higher accuracy and vice versa. For GTTA-MIX, CIFAR10C and CIFAR100C benefit increasingly from performing multiple update steps. They show the best performance at 8 update steps reducing the error rate to 15.0% and 28.3%, respectively. For ImageNet-C the best performance is achieved at 6 update steps reducing the error rate from 60.3% for one update to 56.4%. Since mixup shows its difficulties for natural domain shifts, as covered by ImageNet-R, performing more update steps does not necessarily result in a better performance. For GTTA-ST, the best performance for ImageNet-C is again achieved at 6 update steps, resulting in an error rate of 56.3%. In contrast to mixup, style transfer can benefit from multiple update steps for ImageNet-R, achieving the best performance of 52.3% performing 8 update steps.

Table 5: Component analysis for the continual classification benchmarks. The classification error rate (%) is reported as an average over 5 runs. We investigate the effect of self-training, filtering pseudo-labels, source-replay, and performing mixup and style transfer without self-training (on the right).

	BN-1	self-training		self-training		+ source replay		+ mixup		mixup	style transfer
self-training	n/a	✓		✓		✓		✓		✗	✗
threshold	n/a	✗		✓		✓		✓		n/a	n/a
updates dataset	-	1	4	1	4	1	4	1	4	1	1
CIFAR10C	20.4	19.9	24.8	18.4	19.4	18.1	17.6	17.2	15.6	17.7	-
CIFAR100C	35.4	33.5	40.6	32.0	33.4	30.5	29.1	30.4	28.9	31.3	-
ImageNet-C	68.6	81.9	97.7	66.6	94.1	65.2	88.9	60.3	57.1	63.1	62.2
ImageNet-R	60.4	61.2	79.6	55.7	67.5	55.4	56.0	56.4	56.6	59.7	56.9

Table 6: Classification error rate (%) for different: (a) numbers of update steps; (b) amounts of saved source samples.

(a)							(b)							
	updates dataset	1	2	4	6	8		% source dataset	100	50	25	10	5	1
GTТА-MIX	CIFAR10C	17.2	16.5	15.6	15.2	15.0	CIFAR10C	17.2	17.2	17.2	17.4	17.8	18.8	
	CIFAR100C	30.4	29.7	28.9	28.6	28.3	CIFAR100C	30.4	30.4	30.5	30.6	30.9	32.0	
	ImageNet-C	60.3	58.4	57.1	56.4	56.6	ImageNet-C	60.3	60.0	60.4	60.3	60.3	60.5	
	ImageNet-R	56.4	55.9	56.6	56.1	57.3	ImageNet-R	56.4	56.4	56.1	56.3	55.9	56.5	
GTТА-ST	ImageNet-C	59.7	58.0	57.0	56.3	57.0	ImageNet-C	59.7	59.6	59.7	59.7	59.5	59.8	
	ImageNet-R	53.8	52.9	52.5	52.9	52.3	ImageNet-R	53.8	54.1	54.0	54.1	54.5	54.0	

A.3 Amount of source samples

Since applications can vary in the amount of available memory, we show in Table 6 (b) the error rate for different percentages of saved source samples. While the performance on ImageNet-C and ImageNet-R is only marginally affected by the amount of available source samples for both GTТА-MIX and GTТА-ST, the error rate increases slightly for CIFAR10C and CIFAR100C. Still, using only 10% of the source data does not increase the error rate significantly for both CIFAR10C and CIFAR100C.

A.4 Mixup strength

In Table 7 we investigate the effect of mixup, in particular, how much source or test image is taken into account to create intermediate samples. First, it can be seen that mixup is beneficial for all datasets. ImageNet-R plays a special role in the sense that only a small mixup strength $\lambda_{\text{mix}} = 0.1$ marginally improves upon source replay, which corresponds to a mixup strength of $\lambda_{\text{mix}} = 0$. For higher mixup strengths, the performance drastically decreases. For CIFAR10C and CIFAR100C $\lambda_{\text{mix}} = 1/3$ is a good choice, corresponding to weighting source images twice as strong as test images. ImageNet-C further benefits from higher mixup strength and shows its best performance at $\lambda_{\text{mix}} = 0.5$, corresponding to an equal weighting of source and test images.

