2021 IEEE Intelligent Transportation Systems Conference (ITSC)
Indianapolis, USA. September 19-21, 2021

Continual Unsupervised Domain Adaptation for Semantic Segmentation by Online Frequency Domain Style Transfer

Jan-Aike Termöhlen[1] Marvin Klingner[1] Leon J. Brettin[1] Nico M. Schmidt[2] Tim Fingscheidt[1]

***Abstract*— When deep neural networks are deployed in a highly automated vehicle for environment perception tasks in an unseen (target) domain that differs from the training (source) domain, the mismatch will result in decreased performance. Domain adaptation methods aim at overcoming this mismatch. Many recently investigated methods for unsupervised domain adaptation train a model using labeled source data and unlabeled target data at the same time. These methods assume that data from the target domain is available during the source domain training, which is not always the case in real applications. In this paper we present a way to perform an online style transfer for continual domain adaptation which improves performance on (multiple) unseen target domains using a given perception model. The approach is based on an image style transfer in the frequency domain and requires neither an adjustment of the given source-trained model parameters to the target domain, nor does it require any considerable amount of memory for storing its frequency domain representation of the source domain style, which is particularly important considering the hardware limitations in an autonomous vehicle.**

I. Introduction

Many environment perception tasks for highly automated driving employ deep neural networks which require labeled datasets for training. The generation of these labeled datasets is both time-consuming and costly. For example, pixel-wise labeling for semantic segmentation takes about 90 minutes for one image [6]. The need for cheap labeled data for semantic segmentation has led to the creation of several synthetic datasets, where these labels can be generated at almost no cost [34], [35] and scenes can be generated that would be too dangerous to create in real life, e.g., traffic accidents. However, when models are trained on these synthetic datasets (source domain $\mathcal{D}_S$), the domain gap to real data (target domain $\mathcal{D}_T$) typically leads to decreased performance during inference. To overcome this domain gap without having to label target domain data, methods for unsupervised domain adaptation (UDA) were proposed [13], [12], [44], [10], [41], [52], [32], [43], [18], [40]. For most unsupervised adaptation methods, both (labeled) source and (unlabeled) target data must be provided during training. This means that the target domain must be known in advance and data from it must be available, which is not always guaranteed in real applications such as environment perception in highly automated driving. Domain generalization methods aim at training a network to generalize well towards unseen domains, but they must already be employed during source domain training and are therefore not applicable to the here envisioned task of adapting an already trained model to a new domain (utilizing only a low-memory source-domain representation). There already exist some works that propose methods for a subsequent domain adaptation by replaying source domain feature maps [45], either by model adaptation with weight constraints [24], or by adaptation of the batch norm statistics [19]. All these methods have in common that the network weights (or batch norm statistics) have to be adapted to the changed domain. This adaptation process requires some learning epochs or at least the processing of multiple minibatches. In contrast, our proposed method acts as a simple pre-processor prior to the segmentation network. The adaptation using this method is immediately effective and does not introduce additional algorithmic delay. Moreover, the network weights are not adjusted after the initial training on the source domain, which prevents degradation of the network's performance over time. The same method can be used for multiple different target domains, or a continuously changing target domain, accordingly, it

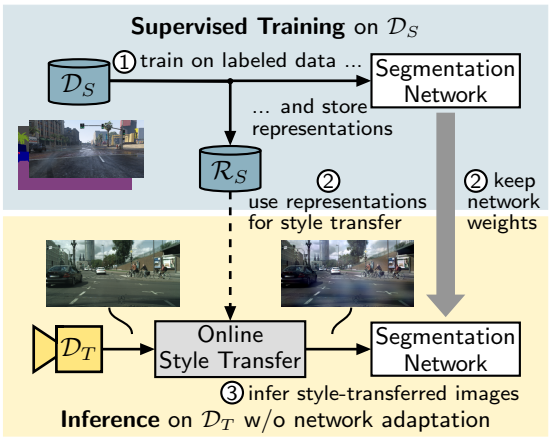

Fig. 1: High-level overview of the **proposed method**. The online style transfer acts as a pre-processor for target domain data $\mathcal{D}_T$ to the segmentation network and requires only a representation $\mathcal{R}_S$ of the source domain with a very small memory footprint. The source domain ($\mathcal{D}_S$) network weights of the segmentation network do not need to be adjusted during inference.

[1] Jan-Aike Termöhlen, Marvin Klingner, Leon J. Brettin, and Tim Fingscheidt are with the Institute for Communications Technology, Technische Universität Braunschweig, Schleinitzstr. 22, 38106 Braunschweig, Germany {j.termoehlen, m.klingner, l.brettin, t.fingscheidt}@tu-bs.de

[2] Nico M. Schmidt is with CARIAD SE, Berliner Ring 2, Brieffach 1080/2, 38440 Wolfsburg nico.schmidt@cariad.technology

2021 IEEE International Intelligent Transportation Systems Conference (ITSC) | 978-1-7281-9142-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ITSC48978.2021.9564566

can be considered as a continuous UDA. Our proposed preprocessor uses an online style transfer that converts images of any target domain to be more similar to those of the source domain. Typically, methods for style transfer use style transfer networks [2], [15], [8], [53], [36] which must be trained to transfer input images to another domain. These style transfer networks can also lead to unexpected artifacts in the style-transferred target images and may alter the semantics in some cases, although some recent improvements have also been made in this area aiming at preventing this [47]. Further disadvantages of these methods are the memory footprint and computational complexity, since the style transfer network, e.g., a generative adversarial network (GAN), has to be stored and inferred during adaptation.

Recently, there is an increasing number of methods that operate in the frequency domain, e.g., to detect and defend against adversarial attacks [16] or to perform a style transfer [48]. The online style transfer approach in this work was inspired by the Fourier domain adaptation (FDA) method by Yang *et al.* [48]. To the best of our knowledge they were the first to employ a style transfer in the frequency domain. However, their approach is only suitable for *offline* domain adaptation, since the segmentation network is trained on source images that are transferred to look like target domain images. This means that data from both domains need to be available during training. We will show in this work that by generating a codebook in the source domain and adapting the mixing function and mask used for the style transfer it is possible to perform an *online* style transfer that increases performance on multiple unseen target domains.

Our contribution with this work is threefold. First, we propose a method for online frequency domain style transfer that does not require a style transfer network to be trained. Second, we analyze the impact of the codebook size on the target domain performance and show that surprisingly small codebooks are best. Third, we show that the method can be used on multiple unseen target domains and is independent of the network architecture, which shows the strong potential of our method for environment perception in highly automated vehicles.

