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

Robert A. Marsden*, Mario Döbler*, and Bin Yang

Institute of Signal Processing and System Theory, University of Stuttgart
Stuttgart, Germany
Email: {robert.marsden, mario.doebler, bin.yang}@iss.uni-stuttgart.de

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 after 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 benchmark: CarlaTTA. It enables the exploration of several non-stationary domain shifts. 1

Index Terms—Test-time Adaptation, Semantic Segmentation

I. 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 violated. This effect is known as data shift [1] and can lead to a large drop in performance on the test data. While it is possible to improve robustness and generalization directly during training [2]–[5], the effectiveness remains limited due to the wide range of potential data shifts [6] 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 after 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. 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. [7] 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 [7].

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 [8]. On the contrary, [8] 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, [8] mentions, among others, evolving road conditions [9] and sensor aging [10]. 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 during deployment to a sequence of test domains as continual test-time adaptation.
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 further 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 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 an 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 non-stationary domain shifts during test-time in the field of autonomous driving.

II. RELATED WORK

a) 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 [11]–[13], the feature space [14], [15], the output space [16], [17], or several spaces in parallel [18], [19]. One line of work relies on adversarial learning, where a domain classifier tries to discriminate whether some feature maps [20] or network outputs [16], [21] belong to the source or target domain. It is also possible to exploit adversarial learning or adaptive instance normalization (AdaIN) [22] for transferring the target style to source images [11], [12]. Lately, self-training has gained a lot of attraction [23]–[27]. 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.

b) One-shot Unsupervised Domain Adaptation: As pointed out in [28], even collecting unlabeled target data can be challenging. Therefore, [28] introduced one-shot UDA, where only one single target image is available during the model adaptation. To address this problem, [28] extends the adaptive instance normalization framework of [22] with a variational autoencoder. By selecting styles for which the segmentation model is uncertain, the domain gap is mitigated. In [29], patch-wise prototypical matching is combined with a style mixing component within the segmentation model.

c) Test-time Adaptation: While generalizing to any test distribution would solve many problems, the lack of information about the test environment during training is a major 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 [30] 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, [31] update the BN layers by entropy minimization. [32] create an ensemble prediction through test-time augmentation [33] 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 [34]–[36] or apply contrastive learning [37]. Recent works make use of diversity regularizers [38], [39] to prevent the collapse to trivial solutions potentially caused by confidence maximization.

d) Continual Test-time Adaptation: In continual TTA, the model is adapted to a sequence of different target domains. While some of the existing methods can be applied to the continual setting, such as the online version of TENT [31], they are often prone to error accumulation due to miscalibrated predictions [7]. CoTTA [7] uses weight and augmentation-averaged predictions to reduce error accumulation and stochastic restore to circumvent catastrophic forgetting [40].

e) 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 [8]. Therefore, numerous methods consider the setting of gradual domain adaptation [41]–[43], where several intermediate domains exist between source and target. While some of the proposed approaches successively adapt the model using adversarial learning [9], [44], self-training has been shown to be well suited in this setting [8].

III. METHODOLOGY

Since in many practical applications environmental conditions can change over time, a model pre-trained on source data \((X^S, Y^S)\) 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 [8], 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
Based on adversarial learning [47], [48], which can be unstable during training. Therefore, we follow [22] 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 \( \mu \) and standard deviation \( \sigma \) as a style feature map \( z_i^T \) extracted from the \( i \)-th test image \( x_i^T \).

\[
\tilde{z}_j = \sigma(\tilde{z}_i^T) \frac{z_j^S - \mu(z_j^S)}{\sigma(z_j^S)} + \mu(z_i^T).
\]  

(2)

Since \( z_j^S \) and \( z_i^T \) 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:

\[
\mathcal{L} = \lambda_s \sum_{i=1}^{L} \left[ \text{MSE}(\mu(E_1(\tilde{x}_j)), \mu(E_1(x_i^T))) + \text{MSE}(\sigma(E_1(\tilde{x}_j)), \sigma(E_1(x_i^T))) \right] + \text{MSE}(E(\tilde{x}_j), \tilde{z}_j)
\]

(3)

where MSE is the mean-squared-error, \( E_1() \) represents the output of the \( l \)-th layer of the encoder, and \( \lambda_s \) 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 [12] 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 perform style replay.