A.5 Single sample test-time adaptation

So far, we only considered the batch setting for the classification task. For some applications, timeliness can be critical, e.g., in autonomous driving, and production data arrives individually rather than in batches. Simply performing prediction and adaptation for a single sample for classification imposes some challenges. First, computing a good estimate of the batch normalization statistics is impossible and second, gradient updates are very noisy. One straightforward approach to overcome these challenges is to save recent data in a buffer and use a sliding window for performing prediction and

Table 7: Classification error rate (%) averaged over 5 runs for different mixup strengths λ_{mix} . While $\lambda_{\text{mix}} = 0$ corresponds to source replay, $\lambda_{\text{mix}} = 0.5$ corresponds to the case of equally weighting source and test images.

λ_{mix} dataset	0	1/10	1/4	1/3	2/5	1/2
CIFAR10C	18.1	18.1	17.7	17.2	17.4	17.7
CIFAR100C	30.5	30.4	30.4	30.4	30.5	30.6
ImageNet-C	65.2	64.0	61.5	60.3	59.9	59.6
ImageNet-R	55.4	55.2	55.8	56.4	57.1	58.0

adaptation. Specifically, we save the last b test samples in a buffer and update the model every b samples due to the strong correlation introduced by the buffer. For current test sample x_{ti}^T at time step t , the sample is first added to the buffer replacing the oldest sample. Now, the whole buffer is forwarded to make a prediction for the current sample x_{ti}^T . Needless to say, this comes with a computational overhead, but allows a good estimate of the batch normalization statistics and better gradient updates.

Table 8 illustrates the results for the previously described setting on the continual corruption benchmarks using a buffer of size 16. Due to the much smaller batch size used in this setting, the performance of the baseline BN-1 slightly degrades, since the estimation of the batch statistics becomes more noisy. While the performance of our approach is also slightly worse compared to the results achieved in the batch setting of TTA, GTTA in the single-sample setting still performs on par with most state-of-the-art methods in the batch setting. Using a larger buffer size of 32 further increases the performance. On ImageNet-C the performance in the single sample setting is now comparable with the batch setting, being only 0.1% and 0.3% behind for GTTA-MIX and GTTA-ST, respectively.

Table 8: Classification error rate (%) for batch size 1 using a sliding window approach with b samples. The weight update is performed once after the complete buffer has changed.