The paper is structured as follows. In Section 2 we provide related work in the field of domain generalization, (offline) UDA, as well as continual UDA. In Section 3 we introduce the employed methods and mathematical notations. Section 4 introduces employed datasets, network architectures, evaluation metrics and the employed hyperparameters of our method. Finally, we will present experimental results in Section 5, before providing conclusions in Section 6.

II. Related Work

In this section we discuss related methods for our continual unsupervised domain adaptation (UDA) for environment perception. We start with methods for domain generalization, followed by methods for offline UDA that require availability of both source and target domain data during training. Afterwards, we introduce related methods for continual UDA.

A. Domain Generalization

Domain generalization methods aim at training a network in a way that it performs better in unseen domains, i.e., the network generalizes better [7], [22], [37]. For this, no data from the target domain is used. However, domain generalization methods are already applied during training and employ, e.g., multiple source datasets for training [51]. In contrast, our method allows a continual adaptation to new domains for *a given model without adapting the network parameters*. Accordingly, our method is even applicable to models that have been trained on a single source domain, but it requires a small extra source domain representation.

B. Offline Unsupervised Domain Adaptation

Methods for offline UDA that require source and target domain data simultaneously can typically be categorized into three different approaches, with many modern methods combining these approaches. Either a domain-adversarial training is employed, which forces the feature extractor to learn domain-invariant features [41], [27], [3], [5], [31], [14], [32], [43], [18], or a self-training method is employed, where pseudo-labels are generated for the target domain images [28], [27], [43], [39], [23], [32], or the source images are style-transferred to the target domain [12], [44], [10], [46], [17], [48], e.g., by using a generative adversarial network. In the following, the related work that employs a style transfer will be discussed in more detail.

Style transfer methods aim at transferring the style of the source domain images to make them look more like the anticipated target domain, but without changing their semantics [12], [44]. Hoffman *et al*. [12] employed a cycle-consistent GAN for the image generation and Gong *et al*. [10] used intermediate domains of the style transfer to achieve a more stable training. A different method proposed by Yang *et al*. [46] performs the translation from target to source domain with an additional image reconstruction from the predicted labels in both domains. Kim *et al*. [17] investigated the specific impact of the texture for style transfer and proposed a method that explicitly transfers the target domain textures to the source domain. Yang *et al*. [48] proposed a style transfer by implanting the low frequencies from the target domain amplitude spectra in the source domain. We will build upon this approach, however, allowing it to be used *for a given model in an online fashion without adapting the network weights*.

C. Continual Unsupervised Domain Adaptation

Continual UDA methods are applied to a given model in a new domain. However, the training data of the source domain can no longer be accessed for the adaptation process. Continual UDA is similar to source-free UDA [19], [26], [49], [20], except that source-free UDA does do not use *any* explicit representation of the source domain other than the trained network weights.

Wulfmeier *et al*. [45] were the first to introduce the idea of continual unsupervised adaptation for visual perception. They used a generative replay model, that was trained to

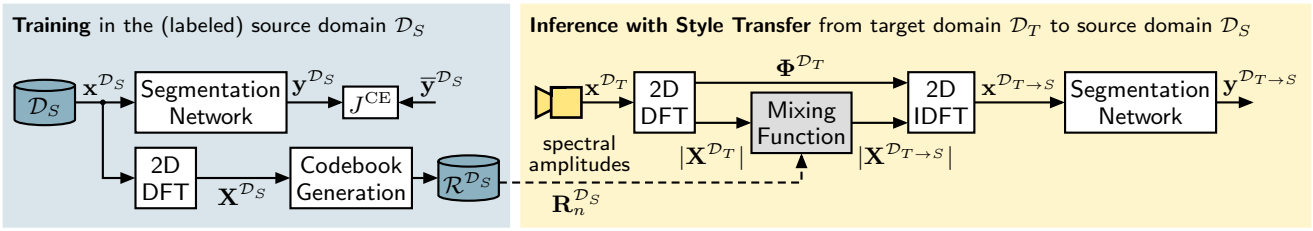

Fig. 2: **Proposed online style transfer** for continual unsupervised domain adaptation. Initially, the segmentation network is trained on images from source domain dataset $\mathcal{D}_S = \{\mathbf{x}^{\mathcal{D}_S}\}$. At the same time, the amplitude DFT spectra of the source domain images are used to generate a codebook $\mathcal{R}^{\mathcal{D}_S}$. During inference, a DFT-domain style transfer using this codebook is applied to the target domain images $\mathbf{x}^{\mathcal{D}_T}$, which makes them look more similar to the images from the source domain. Details of the mixing function are given in Fig. 3.

generate source domain-like feature maps from random noise vectors, to train a feature-level domain discriminator in the new domain. Schutera *et al*. [36] train a style transfer network for night-to-day transfer and use this for domain adaptation. Yeh *et al*. [49] rely only on the domain knowledge in the network parameters and use variational inference in the target domain. Li *et al*. [26] propose a self-training method with pseudo-labels obtained by self-entropy descent and false negative simulation. Li *et al*. [25] propose an online adaptation for visual odometry using statistics of features that are stored during training. Other methods use generator networks to generate source domain-like samples for the adaptation process [24], [20]. All of the above methods are either not applicable to semantic segmentation, since they were designed for other tasks such as object detection [36], [49], [26] and object recognition [24], [20], or they have a large memory footprint caused by the employed generator networks [45], [36]. The unsupervised batch norm adaptation (UBNA) proposed by Klingner *et al*. [19] is effective for semantic segmentation networks and does not require source domain representations. It adapts a given model by updating its batch norm parameters. However, the adaptation is not effective instantaneously, since processing of multiple minibatches is required and the network weights are adjusted in a separate adaptation phase before target domain inference can take place. Although, as a consequence, UBNA cannot be considered to be a continual domain adaptation method, still we report it as a reference approach, to fill the gap of actually comparable baselines.

III. Online Frequency Domain Style Transfer

In this section, the notations and methods are introduced. An overview of our proposed method for continual UDA for semantic segmentation is given in Fig. 2. First, training in the source domain (Fig. 2, left, blue) and afterwards inference in the target domain including the online frequency domain style transfer (Fig. 2, right, yellow) are explained.

A. Source-Domain Training: Segmentation, Codebook

During training in the labeled source domain $\mathcal{D}_S$ not only a segmentation network is trained, but also a codebook $\mathcal{R}^{\mathcal{D}_S}$ is generated, which comprises representations of the source domain in the frequency domain, as we will show now.

