# Continual BatchNorm Adaptation (CBNA) for Semantic Segmentation

Marvin Klingner, *Graduate Student Member, IEEE*, Mouadh Ayache, and Tim Fingscheidt, *Senior Member, IEEE*

***Abstract*— Environment perception in autonomous driving vehicles often heavily relies on deep neural networks (DNNs), which are subject to domain shifts, leading to a significantly decreased performance during DNN deployment. Usually, this problem is addressed by unsupervised domain adaptation (UDA) approaches trained either simultaneously on source and target domain datasets or even source-free only on target data in an offline fashion. In this work, we further expand a source-free UDA approach to a continual and therefore online-capable UDA on a single-image basis for semantic segmentation. Accordingly, our method only requires the pre-trained model from the supplier (trained in the source domain) and the current (unlabeled target domain) camera image. Our method Continual BatchNorm Adaptation (CBNA) modifies the source domain statistics in the batch normalization layers, using target domain images in an unsupervised fashion, which yields consistent performance improvements during inference. Thereby, in contrast to existing works, our approach can be applied to improve a DNN continuously on a single-image basis during deployment without access to source data, without algorithmic delay, and nearly without computational overhead. We show the consistent effectiveness of our method across a wide variety of source/target domain settings for semantic segmentation. Code is available at https://github.com/ifnspaml/CBNA**

***Index Terms*— Domain adaptation, neural networks, deep learning, unsupervised learning, semantic segmentation, batch normalization.**

## I. Introduction

THE information processing concept of an autonomous driving vehicle as shown in Fig. 1 relies heavily on deep neural networks (DNNs) to extract information from sensor inputs such as camera images, RADAR measurements, or LiDAR point clouds. Exemplary tasks executed by such DNNs are semantic segmentation [1], [2], depth estimation [3], [4], instance segmentation [5], [6], or object detection [7], [8], which are expected to provide high-quality outputs for a safe operation of the vehicle. However, DNNs are usually trained on annotated datasets [9], [10], only covering a small portion of real-life scenery. However, when DNNs are deployed in the car, the environment can change drastically due to, e.g., different image appearances from a new camera or day/night shifts, leading to a significantly decreased DNN performance [11], [12]. This problem (known as domain shift [13]) needs to be addressed for a successful deployment of DNNs in highly automated vehicles.

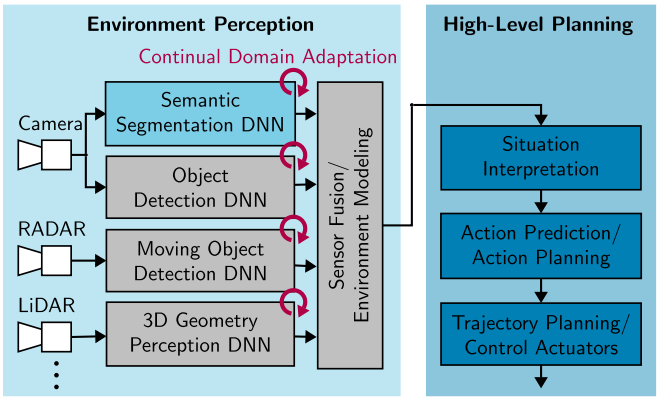



Fig. 1. Overview about **continual unsupervised domain adaptation**, e.g., by CBNA, and its deployment in online environment perception.

Focusing on the semantic segmentation task, two main concepts have been established to improve the performance in a real-world target domain that is unlabeled by nature. Firstly, in domain generalization (DG), the neural network is trained more robust on several different source domains to improve performance on unknown target domains [14], [15]. Here, the target domain is assumed to be unavailable and accordingly one cannot make use of specific target domain images. Secondly, in unsupervised domain adaptation (UDA), the model is trained simultaneously on the labeled source data and unlabeled target data, assuming that data from both domains is available at the same time [12], [16]–[18]. In practice, however, models are often trained on non-public datasets, which cannot be passed on due to data-privacy issues or for other practical reasons, meaning that neither source data nor representations thereof are available, instead only the trained model from the supplier is available for adaptation. In this case, DG as well as standard UDA techniques cannot be applied. Therefore, similar to [19], we focus on UDA without source data, meaning that we adapt a given trained model using only unlabeled target domain data.