Time				$t \longrightarrow$																
	Method	batch size	buffer size	<i>Gaussian</i>	<i>shot</i>	<i>impulse</i>	<i>defocus</i>	<i>glass</i>	<i>motion</i>	<i>zoom</i>	<i>snow</i>	<i>frost</i>	<i>fog</i>	<i>brightness</i>	<i>contrast</i>	<i>elastic-trans</i>	<i>pixelate</i>	<i>jpeg</i>	Mean	
CIFAR10C	BN-0 (src.)	1	-	72.3	65.7	72.9	46.9	54.3	34.8	42.0	25.1	41.3	26.0	9.3	46.7	26.6	58.5	30.3	43.5	
	BN-1	1	16	30.8	28.8	39.2	15.7	37.5	16.8	15.0	20.3	20.2	17.8	10.7	15.5	27.1	23.0	29.9	23.2	
	BN-1	1	32	29.3	27.3	37.7	14.2	36.5	15.4	13.8	19.0	18.8	16.8	9.4	14.3	25.2	21.4	28.8	21.9	
	GTTA-MIX	1	16	26.5	21.5	28.9	12.7	30.5	13.9	12.9	16.8	15.0	13.1	9.2	10.6	23.0	17.5	22.9	18.3	
	GTTA-MIX	1	32	25.5	21.4	28.2	11.9	30.1	13.2	11.9	15.8	14.8	12.8	8.6	10.6	21.4	16.5	23.0	17.7	
	GTTA-MIX	200	-	26.0	21.5	29.7	11.1	30.0	12.2	10.5	15.1	14.1	12.3	7.5	10.0	20.4	15.8	21.4	17.2	
CIFAR100C	BN-0 (src.)	1	-	73.0	68.0	39.4	29.3	54.1	30.8	28.8	39.5	45.8	50.3	29.5	55.1	37.2	74.7	41.2	46.4	
	BN-1	1	16	46.2	44.5	47.1	31.4	46.5	33.3	31.7	39.2	38.6	45.4	30.8	34.7	40.6	37.5	45.2	39.5	
	BN-1	1	32	44.1	42.4	45.2	29.6	43.6	31.0	30.0	37.1	36.7	43.8	28.7	32.6	38.0	35.3	42.9	37.4	
	GTTA-MIX	1	16	40.1	36.1	37.8	28.3	40.4	29.8	28.3	32.7	31.1	34.2	26.2	27.5	33.5	30.4	38.2	33.0	
	GTTA-MIX	1	32	38.9	35.0	36.8	26.6	38.3	28.0	26.5	31.2	30.1	34.0	24.4	26.2	32.1	28.6	36.4	31.5	
	GTTA-MIX	200	-	39.4	34.4	36.6	24.7	36.8	26.6	24.3	30.1	28.9	34.6	22.8	25.1	30.7	26.9	34.7	30.4	
ImageNet-C	BN-0 (src.)	1	-	95.3	94.6	95.3	84.9	91.1	86.9	77.2	84.4	79.7	77.3	44.4	95.6	85.2	76.9	66.7	82.4	
	BN-1	1	16	86.6	85.2	86.0	86.6	86.5	75.7	65.1	67.8	70.5	55.2	37.8	84.6	59.1	55.0	63.7	71.0	
	BN-1	1	32	85.4	84.0	84.9	85.6	85.0	74.1	61.9	66.8	68.4	52.6	36.1	83.9	57.1	52.3	61.7	69.3	
	GTTA-MIX	1	16	80.8	73.8	71.6	79.3	79.3	70.0	61.1	58.1	59.8	47.6	37.5	73.8	54.2	51.1	53.3	63.4	
	GTTA-MIX	1	32	79.8	73.4	70.7	77.3	76.0	64.6	55.8	57.4	58.5	45.5	34.4	67.1	50.5	45.7	49.4	60.4	
	GTTA-MIX	64	-	80.5	74.7	72.4	77.8	75.7	64.3	54.0	57.0	58.6	44.6	33.9	67.5	49.4	44.7	49.3	60.3	
	GTTA-ST	1	16	80.9	74.3	74.7	77.7	77.3	66.3	58.2	59.0	60.4	47.6	37.6	62.9	53.5	48.8	52.6	62.1	
	GTTA-ST	1	32	80.5	74.4	74.0	77.3	74.8	63.5	54.3	56.4	57.6	45.2	33.5	61.8	50.0	45.7	50.7	60.0	
	GTTA-ST	64	-	80.6	74.1	74.3	76.8	74.9	62.3	53.9	56.4	58.0	44.1	33.4	62.2	48.6	44.9	50.4	59.7	

B Ablation studies for CarlaTTA

B.1 Gradual shifts for CarlaTTA

Denoting Δt as a proxy for the gradual shift, we generate a *dynamic* and a *day2night* sequence which allow to further study the benefits of gradual TTA. By sub-sampling the sequences, we achieve varying degrees of gradual shift. For comparison, we evaluate on the least common multiple. As shown in Table 9 we further benefit from a slower gradual domain shift ($\Delta t/2$), gaining another 0.5% on *day2night* and 1.4% on *dynamic*. Having a faster domain shift corresponding to a bigger delta ($2\Delta t$, $4\Delta t$) leads to a reduced performance, indicating that exploiting small gradual shifts is indeed beneficial for our self-training setup. Considering the *dynamic* sequence, for very small shifts ($\Delta t/4$) we can see a slight performance degradation in comparison to $\Delta t/2$, however, the mIoU is still 0.7% better than Δt .

B.2 Investigating abrupt shifts and no shifts for CarlaTTA

Since we do not have gradual shifts all the time in the real world, we investigate two additional settings: (a) no domain shift occurs, i.e., the test domain is equal to the source domain and (b) the domain changes abruptly. The results are reported in Table 10 (a) and (b). For the setting of (a), where no domain shift occurs, BN-0, as expected, shows the best performance with a mIoU of 78.4%. Updating the batch statistics through an exponential moving average, i.e., considering BN-EMA, leads to a performance decrease of 0.8%. As a result, also GTТА-ST's performance cannot achieve BN-0. BN-1 performs even worse at a mIoU of 76.6% illustrating that even in the setting of segmentation in an urban scene a perfect estimation of the batch normalization statistics is not possible for a single sample. For the investigation how GTТА-ST handles abrupt shifts, instead of starting at the first sample of the *day2night* sequence, we begin after 300 samples, corresponding to a sun

Table 9: Investigation of the amount of gradual shift. Δt corresponds to the delta time used for all our standard sequences. We report the mIoU (%) for the least common multiple, averaged over 5 runs. For comparison, BN-0 and BN-1 achieve 58.3 and 61.7 on the *day2night* split and 39.4 and 49.1 on the *dynamic* split, respectively.