Semantic Segmentation: As input to the segmentation network we define the image $\mathbf{x} \in \mathbb{G}^{H\times W\times C}$, where $\mathbb{G}$ denotes the set of integer gray values, H and W the image height and width in pixels, and $C = 3$ the number of color channels. The network transforms the images into outputs $\mathbf{y} = (y_{i,s}) \in \mathbb{I}^{H\times W\times S}$ with posterior probability (score) $y_{i,s} = P(s|i,\mathbf{x})$ for each class $s \in \mathcal{S}$ at pixel index $i \in \mathcal{I} = \{1, 2, ..., H \cdot W\}$. Here, $\mathcal{S} = \{1, 2, ..., S\}$ denotes the set of S classes and $\mathbb{I} = [0, 1]$. The final segmentation map $\mathbf{m} = \arg\max_{s\in\mathcal{S}} \mathbf{y} \in \mathcal{S}^{H\times W}$ is obtained with argmax operating on each pixel i of the network output $\mathbf{y} = (\mathbf{y}_i)$ individually. Superscripts on $\mathbf{x}, \mathbf{y}$ and $\mathbf{m}$ denote the domain from which they stem, with $\mathcal{D}_S$ being the source domain and $\mathcal{D}_T$ being the target domain.

Codebook Generation: During training in the source domain $\mathcal{D}_S$ a codebook $\mathcal{R}^{\mathcal{D}_S}$ is generated. First, the 2D discrete Fourier transform (2D DFT) $\mathbf{X}_c = (X_{c,k,\ell}) \in \mathbb{C}^{H\times W}$ for each color channel $c \in \mathcal{C} = \{1, 2, 3\}$ of the input image is calculated separately as follows:

$$X_{c,k,\ell} = \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} x_{c,h,w} \cdot e^{-j2\pi(\frac{k}{H}h + \frac{\ell}{W}w)}, \qquad (1)$$

with k, ℓ denoting the indices along spectrum height and width, $x_{c,h,w}$ denoting the pixel values of color channel c in the input image $\mathbf{x}$ with the indices h, w along image height and width, and $\mathbb{C}$ being the set of complex numbers. The complex DFT spectrum of the entire input image is obtained by concatenating the complex spectra for all color channels and is denoted as $\mathbf{X} = (\mathbf{X}_c) = (X_{c,k,\ell}) \in \mathbb{C}^{H\times W\times C}$. For the online frequency domain style transfer we only use the amplitude spectrum

$$|X_{c,k,\ell}| = \sqrt{\text{Re}^2[X_{c,k,\ell}] + \text{Im}^2[X_{c,k,\ell}]}, \qquad (2)$$

of the source domain image, with $\text{Re}[\cdot], \text{Im}[\cdot]$ denoting the real and imaginary part of the complex spectrum, respectively. During training, the amplitude spectrum for each source domain sample is computed and then used to generate a codebook $\mathcal{R}^{\mathcal{D}_S} = \{\mathbf{R}_n^{\mathcal{D}_S}\}$ with $n \in \mathcal{N} = \{1, 2, ..., N\}$ denoting the index and N denoting the number of codebook vectors and thereby the number of representations of the source domain that are being stored. Each codebook vector is defined as $\mathbf{R}_n = (R_{c,k,\ell,n}) \in (\mathbb{R}^+)^{H\times W\times C}$, with c, k, ℓ

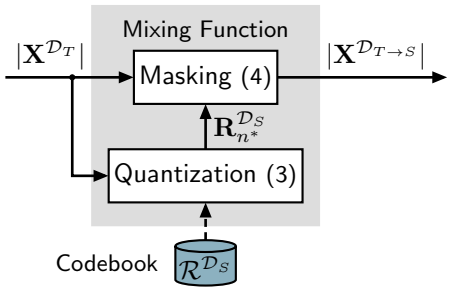

Fig. 3: **Mixing function** transforming $|\mathbf{X}^{\mathcal{D}_T}|$ into $|\mathbf{X}^{\mathcal{D}_{T\to S}}|$, see Fig. 2. The codebook is the only representation of the source domain that has to be stored.

being the indices along the channel, spectrum height, and spectrum width dimension, and $\mathbb{R}^+$ being the set of all positive real-valued numbers. This is the only information from the source domain that is necessary to perform the online style transfer. The codebook is generated using the Linde-Buzo-Gray (LBG) algorithm [29] employing the same training data as is used for semantic segmentation network training (see Sec. IV-A).

B. Target-Domain Inference: Style Transfer

During inference, an online frequency domain style transfer according to the yellow right part of Fig. 2 is applied to the input images from target domain $\mathcal{D}_T$, acting as a pre-processor to the trained segmentation network. First, the 2D DFT is applied to the input images delivering $\mathbf{X}^{\mathcal{D}_T}$, while the phase spectrum $\mathbf{\Phi}^{\mathcal{D}_T} = (\Phi_{c,k,\ell}^{\mathcal{D}_T}) = \arg(\mathbf{X}^{\mathcal{D}_T})$ is kept unaltered, the amplitude spectrum $|\mathbf{X}^{\mathcal{D}_T}|$ is passed to the mixing function depicted in Fig. 3, which delivers the style-transferred amplitude spectrum $|\mathbf{X}^{\mathcal{D}_{T\to S}}|$. The absolute operator $|\cdot|$ is applied element-wise. The processing steps of the mixing function are as follows.

Mixing Function: At first, a quantization is performed identifying the best matching source domain representation $\mathbf{R}_{n^*}^{\mathcal{D}_S}$ from the codebook for the current target domain sample $|\mathbf{X}^{\mathcal{D}_T}|$. This is accomplished by selecting the codebook entry that holds

$$n^* = \arg\min_{n\in\mathcal{N}} \left\| \mathbf{R}_n^{\mathcal{D}_S} - |\mathbf{X}^{\mathcal{D}_T}| \right\|_2^2. \tag{3}$$

The selected codebook entry $\mathbf{R}_{n^*}^{\mathcal{D}_S}$ is then mixed with the target domain amplitude spectrum $|\mathbf{X}^{\mathcal{D}_T}|$ using the following masking technique:

$$|\mathbf{X}^{\mathcal{D}_{T\to S}}| = (\mathbf{1} - \mathbf{M}) \circ |\mathbf{X}^{\mathcal{D}_T}| + \mathbf{M} \circ \mathbf{R}_{n^*}^{\mathcal{D}_S}, \tag{4}$$

with $\circ$ denoting element-wise multiplication, $\mathbf{1}$ being an all-ones tensor, and $\mathbf{M}$ denoting the employed mask.