In this work, we aim at a task which is even more challenging yet also more interesting for practical deployment: We focus on UDA *without access to source data*, where the DNN is adapted *during inference* for every single (target domain) image *in a continual fashion*, see the bottom part

Manuscript received 12 July 2021; revised 17 February 2022; accepted 6 July 2022. Date of publication 27 July 2022; date of current version 7 November 2022. The Associate Editor for this article was K. Wang. *(Corresponding author: Marvin Klingner.)*

The authors are with the Institute of Communications Technology, Technische Universität Braunschweig, 38106 Braunschweig, Germany (e-mail: m.klingner@tu-bs.de; m.ayache@tu-bs.de; t.fingscheidt@tu-bs.de).

Digital Object Identifier 10.1109/TITS.2022.3190263

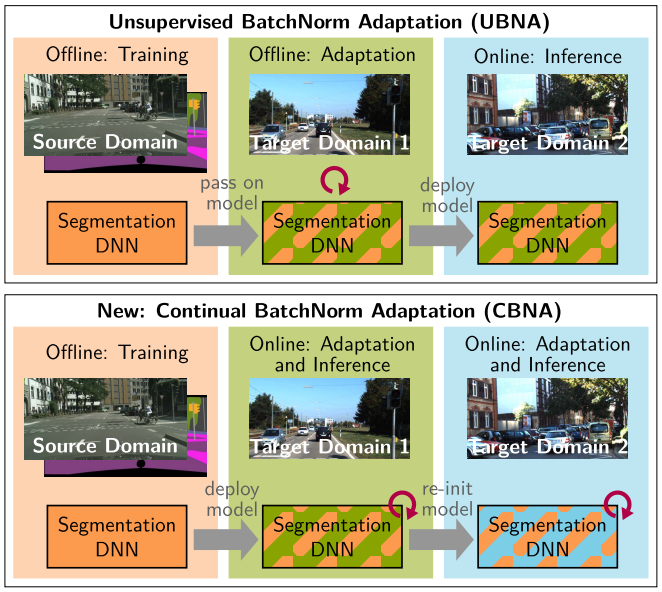



Fig. 2. Overview on **how our novel CBNA approach differs from** other source-free UDA approaches, e.g., **UBNA** [19]. Re-initialization and adaptation with CBNA is performed for *each* new image during inference.

of Fig. 2. Thereby, even if the domains switch rapidly in an image stream from a video (e.g., when driving into a tunnel), the network adapts to this on a single-image basis and can obtain optimal performance in each situation without any delay. Examples of such rapid domain changes could be a different camera illumination when driving into a tunnel or rapid weather/environmental changes, i.e., domain adaptation in adverse conditions [20], [21]. Previous standard UDA and DG approaches are of course inapplicable, as they require access to source data, which is usually unavailable in a highly automated vehicle due to storage limitations. Even our earlier Unsupervised BatchNorm Adaptation (UBNA) work [19] not relying on source data is not applicable to this task, as its adaptation takes place on a separate data subset from the target domain, see top part of Fig. 2. If during deployment the domain changes again, or even permanently before inference can take place, the performance of the model decreases as the adaptation is not applied in a continual fashion. In practice, however, it would be desirable to adapt and infer the DNN on a single image basis at once to optimally match each new domain without algorithmic delay.

To provide a solution for this defined task, we present our Continual BatchNorm Adaptation (CBNA) method (cf. bottom part of Fig. 2) as an extension of our previous work UBNA [19]. Here, we *mix the batch normalization (BN) statistics* (not the data!) of the source domain and a single target domain image in a continual fashion for each new image from the target domain. Thereby, during inference in a vehicle, we can adapt the deep neural network model instantaneously to each new image from the video stream of a camera, while previous approaches [15], [19], [22]–[24] adapt only once in an offline fashion to a single target domain using multiple uncorrelated images. Regarding computational complexity, CBNA introduces only little computational overhead on the forward pass during inference, while reference approaches reaching a similar performance [19], [24] would require a whole additional forward pass through the model. Note, that all aforementioned known approaches have been proposed for offline settings (cf. top part of Fig. 2), making a direct comparison to our new CBNA proposal unfair due to the here envisaged more constrained continual UDA setting (cf. bottom part of Fig. 2). Therefore, we report their performance in our framework under the same constraints as so-called *reference methods*, as no baseline approaches for continual UDA of semantic segmentation exist so far.