	$4\Delta t$	$2\Delta t$	Δt	$\Delta t/2$	$\Delta t/4$
day2night	64.6	66.1	66.7	67.2	67.7
dynamic	53.0	55.2	55.6	57.0	56.3

Table 10: Mean intersection-over-union averaged over 5 runs for a) when no domain shift occurs and the test domain is equal to the source domain; b) the domain changes abruptly from day to night.

(a)						(b)					
Method	BN-0	BN-0.1	BN-1	BN-EMA	GTТА-ST	Method	BN-0	BN-0.1	BN-1	BN-EMA	GTТА-ST
	78.4	78.4	76.6	77.6	77.9±0.14		43.8	52.3	52.8	54.3	59.7±0.32

altitude of 0° . After one complete night-cycle, the sequence reaches its end. Our method still shows the capability to adapt, reaching a 5.4% higher mIoU than the best BN adaptation approach, namely BN-EMA.

B.3 Performance over time on CarlaTTA

In Figure 5 we illustrate the sequences *day2night*, *clear2fog*, *clear2rain*, and *highway* and their corresponding progression of environment conditions. At time step t we visualize the mean intersection-over-union averaged from the beginning $t = 0$ up to time step t . For the *day2night* sequence a decline in performance can be seen at around sample 300, corresponding to the sun setting. In the setting *clear2fog* the increase in cloudiness does not influence the model’s performance much. With the fog density increasing, a steady decrease in performance can be seen. For *clear2rain* the performance of GTТА-ST stays more or less constant, while the source performance steadily decreases with the start of the rain. The *highway* sequence experiences the same weather behaviour as the *dynamic* sequence, but additionally experiences a shift of urban scenery to a highway scene. As a result, the performance decreases drastically in the beginning. At around sample 550 an increase in performance can be experienced. This is due to the car driving back into the city, leaving the city again at sample 770.

C Dataset: CarlaTTA

In the following, we will discuss the specifics about our dataset CarlaTTA. For the generation, CARLA 0.9.13 was used. The data is recorded using an RGB camera and a corresponding semantic segmentation sensor with a resolution of 1920×1024 and a field of view of 40° . Both sensors are positioned 0.5 m forward and 1.2 m upward relative to the ego-vehicle. 14 classes are considered. We visualize the class priors for all sequences in Figure 8.

Table 11: *clear* weather parameters and maximum values used for rain and fog in the settings *clear2rain* and *clear2fog*, respectively.

	sun				rain			fog		
	altitude	azimuth	cloudiness	wind	precipitation	deposits	wetness	density	distance	falloff
unit	°	°	%	%	%	%	%	%	m	-
clear	90	0	10	5	0	0	0	0	0	0
night (max)	-90	180	-	-	-	-	-	-	-	-
rain (max)	-	-	55	90	80	85	100	-	-	-
fog (max)	-	-	55	90	-	-	10	50	0	0.9

clear The source dataset *clear* is the basis for the four domain changes: *day2night*, *clear2fog*, *clear2rain*, and *dynamic*. The data for *clear* is recorded in Town10HD due to being the only town in CARLA with high resolution textures. To increase diversity we generate multiple sequences using different seeds. Specifically, for each seed, up to 40 vehicles and 20 pedestrians are randomly sampled from all (safe) blueprints. We end up with 3500 train and 500 test samples. *clear* is recorded at noon (sun altitude of 90°) with a cloudiness of 10% and a wind intensity of 5%. All weather parameters are fixed in this setting. The complete overview of the weather parameters used for the *clear* setting is given in Table 11.

day2night, clear2fog, clear2rain, dynamic Different domain changes are introduced by the sequences *day2night*, *clear2fog*, *clear2rain*, and *dynamic*. Each sequence starts with the weather parameters of *clear* and follows the behavior illustrated in Figure 6. The weather model is based on the implementation of CARLA*. The maximum values used for the night, rain, and fog setting are depicted in Table 11, with a bar indicating no change from the default parameters of *clear*. While the default sequence length is 1200 samples, *dynamic* contains 6000 samples to have the capability to study long-term behavior. Figure 7 shows every 100th sample of the mentioned domain shifts.