The binary mask is written in centralized form as $\mathbf{M} = (M_{c,k',\ell'}) \in \mathbb{I}^{H\times W\times C}$ with $k' = k - \frac{H}{2}$ and $\ell' = \ell - \frac{W}{2}$ denoting the shifted indices. Following Yang *et al.* [48], the binary mask is defined as:

$$M_{c,k',\ell'} = \begin{cases} 1 & -b \le k' < b \wedge -b \le \ell' < b \\ 0 & \text{else,} \end{cases} \tag{5}$$

TABLE I: **Datasets**: The training sets and full sets are source domain datasets $\mathcal{D}_S$. *Note that we use the 200 training images from the KITTI-2015 dataset as our validation set.

Dataset Name	**# Images in**			
	$\mathcal{D}^{\text{train}}$	$\mathcal{D}^{\text{val}}$	$\mathcal{D}^{\text{test}}$	$\mathcal{D}^{\text{full}}$
GTA5 [34]	12,403	6,382	6,181	24,966
SYNTHIA [35]	9,400	-	-	-
Cityscapes [6]	2,975	500	(1,525)	-
BDD100k [50]	7,000	1,000	(2,000)	-
KITTI(-2015) [9], [1]	29,000	200*	(200)	-

with $B = 2b \ll H, W$ and $B \times B$ being the size of the non-zero part of the mask. To apply the masks to the amplitude spectra following (4), they must be shifted, since the spectra are in non-centralized form. For this, the first and third quadrant and the second and fourth quadrant are swapped.

Finally, the style-transferred image $\mathbf{x}^{\mathcal{D}_{T\to S}} = (x_{c,h,w}^{\mathcal{D}_{T\to S}})$ is computed by the 2D IDFT (cf. Fig. 2, right side) using the unaltered phase spectrum $\mathbf{\Phi}^{\mathcal{D}_T}$ and the mixed amplitude spectrum $|\mathbf{X}^{\mathcal{D}_{T\to S}}|$ from (4) as follows:

$$x_{c,h,w}^{\mathcal{D}_{T\to S}} = \frac{1}{HW} \sum_{k=0}^{H-1} \sum_{\ell=0}^{W-1} X_{c,k,\ell}^{\mathcal{D}_{T\to S}} \cdot e^{j2\pi\left(\frac{h}{H}k + \frac{w}{W}\ell\right)}, \tag{6}$$

with the complex spectrum obtained by

$$X_{c,k,\ell}^{\mathcal{D}_{T\to S}} = |X_{c,k,\ell}^{\mathcal{D}_{T\to S}}| \cdot e^{j\Phi_{c,k,\ell}^{\mathcal{D}_T}}. \tag{7}$$

According to Fig. 2, this style-transferred image is then fed into the given semantic segmentation network that has been trained on the source domain only, providing $\mathbf{y}^{\mathcal{D}_{T\to S}}$.

The memory requirement of a codebook using the full amplitude spectrum is $H \cdot W \cdot C \cdot N \cdot$ real-valued words, for a typical input color image of size 1024×512 this results in approximately $1500N$ kwords. When storing only non-zero portion of size $B\times B = 2\times2$, the codebook footprint results in only $12N$ words.

IV. Experimental Setup

In this section, we will first introduce the employed datasets, architectures, and metric used for evaluation. Afterwards, the training and inference hyperparameters will be explained. All metrics, architectures, and procedures are implemented using `PyTorch` [33].

A. Datasets

The experiments are conducted on synthetic and real datasets for semantic segmentation to identify, in which setup our method provides the highest performance gains. The employed datasets are widely used for semantic segmentation in environment perception for autonomous driving. They are listed in Table I, along with the number of images in the respective training set $\mathcal{D}^{\text{train}}$, validation set $\mathcal{D}^{\text{val}}$, and test set $\mathcal{D}^{\text{test}}$. The images and labels of all datasets are resized for the training.

GTA5: The GTA5 dataset [34] consists of 24,966 synthetic images with a resolution of 1914×1052. This dataset is only used as a source domain $\mathcal{D}_S$. Typically, when training

domain adaptation methods, the full GTA5 dataset is used for training and no distinction is made between $\mathcal{D}^{\text{train}}$, $\mathcal{D}^{\text{val}}$, and $\mathcal{D}^{\text{test}}$, we call this $\mathcal{D}^{\text{full}}$. When $\mathcal{D}^{\text{full}}$ is used as the source domain, then we have $N_{\max} = 24{,}966$ = "all". Following Yang *et al.* [48], during training the images and labels are resized to a resolution of 1280×720 and then randomly cropped to a resolution of 1024×512.

SYNTHIA: The SYNTHIA dataset [35] consists of $9{,}400$ synthetic images with a resolution of 1280×760. For this dataset only $\mathcal{D}^{\text{train}}$ is defined, which means that we use it only for training in the source domain $\mathcal{D}_S$. When trained on SYNTHIA, we have $N_{\max} = 9{,}400$ = "all". As with the GTA5 dataset, during training the images and labels are randomly cropped to a resolution of 1024×512.

Cityscapes: The Cityscapes dataset [6] consists of $5{,}000$ images with a resolution of 2048×1024. When Cityscapes is used as a source domain $\mathcal{D}_S$, we get $N_{\max} = 2{,}975$ = "all". During training, adaptation, and evaluation, the images are resized to 1024×512.

BD100k: The BDD100k dataset [50] consists of $10{,}000$ images with a resolution of 1280×720. The training set $\mathcal{D}^{\text{train}}$ consists of $7,000$ images, that are only used for the domain adaptation with the unsupervised batch norm adaptation (UBNA) reference method [19]. We use the $1{,}000$ images from $\mathcal{D}^{\text{val}}$ for evaluation. The $2{,}000$ images from $\mathcal{D}^{\text{test}}$ cannot be used, since neither labels nor an evaluation server are provided. During adaptation and evaluation, the images are resized to 1024×512.

KITTI(-2015): The KITTI-2015 dataset [1] consists of 400 images with a resolution of 1242×375, although the resolution is not the same for all images and sometimes differs by up to four pixels. That is why we resized the images to 1024×512 during adaptation and evaluation. The 200 training images from the KITTI-2015 dataset are used as $\mathcal{D}^{\text{val}}$. The training set $\mathcal{D}^{\text{train}}$ consists of $29,000$ unlabeled images from the standard KITTI dataset [9] that are only used for the domain adaptation with the unsupervised batch norm adaptation (UBNA) reference method [19].