Our contributions with this work are as follows: Firstly, we present our online-capable CBNA method for continual UDA without source data of semantic segmentation models. Secondly, we show the successful applicability of CBNA on a single-image basis during inference, with only little computational overhead and no algorithmic delay being induced. Thirdly, we show the effectiveness of CBNA across a variety of source/target domain combinations, where we can even find hyperparameters which generalize across different target domains for a given segmentation model proving the practical applicability of CBNA. We will publish our code to facilitate further research on continual UDA without source data.

This work is structured as follows. In Section II we discuss related approaches. Afterwards, in Section III we introduce our CBNA method as well as reference methods, followed by our experimental setup in Section IV. We evaluate our method in Section V and finally conclude this work in Section VI.

## II. Related Work

We give an overview on related domain generalization (DG) and unsupervised domain adaptation (UDA) approaches. For UDA we particularly discuss approaches not relying on source data and approaches making use of normalization layers.

### A. Domain Generalization (DG)

The aim of DG methods [25]–[28] is to improve DNN performance in an unknown target domain using data from (several different) source domains. For semantic segmentation, several approaches have been proposed [14], [25], [29], e.g., Yue *et al.* [14] mix the style of synthetic images with real images, using auxiliary source domain datasets, thereby learning more domain-invariant features. While our CBNA method for continual source-free UDA is applied *after* pre-training and using only the pre-trained model and *target domain* data, DG is applied *during* pre-training on *source data* (usually with labels) and without target data. Thereby, if only a given trained model and unlabeled data from the target domain are available, DG methods cannot be applied, motivating the application of methods for continual UDA without source data. Here, we additionally provide experimental results, where a DNN is first trained using DG methods and afterwards adapted using our CBNA algorithm for source-free continual UDA, showing that both methods for both tasks can be combined.

### B. Unsupervised Domain Adaptation (UDA)

Standard UDA approaches assume that both labeled source data and unlabeled target data are available at the same time. Thereby, the domain transfer can be achieved in an offline fashion by using domain adaptation training techniques.

These techniques can be roughly divided into three subcategories: Firstly, *domain-adversarial training* [12], [30]–[37], can be applied, where domain-invariant features are learned by an additional discriminator (loss). Secondly, *style transfer* [17], [38]–[40], can be used to better match the appearance of source and target domain by image-to-image translation approaches. Thirdly, *self-training* can be employed, where pseudo-labels are used as an additional supervision signal in the target domain [17], [41]–[44]. As these approaches all require labeled source data to be available during the domain adaptation, they are not applicable when source data is not available, e.g., due to data privacy issues. If in this case an improvement in the target domain is still desired, UDA approaches not relying on source data have to be used instead.

### C. UDA Without Source Data

Towards continual adaptation of semantic segmentation models, it is desirable to remove the need for source data during the adaptation, as this is usually a large dataset or a non-available dataset on the car manufacturer side, which cannot be stored on a deployed vehicle. The approaches of [45], [46] employ an auxiliary network which has been trained in the source domain together with the segmentation model. This network replays source domain knowledge to the network during adaptation. Moreover, Stan *et al.* [47] and Termöhlen *et al.* [48] learn a source domain distribution, which is aligned with the target domain distribution during adaptation. These approaches do not make use of source data during the UDA. However, they still require an additional source domain representation for their approach (e.g., an additional network), which is usually also not available for a trained model.

Only few approaches exist for UDA of a *given trained* model relying only on unlabeled target domain data. Some initial approaches relying on training with pseudo labels [49], [50], alignment methods for the latent space distribution [50]–[52], or class-conditional generative adversarial networks [53] focus on simple tasks such as image classification or object detection. However, the aforementioned approaches do not address the semantic segmentation task, which we address in this work. For this task some very recent methods have been developed concurrently: Teja *et al.* [54] apply entropy minimization on the posterior and maximize the noise robustness of latent features. Kundu *et al.* [55] use self-training on pseudo labels. Liu *et al.* [56] also make use of this technique and in addition apply data-free knowledge distillation. Our main distinguishing aspect from these works is the proposal of an efficient continual domain adaptation on a single-image basis, while to the best of our knowledge all other source-free methods for semantic segmentation rely on a time-consuming second training stage on many images in the target domain.