day2night-slow and dynamic-slow *day2night-slow* and *dynamic-slow* are only considered for the gradual domain shift ablation study. Compared to the regular sequences, where 1200 samples correspond to one complete day-night cycle, in the slow setting, 4800 samples correspond to one day-night cycle. This enables the investigation of smaller domain shifts.

highway Since Town10HD does not contain any other settings than urban scenery, we use Town04 for the *highway* sequence. For the corresponding source dataset, we use sequences generated from Town02 (urban setting, similar to Town04) and sequences from Town04 where the ego vehicles only drives in the city. The source dataset contains overall 3000 train and 500 test samples. The *highway* sequence starts in the city of Town04 and shortly after continues on the highway. It follows the same weather behavior as *dynamic*. Examples of the sub-sampled sequence are visualized in the last column in Figure 7.

*https://github.com/carla-simulator/carla/blob/0.9.13/PythonAPI/examples/dynamic_weather.py

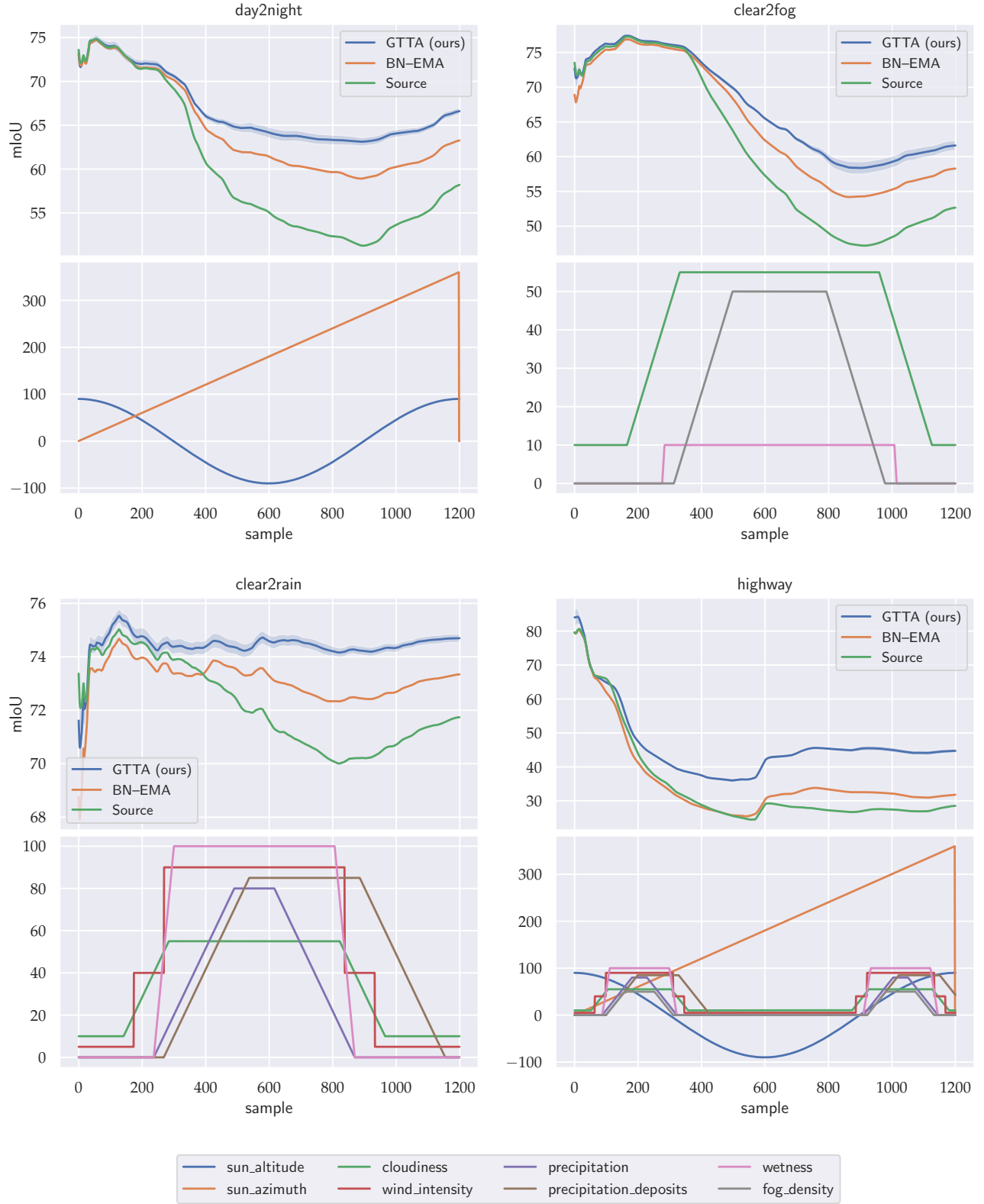


Figure 5: mIoU up to time step t for *day2night*, *clear2fog*, *clear2rain*, and *highway*. Additionally, we illustrate the progression of the weather parameters to have a direct comparison when a domain shift occurs.

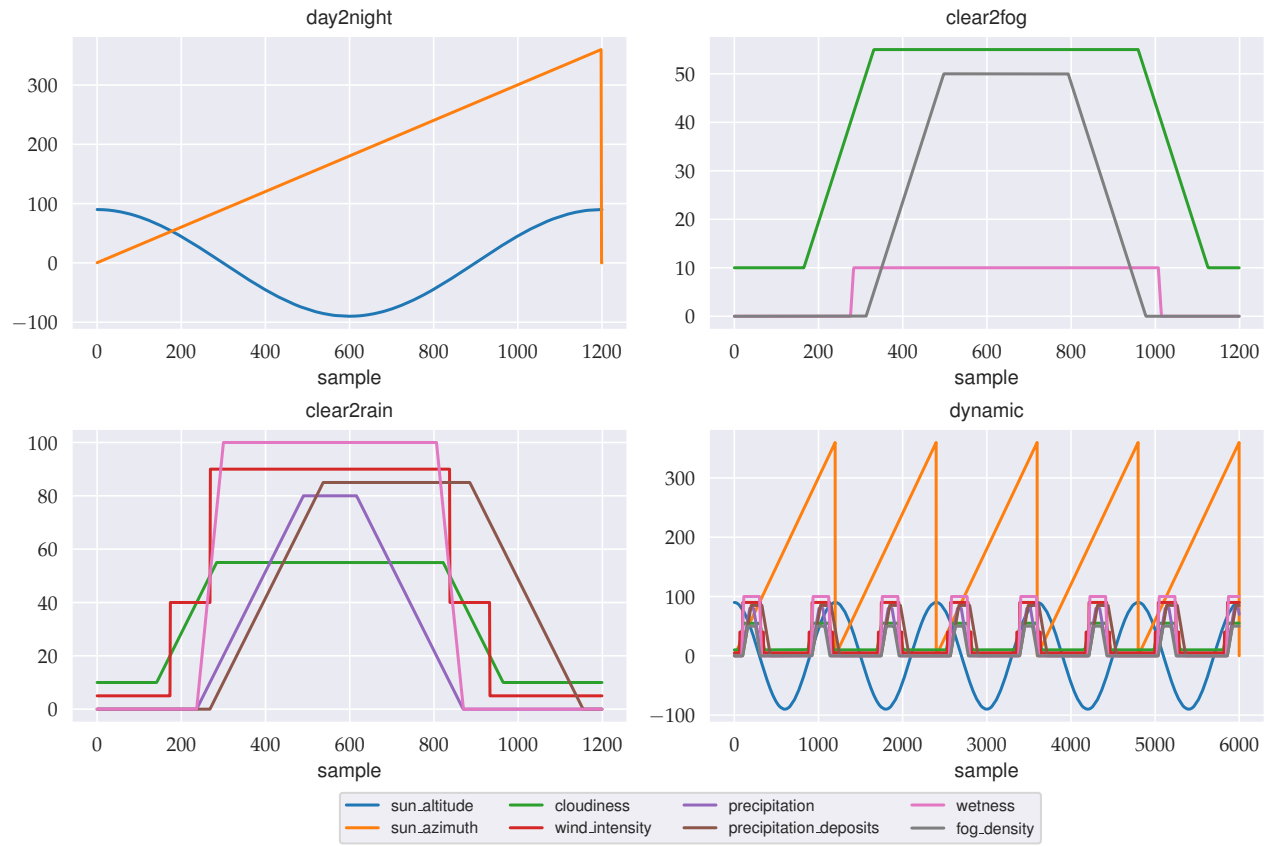


Figure 6: Progression of the weather parameters for the sequences *day2night*, *clear2fog*, *clear2rain*, and *dynamic*. Note that *highway* follows the weather behavior of *dynamic*.