B. Network Architectures and Evaluation Metric

We employ two different segmentation networks. First, a `DeepLabv2` [4] model with a `ResNet101` [11] feature extractor, and second, an `FCN-8s` [30] model with a `VGG-16` [38] feature extractor. We evaluate our method using the mean intersection-over-union (mIoU) metric with $S = 19$ classes as specified in the Cityscapes dataset [6]. For models that employ a `ResNet101` feature extractor, such as the `DeepLabv2` [4] model, and that have been trained on the SYNTHIA dataset [35], we follow common practice [21], [42], [19] and evaluate on a subset containing only 13 classes.

C. Training and Inference Settings

In the following, we will provide some training settings and hyperparameters used for the segmentation training in a labeled source domain. Afterwards, settings for inference including the online style transfer are given.

Semantic Segmentation Training: The segmentation network takes the images $\mathbf{x}^{\mathcal{D}_S}$ as input and generates the output scores $\mathbf{y}^{\mathcal{D}_S}$. We then compute the cross-entropy (CE) loss

$$J^{\text{CE}} = \sum_{i \in \mathcal{I}} \sum_{s \in \mathcal{S}} \overline{y}_{i,s} \cdot \log(y_{i,s}) \,, \tag{8}$$

where $\overline{y}_{i,s}$ denotes the one-hot encoded label for class s at pixel position i, which is derived from the i-th entry in the label mask $\overline{\mathbf{m}} = (\overline{m}_i)$, with $\overline{m}_i \in \mathcal{S}$.

For source domain training we mostly select the hyperparameters following Yang *et al.* [48]. The maximum number of iterations is set to $\tau_{\max} = 150,000$. For the GTA5 dataset we employ early stopping after $100,000$ iterations. For `DeepLabv2`, the stochastic gradient descent (SGD) optimizer is used with momentum of $\beta = 0.9$, while the learning rate follows a polynomial scheduling $\eta(\tau) = \eta_0(1 - \frac{\tau}{\tau_{\max}})^{0.9}$, with η_0 being the initial learning rate that is set to $\eta_0 = 2.5 \cdot 10^{-4}$ when trained on GTA5, and to $\eta_0 = 1 \cdot 10^{-4}$ when trained on SYNTHIA. We use a weight decay of 0.0005. For `FNC-8s`, the Adam optimizer is employed with an initial learning rate $\eta_0 = 1 \cdot 10^{-5}$ and a learning rate schedule that decreases the learning rate by a factor of 0.1 every $50,000$ iterations. The momentum values for Adam are $\beta_1 = 0.9$ and $\beta_2 = 0.99$.

During training on the source domain we compute the 2D DFT (1) on the source domain images, extract the amplitude spectra and train a codebook $\mathcal{R}^{\mathcal{D}_S}$ using the LBG algorithm [29].

Inference with Online Style Transfer: During inference, the images get resized to the size of the source domain images for the low-frequency amplitude spectrum mixing (4). The output of the segmentation network is resized to the original label size of the respective input sample.

Unsupervised BatchNorm Adaptation: We will compare our approach to the unsupervised batch norm adaptation (UBNA) by Klingner *et al.* [19] as reference method. Since they use a different network architecture in their paper than we do, we re-simulate their method using our models. The adaptation is done as suggested in the paper in $\kappa = 50$ steps with a minibatch size of 6.

V. Experimental Results

In the following chapter we show the experimental results of our approach. We evaluate our method for the perception task of semantic segmentation on multiple established benchmarks for UDA from synthetic-to-real data (default) and real-to-real data. First, we will perform multiple experiments showing the influence of the employed codebook size N and the influence of the mask size $B \times B$ used in the mixing function. Afterwards, we will compare our method to non-continual UDA methods. Finally, we will show results for real-to-real domain adaptation, again employing multiple target domain datasets, and synthetic-to-real domain adaptation for different network architectures.

A. Influence of Codebook and Mask Size

We investigate the effect of varying codebook sizes $N \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, \text{all}\}$ on the

TABLE II: Performance (mIoU [%]) of offline reference methods, offline UDA reference w/o source domain representations (UBNA [19]), no adaptation, and our proposed continual UDA by style transfer. The adaptation is performed from **GTA5** ($\mathcal{D}^{\text{full}}_{S=\text{GTA5}}$) or **SYNTHIA** ($\mathcal{D}^{\text{train}}_{S=\text{SYNTHIA}}$) to **Cityscapes** ($\mathcal{D}^{\text{val}}_{T=\text{Cityscapes}}$). The `DeepLabv2` model is used.

Source Domain	Method	Continual Adaptation	Class IoU [%]																			mIoU [%]
			Road	Sidewalk	Building	Wall	Fence	Pole	T. Light	T. Sign	Veg.	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	M.bike	Bicycle	
SYNTHIA	DCAN [44]	✗	81.5	33.4	72.4	-	-	-	8.60	10.5	71.0	-	68.7	51.5	18.7	75.3	-	22.7	-	12.8	28.1	42.7
	BDL [27]	✗	86.0	46.7	80.3	-	-	-	14.1	11.6	79.2	-	81.3	54.1	27.9	73.7	-	42.2	-	25.7	45.3	51.4
	LTIR [17]	✗	92.6	53.3	79.2	-	-	-	1.6	7.5	78.6	-	84.4	52.6	20.0	82.1	-	34.8	-	14.6	39.1	49.3
	FDA-MBT [48]	✗	79.3	35.0	73.2	-	-	-	19.9	24.0	61.7	-	82.6	61.4	31.1	83.9	-	40.8	-	38.4	51.1	52.5
	LDR [46]	✗	85.1	44.5	81.0	-	-	-	16.4	15.2	80.1	-	84.8	59.4	31.9	73.2	-	41.0	-	31.6	44.7	**53.1**
	UBNA [19]	✗	45.1	22.7	65.9	-	-	-	7.9	17.8	75.6	-	80.5	51.2	18.8	46.8	-	11.0	-	17.4	24.6	37.3
	w/o adaptation	-	36.8	17.7	65.6	-	-	-	3.8	11.8	70.7	-	79.2	54.4	18.6	43.8	-	20.8	-	12.8	24.3	35.4
	Ours ($N=64; B=2$)	✓	63.6	24.0	65.7	-	-	-	4.3	13.7	62.5	-	77.2	54.8	20.0	62.1	-	9.3	-	15.5	29.9	**38.7**
GTA5	DCAN [44]	✗	88.5	37.4	79.3	24.8	16.5	21.3	26.3	17.4	80.8	30.9	77.6	50.2	19.2	77.7	21.6	27.1	2.70	14.3	18.1	38.5
	DLOW [10]	✗	87.1	33.5	80.5	24.5	13.2	29.8	29.5	26.6	82.6	26.7	81.8	55.9	25.3	78.0	33.5	38.7	0.0	22.9	34.5	42.3
	BDL [27]	✗	91.0	44.7	84.2	34.6	27.6	30.2	36.0	36.0	85.0	43.6	83.0	58.6	31.6	83.3	35.3	49.7	3.3	28.8	35.6	48.5
	LTIR [17]	✗	92.9	55.0	85.3	34.2	31.1	34.9	40.7	34.0	85.2	40.1	87.1	61.0	31.1	82.5	32.3	42.9	0.3	36.4	46.1	50.2
	FDA-MBT [48]	✗	92.5	53.3	82.4	26.5	27.6	36.4	40.6	38.9	82.3	39.8	78.0	62.6	34.4	84.9	34.1	53.1	16.9	27.7	46.4	**50.5**
	LDR [46]	✗	90.8	41.4	84.7	35.1	27.5	31.2	38.0	32.8	85.6	42.1	84.9	59.6	34.4	85.0	42.8	52.7	3.4	30.9	38.1	49.5
	UBNA [19]	✗	78.8	30.1	75.5	21.2	23.8	30.6	34.1	18.3	82.6	25.0	73.2	59.4	25.3	73.9	27.0	18.9	0.1	26.4	26.3	39.5
	w/o adaptation	-	60.3	18.1	64.3	8.5	13.9	26.8	28.5	14.8	57.9	10.0	67.4	54.2	25.5	71.0	28.7	2.9	0.2	26.30	23.0	31.7
	Ours ($N=64; B=2$)	✓	83.5	28.0	75.9	18.0	22.2	30.3	30.6	19.4	82.0	34.4	71.7	56.3	25.3	71.4	21.7	33.5	0.1	28.2	33.0	**40.3**