### D. UDA via Normalization Layers

The initial works of Li *et al.* [22], [23] for image classification and Zhang *et al.* [15] for semantic segmentation show that the re-estimation of batch normalization (BN) statistics in the target domain can be used for UDA without source data. The UBNA method from Klingner *et al.* [19] has shown that mixing statistics from the source and target domain outperforms these initial works, which we build upon for our method design. These findings for domain adaptation are also supported by the work on adversarial robustness of Schneider *et al.* [24], where the beneficial effect of mixing statistics from perturbed and clean images is shown. However, the approaches mentioned before are only applicable to an offline UDA on a dataset, i.e., they still require statistics from multiple uncorrelated images in the target domain for a successful application. This is disadvantageous for continual UDA settings, where it would be desirable to continuously adapt on a single-image basis to avoid algorithmic delay during deployment in rapidly changing domains. In contrast to existing methods [15], [19], [22]–[24], our CBNA method is applicable to these continual UDA settings, which we will show by our successful single-image adaptation results without the usage of additional uncorrelated images. Another novelty of CBNA is its integration into the single-image inference forward pass of an already trained model, which introduces nearly no computational overhead during inference.

## III. BatchNorm Adaptation Methods

In this section we first revisit the batch normalization (BN) layer and thereby introduce notations. Afterwards, we provide reference methods for continual UDA of BN parameters during inference, which we derive from their originally published offline versions. Finally, we introduce our novel CBNA method.

### A. Revisiting the Batch Normalization Layer, Notations

As our adaptation method relies on the usage of batch normalization (BN) layers, we briefly revisit the BN operation for the scope of a fully convolutional DNN with two spatial dimensions following [57]. Each BN layer then processes a batch of input feature maps $\boldsymbol{f}_\ell \in \mathbb{R}^{B\times H_\ell \times W_\ell \times C_\ell}$ with batch size $B$, height $H_\ell$, width $W_\ell$, and number of channels $C_\ell$ of the feature map in the BN layer with index $\ell$. Then the normalization is given by

$$\hat{f}_{b,\ell,i,c} = \gamma_{\ell,c} \cdot (f_{b,\ell,i,c} - \mu_{\ell,c}) \cdot \left(\sigma^2_{\ell,c} + \epsilon\right)^{-\frac{1}{2}} + \beta_{\ell,c}\ , \quad (1)$$

where each feature $f_{b,\ell,i,c} \in \mathbb{R}$ is normalized over the batch and spatial dimensions with indices $b \in \mathcal{B} = \{1,\dots,B\}$ and $i \in \mathcal{I}_\ell = \{1,\dots,H_\ell \cdot W_\ell\}$, respectively, on a channel-wise basis (channel index $c \in \mathcal{C}_\ell = \{1,\dots,C_\ell\}$), yielding the normalized output $\hat{\boldsymbol{f}}_\ell$. In (1), $\boldsymbol{\mu}_\ell = (\mu_{\ell,c}) \in \mathbb{R}^{C_\ell}$ and $\boldsymbol{\sigma}_\ell = (\sigma_{\ell,c}) \in \mathbb{R}^{C_\ell}_+$ are the channel-wise computed mean and standard deviations in layer $\ell$, respectively, while $\boldsymbol{\gamma}_\ell = (\gamma_{\ell,c}) \in \mathbb{R}^{C_\ell}$ and $\boldsymbol{\beta}_\ell = (\beta_{\ell,c}) \in \mathbb{R}^{C_\ell}$ are learnable scaling and shifting parameters. The constant $\epsilon > 0$ is a small number avoiding divisions by zero.