B. 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 \( \hat{y}_{ti}^S = \text{argmax}(p_{ti}^T) \). These pseudo-labels are subsequently used to minimize the target cross-entropy

\[
\mathcal{L}_E(\hat{y}_{ti}^S, p_{ti}^T) = -\frac{1}{N_t} \sum_{i=1}^{N_t} \sum_{c=1}^{C} \hat{y}_{ti}^S \log(p_{ti,c}).
\]

(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 the cross-entropy loss has large gradient magnitudes for uncertain predictions [39]. 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.

A. Generating intermediate domains

To generate intermediate domains, we propose two ideas: mixup known for improving robustness [45] 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.

a) 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:

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

(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 \( x_T \) that has the highest similarity in terms of the largest dot product in the output softmax probability space.

b) Style transfer: Another possibility to create intermediate domains is to use style transfer. However, performing style transfer during test-time imposes some challenges. It should be of light weight to enable real-time processing and the network should be easily trainable during test-time, even when only having one test sample at a time. While [46] 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
dom. 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 $\alpha_{th} = 0.1$ that leverages the current softmax probabilities as follows:

$$\gamma_t = (1 - \alpha_{th}) \gamma_{t-1} + \alpha_{th} \frac{1}{N_t} \sum_{i=1}^{N_t} \max(e^{p_{t,i}^T}).$$

(5)

IV. 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 [49], ACDC [50], BDD100K [51], SYNTHIA [52], and GTA5 [53], 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 (Fig. 3). It is based on CARLA 0.9.13 [54], an open-source simulator for autonomous driving research. The data is recorded using an RGB camera and a corresponding semantic segmentation sensor with a resolution of 1920 x 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.

a) 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.

b) 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 weather model based on the implementation of CARLA1. 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. While the default sequence length is 1200 samples, dynamic contains 6000 samples and thus five day-night cycles, allowing to study long-term behavior.

c) 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 gradual shifts.

d) 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. 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.

V. EXPERIMENTS

a) Baselines: Since BN has proven to be very effective during test-time [30], we consider several variations that can be derived from the following equations:

$$\mu_{tm} = (1 - \alpha)\hat{\mu}_m^S + \alpha \mu_{tm}^T$$

$$\sigma_{tm} = (1 - \alpha)\hat{\sigma}_m^S + \alpha \sigma_{tm}^T.$$  

(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 $(t - 1)$:

$$\bar{\mu}_{tm} = (1 - \alpha)\bar{\mu}_{(t-1)m} + \alpha \mu_{tm}^T$$

$$\bar{\sigma}_{tm} = (1 - \alpha)\bar{\sigma}_{(t-1)m} + \alpha \sigma_{tm}^T.$$  

(7)

To further evaluate our method, we compare to several approaches from related fields: TENT [31] uses BN-1 in combination with an entropy minimization strategy with respect to the BN parameters. CoTTA [7] utilizes BN-1 and a mean teacher with test-time augmentation to perform entropy minimization. To prevent catastrophic forgetting, stochastic source restore is leveraged. AdaContrast [37] uses pseudo-label refinement for self-training and contrastive learning.

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

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
For segmentation, we additionally consider MEMO [32], which combines test-time augmentation and entropy minimization and two methods from one-shot UDA. While ASM [28] uses an AdaIN based style transfer model to explore the style space, SM-PPM [29] integrates style mixing into the segmentation network and combines it with patch-wise prototypical matching.