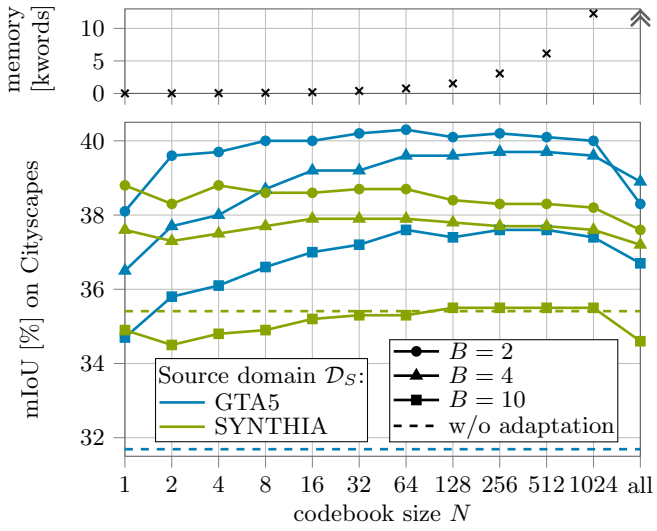

Fig. 4: **Influence of the codebook size** N on the mIoU [%] and the memory footprint [kwords] of the codebook. Performance of `DeepLabv2` on **Cityscapes** ($\mathcal{D}^{\text{val}}_{T=\text{Cityscapes}}$) after training on **GTA5** ($\mathcal{D}^{\text{full}}_{S=\text{GTA5}}$) or **SYNTHIA** ($\mathcal{D}^{\text{train}}_{S=\text{SYNTHIA}}$).

performance in the target domain for different mask sizes $B \times B$ with $B \in \{2, 4, 10\}$, with $B = 10$ being used for the smallest mask that was employed by Yang *et al.* [48]. As can be seen in Figure 4, the performance of the style transfer depends on the mask size. With smaller mask sizes we also reduce visual artifacts that are caused by the implantation of source domain amplitudes. It can be seen that our method yields better performance than the baseline without adaptation (dashed line) for all codebook sizes N,

TABLE III: Performance (mIoU [%]) of the `DeepLabv2` model trained on **Cityscapes** ($\mathcal{D}^{\text{train}}_{S=\text{Cityscapes}}$) in **various target domains** using **various codebook sizes** N and $B = 2$ during inference. BDD100k ($\mathcal{D}^{\text{val}}_{T=\text{BDD100k}}$) and KITTI-2015 ($\mathcal{D}^{\text{val}}_{T=\text{KITTI-2015}}$) are used as target domains. Note also the performance in the source domain ($\mathcal{D}^{\text{val}}_{S=\text{Cityscapes}}$).

Method	mIoU [%] on		
	Cityscapes	BDD100k	KITTI-2015
w/o adaptation	62.1	41.1	44.3
UBNA [19] ($\rightarrow$ BDD100k)	59.3	**43.3**	47.9
UBNA [19] ($\rightarrow$ KITTI)	57.6	42.6	**53.6**
Ours ($N = 1; B=2$)	61.8	40.6	41.3
Ours ($N = 64; B=2$)	**62.2**	41.6	43.5
Ours ($N =$ "all"; $B=2$)	62.1	41.8	44.8

when using GTA5 ($\mathcal{D}^{\text{full}}_{S=\text{GTA5}}$) as a source domain (blue plots) with a maximum mIoU of 40.3 % for $N = 64$ and $B = 2$. This equals a performance gain over the baseline without adaptation (dashed blue line) of 8.6 % absolute. When trained on SYNTHIA ($\mathcal{D}^{\text{train}}_{S=\text{SYNTHIA}}$) (green plots) already smaller masks with $B \in \{2, 4\}$ yield better results than the baseline (dashed green line) for all codebook sizes N, while $B = 10$ already seems to be too large. The maximum gain that we can achieve is 3.4 % absolute for $N = 1$ and $B = 2$. For both datasets, it is interesting to observe that the largest codebook size (i.e., storing all source domain DFT amplitude spectra as "codebook", dubbed "all") does not yield the best results. Since with increasing codebook size the memory requirement also increases, as can be seen in the upper plot in Figure 4, we suggest using a codebook size of $N = 64$, which gives the best results when trained on GTA5 and is close to optimal when trained on SYNTHIA

TABLE IV: Performance (mIoU [%]) of DeepLabv2 and FCN-8s on **Cityscapes** ($\mathcal{D}^{\text{val}}_{T=\text{Cityscapes}}$) after training on **GTA5** ($\mathcal{D}^{\text{full}}_{S=\text{GTA5}}$).