During learning step $K$ in training, the mean vector $\check{\boldsymbol{\mu}}^{(K)}_\ell = (\check{\mu}^{(K)}_{\ell,c})$ and standard deviation vector $\check{\boldsymbol{\sigma}}^{(K)}_\ell = (\check{\sigma}^{(K)}_{\ell,c})$ of the features $\boldsymbol{f}_\ell$ from the current batch $\mathcal{B}$ are calculated as

$$\check{\mu}^{(K)}_{\ell,c} = \frac{1}{BH_\ell W_\ell} \sum_{b\in\mathcal{B}} \sum_{i\in\mathcal{I}_\ell} f_{b,\ell,i,c}, \quad (2)$$

$$\left(\check{\sigma}^{(K)}_{\ell,c}\right)^2 = \frac{1}{BH_\ell W_\ell} \sum_{b\in\mathcal{B}} \sum_{i\in\mathcal{I}_\ell} \left(f_{b,\ell,i,c} - \check{\mu}^{(K)}_{\ell,c}\right)^2. \quad (3)$$

During training, these values are directly used for the forward pass computation in (1), i.e., $\boldsymbol{\mu}_\ell = \check{\boldsymbol{\mu}}_\ell^{(K)}$ and $\boldsymbol{\sigma}_\ell = \check{\boldsymbol{\sigma}}_\ell^{(K)}$. However, during inference, one does not desire a normalization over the batch dimension, as this would make the output of the DNN on one image dependent on the other images in the batch, inducing indeterministic performance. Therefore, as preparation for inference, the BN statistics of the entire training dataset is approximated by recursively tracking mean and variance from (2) and (3) as

$$\check{\mu}_{\ell,c}^{(K)} = (1-\eta)\cdot\check{\mu}_{\ell,c}^{(K-1)} + \eta\cdot\check{\mu}_{\ell,c}^{(K)}, \tag{4}$$

$$\left(\check{\sigma}_{\ell,c}^{(K)}\right)^2 = (1-\eta)\cdot\left(\check{\sigma}_{\ell,c}^{(K-1)}\right)^2 + \eta\cdot\left(\check{\sigma}_{\ell,c}^{(K)}\right)^2, \tag{5}$$

using a momentum parameter $\eta \in [0,1]$. The final values from (4) and (5) after $K$ learning steps are then stored and used later for inference, i.e., in (1) we employ $\boldsymbol{\mu}_\ell = \check{\boldsymbol{\mu}}_\ell^{(K)}$ and $\boldsymbol{\sigma}_\ell = \check{\boldsymbol{\sigma}}_\ell^{(K)}$.

### *B. Continuous Adaptation Reference Methods (**C-X**)*

To improve the semantic segmentation DNN's performance during inference, we aim at adapting to each single image $\boldsymbol{x}_t^{\mathcal{D}^\mathrm{T}}$ from the target domain $\mathcal{D}^\mathrm{T}$ from a video at time $t$, implementing a continual UDA. We assume that besides the input image $\boldsymbol{x}_t^{\mathcal{D}^\mathrm{T}}$ only the trained model parameters from the source domain are available for this purpose. For semantic segmentation, there are no baseline methods known for this task, however, we still want to allow a comparison to previous works and therefore we modify several approaches to fit into our defined task, which then serve as reference approaches **C-X** to our CBNA method.

The first such reference is a version of the AdaBN approach from Li *et al.* [22], [23], who replace the source domain's BN statistics $\check{\boldsymbol{\mu}}_\ell^{(K)}$, $\check{\boldsymbol{\sigma}}_\ell^{(K)}$ by the target domain's BN statistics during inference. Originally, Li *et al.* employ the statistics from all uncorrelated images of the test set in the computation. This, however, is not suitable for our single-image continual UDA task and would incur a large algorithmic delay of the method. Therefore, to fit into our task definition, we modify AdaBN [22], [23] as follows: In (1), the BN mean $\mu_{\ell,c}$ and variance $\sigma_{\ell,c}^2$ of each layer $\ell$ and channel $c$ are set individually for each target domain image $\boldsymbol{x}_t^{\mathcal{D}^\mathrm{T}}$ during inference, i.e., $\mu_{\ell,c} \equiv \mu_{t,\ell,c}$ and $\sigma_{\ell,c}^2 \equiv \sigma_{t,\ell,c}^2$. They are calculated as

$$\mu_{t,\ell,c} = \check{\mu}_{t,\ell,c} \;\text{ and }\; \sigma_{t,\ell,c}^2 = \check{\sigma}_{t,\ell,c}^2, \tag{6}$$

where $\check{\mu}_{t,\ell,c}$ and $\check{\sigma}_{t,\ell,c}^2$ are computed according to (2) and (3), respectively, using only a batch size of $B=1$, which is only the available single-image input $\boldsymbol{x}_t^{\mathcal{D}^\mathrm{T}}$. We dub this method **C-Li**, "continuous Li", noting that this procedure requires only a single forward pass during inference, as (6) can be computed during the inference forward pass.[1] Interestingly, the approach from Zhang *et al.* [15] reduces to the same formulation, if only a single target-domain image is used for adaptation during inference. We dub it **C-Zhang**.