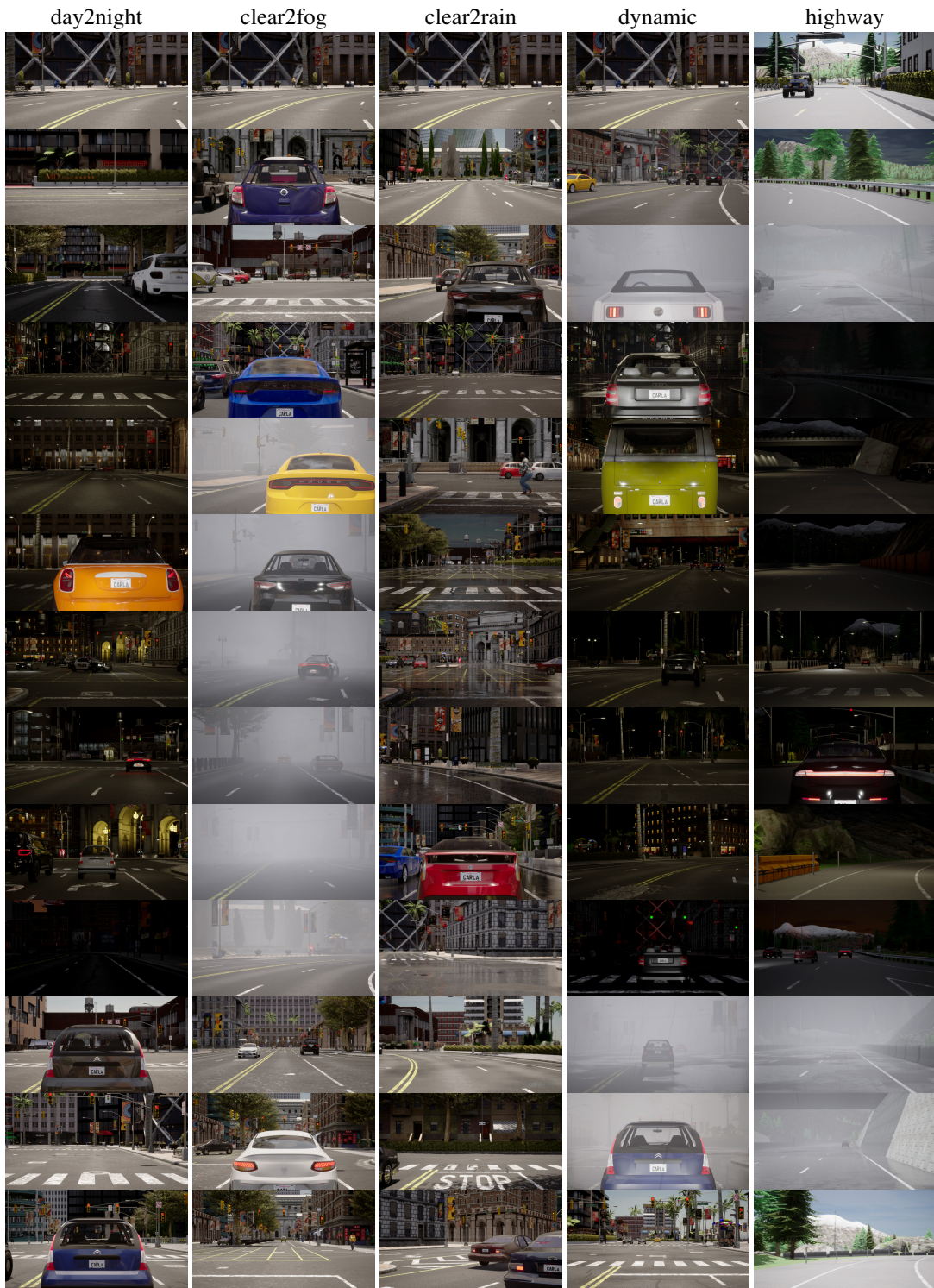


Figure 7: Sequence for the different domain changes. Every 100th example is shown.