A. Experiments on CarlaTTA

   a) Implementation details: Following the standard framework in UDA for semantic segmentation [16], we use the DeepLab-V2 [55] architecture with a ResNet-101 backbone. The model is trained with SGD using a constant learning rate of $2.5 \times 10^{-4}$, momentum of 0.9, and weight decay of $5 \times 10^{-4}$. All methods use the same pre-trained source 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 $1024 \times 512$. The style transfer network consists of the same VGG19 based encoder-decoder architecture as used in [12] and is pre-trained for 20k iterations on the source domain using Adam with learning rate $1 \times 10^{-4}$. During test-time adaptation, we use batches consisting of two source samples and two random 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. We report the mean intersection-over-union (mIoU) over the entire test sequence.

1) Results for CarlaTTA: Our results are summarized in Table I. 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 [29], 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. While SM-PPM performs better than TENT or ASM, it is still slightly worse on average compared to BN-EMA. CoTTA does also

| Method          | source-free | day2night | clear2fog | clear2rain | 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-continual  | ✓           | 61.5      | 56.0      | 70.9       | 50.3    | 32.0    |
| TENT-episodic   | ✓           | 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    |
| GTTA-ST (ours)  | ✗           | 66.7      | 61.6      | 74.7       | 60.3    | 44.8    |

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

Fig. 4. mIoU up to time step $t$ for the dynamic sequence.
not perform better than BN–1 and the results significantly drop 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, which employs style transfer (GTTA-ST) 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 dynamic, day2night, and highway as through the intermediate domain introduced by style transfer, self-training benefits from more reliable pseudo-labels.

In Figure 4, we show the mIoU up to time step $t$ for the dynamic sequence. While the performance of the source model strongly degrades due to the domain shift, GTTA-ST maintains good performance throughout the whole test sequence.

2) Ablation studies for CarlaTTA:

a) Gradual shifts: Denoting $\Delta t$ 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 II, 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 $\Delta t$.

b) Investigating abrupt shifts and no shifts: Since we do not have gradual shifts all the time in the real world, we investigate two additional settings: no domain shift occurs, i.e., the test domain is equal to the source domain and the domain changes abruptly. The results are reported in Table III. If 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 GTTA-ST’s performance cannot achieve BN–0. BN–1 performs even worse with a mIoU of 76.6%, demonstrating that even in the setting of segmentation of urban scenes, a perfect estimation of the batch normalization statistics is not possible for a single sample. For the investigation how GTTA-ST handles abrupt shifts, instead of starting at the first sample of the day2night sequence, we begin after 300 samples, corresponding to a sun 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. Adapting to shifts caused by corruptions

a) Corruption benchmarks: CIFAR10-C, CIFAR100-C, and ImageNet-C were originally published to evaluate robustness of neural networks [56]. The benchmark comprises of 15 corruptions with 5 severity levels, which are applied to the validation and test images of ImageNet [57] and CIFAR [33], respectively. In accordance with the RobustBench benchmark [58], a pre-trained WideResNet-28 is used for CIFAR10-to-CIFAR10-C, ResNeXt-29 for CIFAR100-to-CIFAR100-C, and ResNet-50 for ImageNet-to-ImageNet-C. Following the implementation and hyperparameters of [7], a batch size of 200 is utilized for CIFAR and a batch size of 64 for ImageNet. We use Adam [59] 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 CIFAR10-C and CIFAR100-C. A mixup strength of $\lambda_{mix} = \frac{1}{2}$ is used for all experiments. For ImageNet-C, we additionally compare to the style transfer variant GTTA-ST. The style transfer network is the same as before, omitting the class-conditional AdaIN layers.

b) Continual corruption benchmarks: We first consider the continual TTA setting, as proposed in [7]. 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 a domain shift. The adaptation is performed under the highest corruption severity level 5.