The second reference method is derived from the UBNA approach of Klingner *et al.* [19] (**C-Klingner**), which adapts a model on a separate adaptation set by mixing the source domain BN statistics with the target domain BN statistics. While they do this using 50 adaptation steps, a separate adaptation to each single image with 50 additional forward passes may cause too much computational overhead for deployment of the method in a vehicle. However, it can be shown that in the limit of using the same single adaptation image in all 50 adaptation steps, UBNA can be reduced to a single *additional* forward pass. On the first forward pass, the statistics $\check{\boldsymbol{\mu}}_{t,\ell}$, $\check{\boldsymbol{\sigma}}_{t,\ell}$ of the target domain image $\boldsymbol{x}_t^{\mathcal{D}^\mathrm{T}}$ are determined as in C-Li. Afterwards, before the second forward pass with the same image, the image-specific BN statistics $\boldsymbol{\mu}_{t,\ell}$, $\boldsymbol{\sigma}_{t,\ell}$ used during inference are updated element-wise as:

$$\mu_{t,\ell,c} = \left(1-\eta^{\mathcal{D}^\mathrm{S}}\right)\cdot\check{\mu}_{\ell,c}^{(K)} + \eta^{\mathcal{D}^\mathrm{S}}\cdot\check{\mu}_{t,\ell,c}, \tag{7}$$

$$\sigma_{t,\ell,c}^2 = \left(1-\eta^{\mathcal{D}^\mathrm{S}}\right)\cdot\left(\check{\sigma}_{\ell,c}^{(K)}\right)^2 + \eta^{\mathcal{D}^\mathrm{S}}\cdot\check{\sigma}_{t,\ell,c}^2, \tag{8}$$

by additionally considering the source-domain statistics $\check{\boldsymbol{\mu}}_\ell^{(K)}$, $\check{\boldsymbol{\sigma}}_\ell^{(K)}$ (obtained from (4) and (5) after $K$ training steps), which were disregarded in C-Li. The mixing weight $\eta^{\mathcal{D}^\mathrm{S}} \in [0,1]$ is used to weigh the influence of the target domain statistics. Interestingly, the same formulation can be derived from the method of [24], although they use a different hyperparameter formulation for $\eta^{\mathcal{D}^\mathrm{S}}$ and apply their method to improve adversarial robustness. While the mixing of source and target-domain statistics in C-Klingner by (7) and (8) is shown to be beneficial for performance, it also induces a second forward pass, which is disadvantageous in terms of computational complexity.

**Algorithm 1** Model Adaptation and Inference With **CBNA**

1: Load segmentation model trained on source data, including the source domain's BN statistics $\check{\boldsymbol{\mu}}_\ell^{(K)}$, $\check{\boldsymbol{\sigma}}_\ell^{(K)}$ as trained in $K$ steps of (4), (5)
2: Take current image $\boldsymbol{x}_t^{\mathcal{D}^\mathrm{T}}$ from the target domain $\mathcal{D}^\mathrm{T}$
3: **CBNA**: Initialize BN momentum $\eta^{\mathcal{D}^\mathrm{S}}$ for all BN layers $\ell$
4: Pass image $\boldsymbol{x}_t^{\mathcal{D}^\mathrm{T}}$ through the model until the first BN layer
5: **for** BN layer $\ell \in \{1,\ldots,L\}$ **do**
6: **CBNA**: Calculate BN statistics according to (9), (10)
7: **CBNA**: Update BN statistics according to (11), (12)
8: Pass features through the BN layer according to (1)
9: Pass features further until the next BN layer $\ell+1$
10: **end for**
11: Pass features up to the end and generate the output $\boldsymbol{y}_t^{\mathcal{D}^\mathrm{T}}$

### *C. Novel Continuous BatchNorm Adaptation (CBNA)*

While both presented reference methods C-X come with the mentioned disadvantages, our **CBNA** method is able to mix BN statistics $\check{\boldsymbol{\mu}}_\ell^{(K)}$, $\check{\boldsymbol{\sigma}}_\ell^{(K)}$ from the source domain and the statistics of a single target-domain image $\boldsymbol{x}_t^{\mathcal{D}^\mathrm{T}}$ during a *single* inference forward pass as shown in Fig. 3. Before the features $\boldsymbol{f}_{t,\ell}$ are normalized, their statistics are calculated and mixed with the stored source domain statistics. The mixed statistics are subsequently used to normalize the features. This method, also described by Algorithm 1, is simply executed as one

[1] This is essentially the same (efficient) computation which is also carried out during training of the BN layer with a batch size of $B=1$.