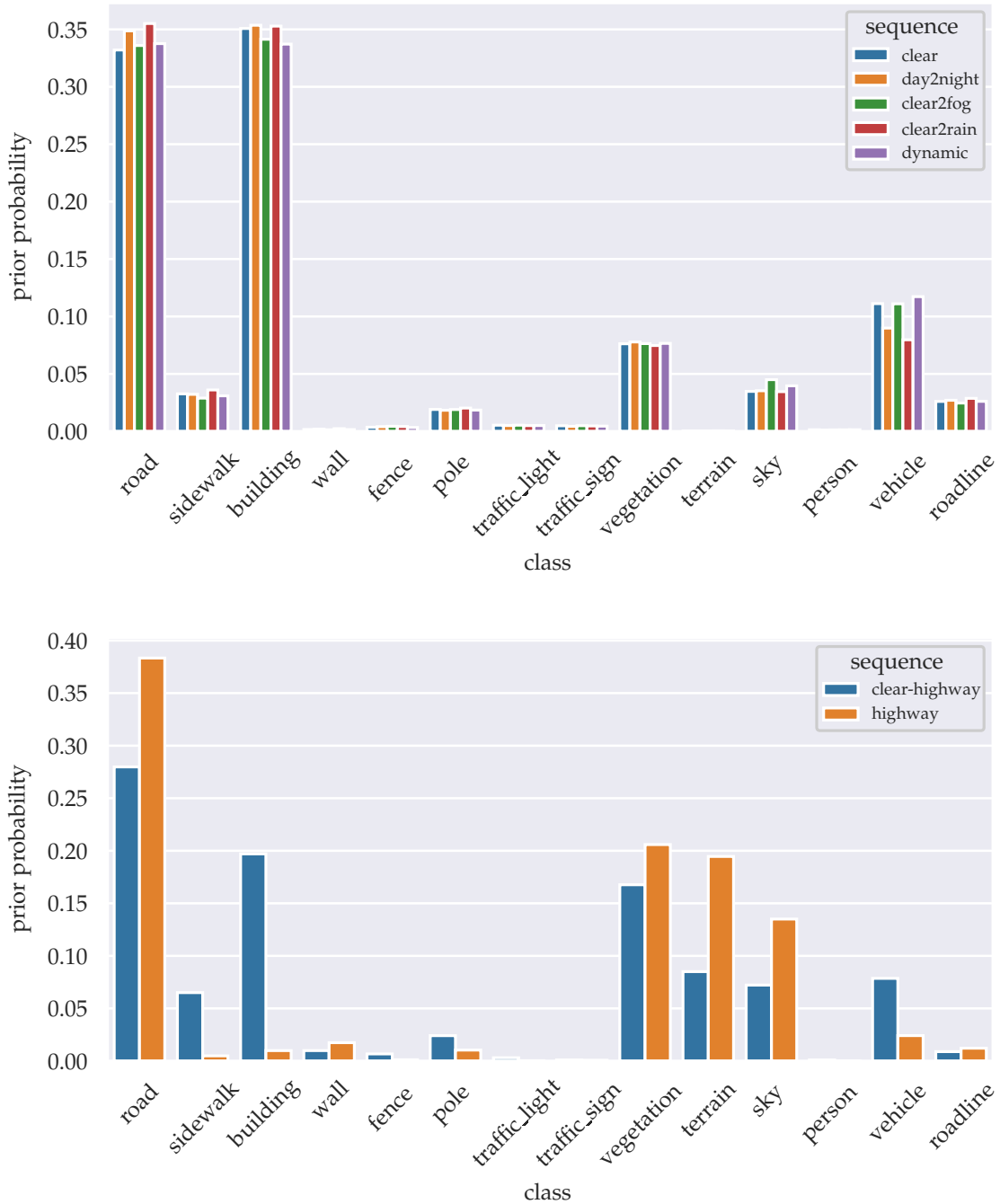


Figure 8: Prior probabilities for the 14 classes. On the top we visualize the prior probabilities corresponding to data originating from Town10HD. On the bottom, we compare *clear-highway* to *highway*. The label distribution shift is clearly prominent as the test-time sequence evolves from an urban scenery into a highway setting.