The results are reported in Table IV. Simply evaluating the pre-trained source model yields an average error of 43.5% for CIFAR10-C, 46.4% for CIFAR100-C, 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 [7], TENT-continual outperforms BN–1 in early stages, but quickly deteriorates...
TABLE IV
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.

| Method          | BN–0 (src.) | BN–1 | TENT-cont. | AdaContrast | CoTTA | GTTA-MIX | GTTA-ST | GTTA-ST | Mean |
|-----------------|-------------|------|------------|-------------|-------|----------|---------|---------|------|
| Time updates    |            |      |            |             |       |          |         |         |      |
| Time           | Gaussian    | shut | impulse    | de-focus    | glass | motion   | rain    | cloud   | fog  |
| Source-free    | 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 |
| Time           | BN–1        | 28.1 | 6.6       | 13.6        | 12.8  | 35.3     | 14.2    | 12.1    | 17.3 | 17.4 | 15.3 | 8.4  | 12.6 | 23.8 | 19.7 | 23.7 | 20.4 |
| Source-free    | 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 |
| Time           | AdaContrast | 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 |
| Source-free    | 24.3        | 21.3 | 26.6       | 11.6        | 27.6  | 7.2      | 12.2    | 10.3    | 14.8 | 14.1 | 12.4 | 7.5  | 10.6 | 18.3 | 13.4 | 17.3 | 16.2 |
| Time           | CoTTA       | 26.0 | 21.3       | 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 |
| Source-free    | 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 |

TABLE V
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.

| Method          | BN–0 (src.) | BN–1 | TENT-cont. | AdaContrast | CoTTA | GTTA-MIX | GTTA-ST | GTTA-ST | Mean |
|-----------------|-------------|------|------------|-------------|-------|----------|---------|---------|------|
| Time updates    |            |      |            |             |       |          |         |         |      |
| Method          |            |      |            |             |       | Source-free |        |         |      |
| Time           | Gaussian    | shut | impulse    | de-focus    | glass | motion   | rain    | cloud   | fog  |
| Source-free    | 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 |
| Time           | 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 |
| Source-free    | 41.2        | 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 | 90.4 | 60.9 |
| Time           | AdaContrast | 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 |
| Source-free    | 40.1        | 37.7 | 39.7       | 26.9        | 38.0  | 27.9     | 26.4    | 32.8    | 31.8 | 40.3 | 24.7 | 24.7 | 26.9 | 23.5 | 28.3 | 33.5 |
| Time           | CoTTA       | 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 |
| Source-free    | 36.4        | 32.1 | 34.0       | 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 |

after a few corruptions. This becomes particularly evident for CIFAR10-C, 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 is source replay. This has the restriction of requiring access to source data, but stabilizes self-training, e.g., TENT achieves an error rate of 31.2% on CIFAR10-C. CoTTA shows its strong suits for CIFAR10-C outperforming BN–1 by 4.2% while performing comparably to TENT on ImageNet-C. AdaContrast outperforms BN–1, but lacks behind CoTTA on all datasets. Our method GTTA successfully shows on all datasets that generating intermediate domains by mixup or style transfer allows a better adaptation via self-training. Performing a single update per test batch leads to state-of-the-art results on CIFAR100-C and ImageNet-C. Performing four updates results in a further improvement on all datasets and also sets state-of-the-art results on CIFAR10-C. Note that utilizing more update steps for source-free methods like CoTTA does not improve the performance, on the contrary.

C) 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 incre-
Adapting to real-world distribution shifts

a) ImageNet-R: To investigate the performance in the presence of distribution shifts not caused by corruptions, we also analyze ImageNet-R [3] 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 VI. While GTTA-MIX again outperforms previous methods on this benchmark, GTTA-ST shows a tremendous improvement.

b) Comparing GTTA-MIX and GTTA-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 GTTA-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 [22]. Even though GTTA-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.

D. Ablation studies for classification benchmarks

1) Trade-off between efficiency and performance: In Table VII, 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, CIFAR10-C and CIFAR100-C 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.