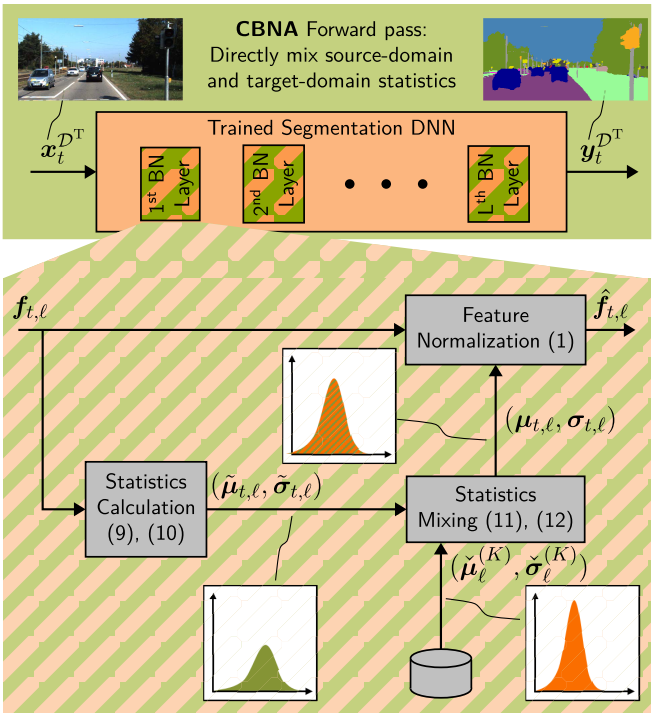



Fig. 3. Overview on how our **novel CBNA** approach mixes source and target domain BN statistics on a single forward pass. The color code shows, whether the network parts are optimized using information from the target-domain image (green) or the source-domain data (orange).

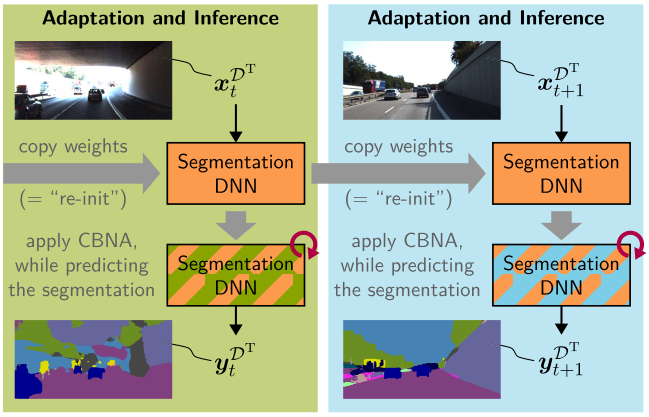



Fig. 4. **Online adaptation and inference setup** of our **CBNA** method during deployment. Shown is a detail of Fig. 2 (bottom).

forward pass for each new target domain image $\boldsymbol{x}_t^{\mathcal{D}^\mathrm{T}}$. Note that in contrast to previous source-free UDA methods [54]–[56] our CBNA method is a *continual* source-free UDA method (cf. Fig. 2).

In contrast to C-Li, our CBNA method mixes source and target domain statistics, which is beneficial for performance and stability (cf. Table II). In contrast to C-Klingner, the mixing of source and target domain statistics is done in a single forward pass, which significantly reduces the additional computational complexity introduced through the continual adaptation (cf. Table III).

The details of our proposed CBNA are as follows. We initialize by imposing a weighting factor $\eta^{\mathcal{D}^\mathrm{S}}$ between source and target domain. This factor has to be chosen w.r.t. the source domain model and should not differ for different target domains as the information about the target domain is only available during deployment and cannot always be known in advance. Notably, target domain information is, however, required for all previously proposed (offline) methods [15], [19], [22], [23].