Method	Segmentation Network	
	DeepLabv2	FCN-8s
w/o adaptation	31.7	13.6
UBNA [19] ($\rightarrow$ Cityscapes)	39.5	13.6
Ours ($N=1; B=2$)	38.1	17.4
Ours ($N=64; B=2$)	**40.3**	**18.1**
Ours ($N=$"all"; $B=2$)	38.3	17.0

with 38.7 %. By only storing the non-zero components used by the smallest mask with $B = 2$, the codebook requires only $B^2 \cdot C \cdot N = 2^2 \cdot 3 \cdot 64 = 768$ words. With a word length of 32 bit this equals 3072 bytes. In further experiments we will always use $B = 2$ and evaluate the two extrema ($N=1$ and $N=$"all"), as well as the proposed $N=64$.

B. Comparison to UDA Approaches

The unique aspect of our method is that we do not perform offline adaptation where both source and target domain data would be required. Nevertheless, in Table II we also report on reference methods that perform offline adaptation by using (amongst other approaches) a style transfer. As expected, the performance of a continual adaptation method is somewhat lower than that of the offline approaches. Compared to the re-simulated UBNA reference method [19], which even requires a separate target domain adaptation phase prior to inference (no continual method), our continual approach achieves a 1.4 % absolute higher mIoU when the model was trained on SYNTHIA, and a 0.8 % absolute higher mIoU when it was trained on GTA5.

C. Real-to-Real Domain Adaptation

We also investigate our proposed method when domain adaptation takes place in a real-to-real setting. The results are shown in Table III. It can be seen that our preferred setting ($N=64; B=2$) improves a bit on BDD100k (0.5 % absolute), while on KITTI-2015 (very similar to the Cityscapes source domain) only the codebook size "all" shows small improvements. In this setting, the UBNA method performs better, but it must be adapted to each target domain before inference, which is indicated by the arrows ($\rightarrow$ BDD100k and $\rightarrow$ KITTI). Since UBNA adjusts network parameters, it is not unexpected that the performance on the source domain decreases from 62.1 % to 59.3 % or 57.6 %, respectively. When using our pre-processor, we do not see a performance drop on the source domain for our preferred setting ($N = 64; B=2$).

D. Different Network Architectures

Finally, we want to show that our proposed method works as a pre-processor regardless of the network architecture. Two semantic segmentation networks that are widely reported in the field of domain adaptation are evaluated in the (preferred) synthetic-to-real setting. It can be seen in Table IV that a performance gain is achievable both for DeepLabv2 and FCN-8s. In the latter case, the UBNA method does not help at all, since the standard FCN-8s does not use any batch norm layers and therefore no UBNA adaptation can take place.

VI. Conclusions

In this work we present a method to perform a continual unsupervised domain adaptation for semantic segmentation which only utilizes a small footprint representation of the source domain. The method can be used as a simple pre-processor to any existing segmentation network for environment perception, when retraining in the source domain is not possible. The low memory footprint and computational complexity allow an application under very constrained hardware settings, e.g., in a highly automated vehicle. We show that the method works preferably well in a synthetic-to-real setting, independently of the used source domain and the network architecture, and is free of any algorithmic delay.

VII. Acknowledgment

The authors gratefully acknowledge support of this work by CARIAD SE, Wolfsburg, Germany. The research leading to these results is funded by the Federal Ministry for Economic Affairs and Energy within the project "KI-DeltaLearning" (project number: 19A19013C). The authors would like to thank the consortium for the successful cooperation.

References

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

[2] Amir Atapour-Abarghouei and Toby P. Breckon. Real-Time Monocular Depth Estimation Using Synthetic Data With Domain Adaptation via Image Style Transfer. In *Proc. of CVPR*, pp. 2800–2810, Salt Lake City, UT, USA, June 2018.

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

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic Image Segmentation With Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, Apr. 2018.

[5] Minghao Chen, Hongyang Xue, and Deng Cai. Domain Adaptation for Semantic Segmentation With Maximum Squares Loss. In *Proc. of ICCV*, pp. 2090–2099, Seoul, Korea, Oct. 2019.

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of CVPR*, pp. 3213–3223, Las Vegas, NV, USA, June 2016.

[7] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain Generalization via Model-Agnostic Learning of Semantic Features. In *Proc. of NeurIPS*, pp. 6447–6458, Vancouver, Canada, Dec. 2019.

[8] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image Style Transfer Using Convolutional Neural Networks. In *Proc. of CVPR*, pp. 2414–2423, Las Vegas, NV, USA, June 2016.

[9] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision Meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, Aug. 2013.

[10] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. DLOW: Domain Flow for Adaptation and Generalization. In *Proc. of CVPR*, pp. 2477–2486, Long Beach, CA, USA, June 2019.

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

[12] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Philip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *Proc. of ICML*, pp. 1989–1998, Stockholm, Sweden, July 2018.

[13] Judy Hoffman, Dequan Wang, Fischer Yu, and Trevor Darrell. FCNs in the Wild: Pixel-Level Adversarial and Constraint-Based Adaptation. *arXiv*, (1612.02649), Dec. 2016.

[14] Jiaxing Huang, Shijian Lu, Dayan Guan, and Xiaobing Zhang. Contextual-Relation Consistent Domain Adaptation for Semantic Segmentation. In *Proc of ECCV*, pp. 705–722, Glasow, UK, Aug. 2020.

[15] Justin Johnson and Alexandre Alahi and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Proc. of ECCV*, pp. 694–711, Amsterdam, Netherlands, Oct. 2016.

[16] Nikhil Kapoor, Andreas Bär, Serin Varghese, Jan D. Schneider, Fabian Hüger, Peter Schlicht, and Tim Fingscheidt. From a Fourier-Domain Perspective on Adversarial Examples to a Wiener Filter Defense for Semantic Segmentation. *arXiv*, (2012.01558), Dec. 2020.

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

[18] Marvin Klingner, Jan-Aike Termöhlen, Jonas Mikolajczyk, and Tim Fingscheidt. Self-Supervised Monocular Depth Estimation: Solving the Dynamic Object Problem by Semantic Guidance. In *Proc. of ECCV*, pp. 582–600, Glasgow, UK, Aug. 2020.

[19] Marvin Klingner, Jan-Aike Termöhlen, Jacob Ritterbach, and Tim Fingscheidt. Unsupervised BatchNorm Adaptation (UBNA): A Domain Adaptation Method for Semantic Segmentation Without Using Source Domain Representations. *arXiv*, (2011.08502), Nov. 2020.

[20] Vinod K. Kurmi, Venkatesh K. Subramanian, and Vinay P. Namboodiri. Domain Impression: A Source Data Free Domain Adaptation Method. In *Proc. of WACV*, pp. 615–625, Virtual, Jan. 2021.