2) Amount of source samples: Since applications can vary in the amount of available memory, we also show in Table VII 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 GTTA-MIX and GTTA-ST, the error rate increases slightly for CIFAR10-C and CIFAR100-C. Still, using only 10% of the source data does not increase the error rate significantly for both CIFAR10-C and CIFAR100-C.

3) Mixup strength: In Table VII, we also 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 CIFAR10-C and CIFAR100-C $\lambda_{\text{mix}} = 1/3$ 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.

E. Single sample test-time adaptation

So far, we only considered the batch setting for the classification task. Therefore, we now also analyze the situation where only one test sample is available at a time, which imposes some challenges. First, computing a good estimate of the batch normalization statistics is impossible and second, gradient updates become very noisy. One straightforward approach to overcome these challenges is to save recent test data in a buffer and use a sliding window for performing prediction and adaptation. Specifically, we save the last $b$ test samples in a buffer and only update the model every $b$ samples due to the strong correlation introduced by the buffer. For current test sample $x^t_{ti}$ 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 $x_i$. While this comes with a computational overhead, it allows a good estimate of the batch normalization statistics and better gradient updates.

Table VIII illustrates the results on the continual corruption benchmarks using a buffer of size 32. Due to the smaller batch size used in this setting, the performance of the baseline BN–1 slightly degrades. 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 comparable or better than most state-of-the-art methods in the batch setting.

VI. 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.

REFERENCES

[1] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. Mit Press, 2008.

[2] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, “Augmix: A simple data processing method to improve robustness and uncertainty,” *arXiv preprint arXiv:1912.02781*, 2019.

[3] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo *et al.*, “The many faces of robustness: A critical analysis of out-of-distribution generalization,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8340–8349.

[4] K. Muandet, D. Balduzzi, and B. Schölkopf, “Domain generalization via invariant feature representation,” in *International Conference on Machine Learning*. PMLR, 2013, pp. 10–18.

[5] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boschoon, and S. Birchfield, “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*, 2018, pp. 969–977.

[6] E. Mintun, A. Kirillov, and S. Xie, “On interaction between augmentations and corruptions in natural corruption robustness,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[7] Q. Wang, O. Fink, L. Van Gool, and D. Dai, “Continual test-time domain adaptation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7201–7211.

[8] A. Kumar, T. Ma, and P. Liang, “Understanding self-training for gradual domain adaptation,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 5468–5479.

[9] A. Bobu, E. Tzeng, J. Hoffman, and T. Darrell, “Adapting to continuously shifting domains,” 2018.

[10] A. Vergara, S. Vembu, T. Ayhan, M. A. Ryan, M. L. Homer, and R. Huerta, “Chemical gas sensor drift compensation using classifier ensembles,” *Sensors and Actuators B: Chemical*, vol. 166, pp. 320–329, 2012.

[11] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, A. Efros, and T. Darrell, “Cycada: Cycle-consistent adversarial domain adaptation,” in *International conference on machine learning*. PMLR, 2018, pp. 1989–1998.

[12] R. A. Marsden, F. Wiewel, M. Döbler, Y. Yang, and B. Yang, “Continual unsupervised domain adaptation for semantic segmentation using a class-specific transfer,” in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.

[13] Z. Wu, X. Wang, J. E. Gonzalez, T. Goldstein, and L. S. Davis, “Ace: adapting to changing environments for semantic segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2121–2130.

[14] Q. Zhang, J. Zhang, W. Liu, and D. Tao, “Category anchor-guided unsupervised domain adaptation for semantic segmentation,” in *Advances in Neural Information Processing Systems*, 2019, pp. 435–445.

[15] R. A. Marsden, A. Bartzler, M. Döbler, and B. Yang, “Contrastive learning and self-training for unsupervised domain adaptation in semantic segmentation,” in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.

[16] H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker, “Learning to adapt structured output space for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7472–7481.

[17] Y.-H. Tsai, K. Sohn, S. Schulter, and M. Chandraker, “Domain adaptation for structured output via discriminative patch representations,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1456–1465.

[18] Y. Li, L. Yuan, and N. Vasconcelos, “Bidirectional learning for domain adaptation of semantic segmentation,” in *Proceedings of the IEEE"
Y. Yang and S. Soatto, “Fda: Fourier domain adaptation for semantic segmentation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 4085–4095.

Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by back-propagation,” in International conference on machine learning. PMLR, 2015, pp. 1180–1189.

T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, “Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2019, pp. 2517–2526.

X. Huang and S. Belongie, “Arbitrary style transfer in real-time with cross-domain mixed sampling,” in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 1379–1389.

K. Mei, C. Zhu, J. Zou, and S. Zhang, “Instance adaptive self-training for unsupervised domain adaptation,” arXiv preprint arXiv:2008.12197, 2020.

P. Zhang, B. Zhang, T. Zhang, D. Chen, Y. Wang, and F. Wen, “Prototype-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, 2021, pp. 12414–12424.

L. Hoyer, D. Dai, and L. Van Gool, “Dafomer: Improving network architectures and training strategies for domain-adaptive semantic segmentation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 11539–11551, 2020.

G. Li, G. Kang, W. Liu, Y. Wei, and Y. Yang, “Content-consistent matching for domain adaptive semantic segmentation,” in European Conference on Computer Vision. Springer, 2020, pp. 440–456.

Y. Luo, P. Liu, T. Guan, J. Yu, and Y. Yang, “Adversarial style mining for one-shot unsupervised domain adaptation,” Advances in Neural Information Processing Systems, vol. 33, pp. 20612–20623, 2020.

X. Wu, Z. Wu, Y. Lu, L. Ju, and S. Wang, “Style mixing and patch-wise prototypical matching for one-shot unsupervised domain adaptive semantic segmentation,” arXiv preprint arXiv:2112.04665, 2021.

S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, “Improving robustness against common corruptions by covariate shift adaptation,” Advances in Neural Information Processing Systems, vol. 33, pp. 11539–11551, 2020.

D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, “Tent: Fully test-time adaptation by entropy minimization,” arXiv preprint arXiv:2006.10726, 2020.

M. Zhang, S. Levine, and C. Finn, “Memo: Test time robustness via adaptation and augmentation,” arXiv preprint arXiv:2110.09506, 2021.

A. Krizhevsky, G. Hinton et al., “Learning multiple layers of features from tiny images,” 2009.

Y. Sun, X. Wang, Z. Liu, J. Müller, A. Efros, and M. Hardt, “Test-time training with self-supervision for generalization under distribution shifts,” in International Conference on Machine Learning. PMLR, 2020, pp. 9229–9248.

Y. Liu, P. Kothari, B. van Delft, B. Bellot-Gurlet, T. Mordan, and A. Alahi, “Tit++: When does self-supervised test-time training fail or thrive?” Advances in Neural Information Processing Systems, vol. 34, 2021.

A. Bartler, A. Bühlner, F. Wiewel, M. Döbler, and B. Yang, “Mi3: Meta test-time training for self-supervised test-time adaptation,” in International Conference on Artificial Intelligence and Statistics. PMLR, 2022, pp. 3080–3090.

D. Chen, D. Wang, T. Darrell, and S. Ebrahimi, “Contrastive test-time adaptation,” in CVPR, 2022.

J. Liang, D. Hu, and J. Feng, “Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation,” in International Conference on Machine Learning. PMLR, 2020, pp. 6028–6039.

C. K. Mummad, R. Hutmacher, K. Rambach, E. Levinkov, T. Brox, and J. H. Metzen, “Test-time adaptation to distribution shift by confidence maximization and input transformation,” arXiv preprint arXiv:2106.14999, 2021.