[21] Kuan-Hui Lee, German Ros, Jie Li, and Adrien Gaidon. SPIGAN: Privileged Adversarial Learning from Simulation. In *Proc. of ICLR*, pp. 1–14, New Orleans, LA, USA, Apr. 2019.

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

[23] Guangrui Li, Guoliang Kang, Wu Liu, Yunchao Wei, and Yi Yang. Content-Consistent Matching for Domain Adaptive Semantic Segmentation. In *Proc. of ECCV*, pp. 1–18, Glasgow, UK, Aug. 2020.

[24] Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model Adaptation: Unsupervised Domain Adaptation Without Source Data. In *Proc. of CVPR*, pp. 9641–9650, Seattle, WA, USA, June 2020.

[25] Shunkai Li, Xin Wang, Yingdian Cao, Fei Xue, Zike Yan, and Hongbin Zha. Self-Supervised Deep Visual Odometry With Online Adaptation. In *Proc. of CVPR*, pp. 6339–6348, Seattle, WA, USA, June 2020.

[26] Xianfeng Li, Weijie Chen, Di Xie, Shicai Yang, Peng Yuan, Shiliang Pu, and Yueting Zhuang. A Free Lunch for Unsupervised Domain Adaptive Object Detection Without Source Data. *arXiv:2012.05400*, Dec. 2020.

[27] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional Learning for Domain Adaptation of Semantic Segmentation. In *Proc. of CVPR*, pp. 6936–6945, Long Beach, CA, USA, June 2019.

[28] Qing Lian, Fengmao Lv, Lixin Duan, and Boqing Gong. Constructing Self-Motivated Pyramid Curriculums for Cross-Domain Semantic Segmentation: A Non-Adversarial Approach. In *Proc. of ICCV*, pp. 6758–6767, Seoul, Korea, Oct. 2019.

[29] Yoseph Linde, Andrés Buzo, and Robert Gray. An Algorithm for Vector Quantizer Design. *IEEE Trans. on Communications*, 28(1):84–95, Jan. 1980.

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

[31] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a Closer Look at Domain Shift: Category-Level Adversaries for Semantics Consistent Domain Adaptation. In *Proc. of CVPR*, pp. 2507–2516, Long Beach, CA, USA, June 2019.

[32] Fei Pan, Inkyu Shin, Francois Rameau, Seokju Lee, and In So Kweon. Unsupervised Intra-Domain Adaptation for Semantic Segmentation Through Self-Supervision. In *Proc. of CVPR*, pp. 3764–3773, Seattle, WA, USA, June 2020.

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

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

[35] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In *Proc. of CVPR*, pp. 3234–3243, Las Vegas, NV, USA, June 2016.

[36] Mark Schutera, Hussein Mostafa, Jochen Abhau, Ralf Mikut, and Markus Reischl. Night-to-Day: Online Image-to-Image Translation for Object Detection Within Autonomous Driving by Night. *IEEE Transactions on Intelligent Vehicles*, Nov. 2020.

[37] Seonguk Seo, Yumin Suh, Dongwan Kim, Geeho Kim, Jongwoo Han, and Bohyung Han. Learning to Optimize Domain Specific Normalization for Domain Generalization. In *Proc. of ECCV*, pp. 68–83, Glasgow, UK, Aug. 2020.

[38] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proc. of ICLR*, pp. 1–27, San Diego, CA, USA, May 2015.

[39] Muhammad Naseer Subhani and Mohsen Ali. Learning from Scale-Invariant Examples for Domain Adaptation in Semantic Segmentation. In *Proc. of ECCV*, pp. 290–306, Glasgow, UK, Aug. 2020.

[40] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. DACS: Domain Adaptation via Cross-Domain Mixed Sampling. In *Proc. of WACV*, pp. 1379–1389, Virtual, Jan. 2021.

[41] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation. In *Proc. of CVPR*, pp. 2517–2526, Long Beach, CA, USA, June 2019.

[42] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. DADA: Depth-Aware Domain Adaptation in Semantic Segmentation. In *Proc. of ICCV*, pp. 7364–7373, Seoul, Korea, Oct. 2019.

[43] Zhonghao Wang, Mo Yu, Yunchao Wei, Rogerio Feris, Jinjun Xiong, Wen-Mei Hwu, Thomas S. Huang, and Honghui Shi. Differential Treatment for Stuff and Things: A Simple Unsupervised Domain Adaptation Method for Semantic Segmentation. In *Proc. of CVPR*, pp. 12635–12644, Seattle, WA, USA, June 2020.

[44] Zuxuan Wu, Xintong Han, Yen-Liang Lin, Mustafa Gokhan Uzunbas, Tom Goldstein, Ser Nam Lim, and Larry S. Davis. DCAN: Dual Channel-Wise Alignment Networks for Unsupervised Scene Adaptation. In *Proc. of ECCV*, pp. 535–552, Munich, Germany, Sept. 2018.

[45] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Incremental Adversarial Domain Adaptation for Continually Changing Environments. In *Proc. of ICRA*, pp. 4489–4495, Brisbane, Australia, May 2018.

[46] Jinyu Yang, Weizhi An, Sheng Wang, Xinliang Zhu, Chaochao Yan, and Junzhou Huang. Label-Driven Reconstruction for Domain Adaptation in Semantic Segmentation. In *Proc. of ECCV*, pp. 480–498, Glasgow, UK, Aug. 2020.

[47] Yanchao Yang, Dong Lao, Ganesh Sundaramoorthi, and Stefano Soatto. Phase Consistent Ecological Domain Adaptation. In *Proc. of CVPR*, pp. 9011–9020, Seattle, WA, USA, June 2020.

[48] Yanchao Yang and Stefano Soatto. FDA: Fourier Domain Adaptation for Semantic Segmentation. In *Proc. of CVPR*, pp. 4085–4095, Seattle, WA, USA, June 2020.

[49] Hao-Wei Yeh, Baoyao Yang, Pong C. Yuen, and Tatsuya Harada. SoFA: Source-Data-Free Feature Alignment for Unsupervised Domain Adaptation. In *Proc. of WACV*, pp. 474–483, Virtual, Jan. 2021.

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

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

[52] Qiming Zhang, Jing Zhang, Wei Liu, and Dacheng Tao. Category Anchor-Guided Unsupervised Domain Adaptation for Semantic Segmentation. In *Proc. of NeurIPS*, pp. 433–443, Vancouver, Canada, Dec. 2019.

[53] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *Proc. of ICCV*, pp. 2223–2232, Venice, Italy, Oct. 2017.