During the *single* inference forward pass, CBNA is applied, while the image $\boldsymbol{x}_t^{\mathcal{D}^\mathrm{T}}$ is processed by the segmentation DNN. When the feature processing in the DNN reaches BN layer $\ell$, we first compute the layer's image-specific BN statistics as

$$\tilde{\mu}_{t,\ell,c} = \frac{1}{H_\ell W_\ell} \sum_{i \in \mathcal{I}_\ell} f_{t,\ell,i,c}, \quad c \in \mathcal{C}_\ell, \tag{9}$$

$$\tilde{\sigma}^2_{t,\ell,c} = \frac{1}{H_\ell W_\ell} \sum_{i \in \mathcal{I}_\ell} \left(f_{t,\ell,i,c} - \tilde{\mu}_{t,\ell,c}\right)^2, \quad c \in \mathcal{C}_\ell, \tag{10}$$

where the statistics are not only computed in dependency of the target domain image's statistics as in C-Li and C-Zhang and in the first forward pass of C-Klingner, but in dependency of the mixed statistics $\boldsymbol{\mu}_{t,\lambda}$ and $\boldsymbol{\sigma}_{t,\lambda}$ from all previous BN layers $\lambda \in \{1, 2, \ldots, \ell - 1\}$ (cf. Fig 3), the $\ell$-th BN layer features depend upon.[2] Consequently, these image-specific statistics $\tilde{\boldsymbol{\mu}}_{t,\ell}$ and $\tilde{\boldsymbol{\sigma}}_{t,\ell}$ from (9) and (10), respectively, are *applied immediately* to update the BN statistics used for normalization of the features $\boldsymbol{f}_{t,\ell}$ in BN layer $\ell$ as

$$\mu_{t,\ell,c} = \left(1 - \eta^{\mathcal{D}^\mathrm{S}}\right) \cdot \check{\mu}^{(K)}_{\ell,c} + \eta^{\mathcal{D}^\mathrm{S}} \cdot \tilde{\mu}_{t,\ell,c}, \quad c \in \mathcal{C}_\ell, \tag{11}$$

$$\sigma^2_{t,\ell,c} = \left(1 - \eta^{\mathcal{D}^\mathrm{S}}\right) \cdot \left(\check{\sigma}^{(K)}_{\ell,c}\right)^2 + \eta^{\mathcal{D}^\mathrm{S}} \cdot \tilde{\sigma}^2_{t,\ell,c}, \quad c \in \mathcal{C}_\ell, \tag{12}$$

Finally, the features are normalized according to (1) using the statistics from (11) and (12) and processed further until the next BN layer $\ell + 1$. This procedure is repeated progressively through all BN layers of the model until the segmentation mask $\boldsymbol{y}_t^{\mathcal{D}^\mathrm{T}}$ has been generated (cf. Algorithm 1). Note that the application of CBNA does only involve a single inference forward pass through the model with minimal computational overhead for computing (9) and (10) in each BN layer, and for updating the statistics in (11) and (12) in each BN layer, which presents a strong advantage over the C-Klingner reference method.

During deployment, our CBNA method can be used for DNN adaptation during the inference forward pass of each individual image $\boldsymbol{x}_t^{\mathcal{D}^\mathrm{T}}$ of a video as shown in Fig. 4. Thereby, at time index $t$, the pre-trained model's BN statistics $\check{\boldsymbol{\mu}}^{(K)}_\ell$, $\check{\boldsymbol{\sigma}}^{(K)}_\ell$ are adapted to the current (target domain) image $\boldsymbol{x}_t^{\mathcal{D}^\mathrm{T}}$ by CBNA, as detailed in Algorithm 1. For the next image $\boldsymbol{x}_{t+1}^{\mathcal{D}^\mathrm{T}}$, again the pre-trained model from the source domain (with BN statistics $\check{\boldsymbol{\mu}}^{(K)}_\ell$, $\check{\boldsymbol{\sigma}}^{(K)}_\ell$) is used as re-initialization before the adaptation with CBNA during the inference forward pass. Thereby, CBNA can be applied in a continual fashion with very little computational overhead during deployment of a semantic segmentation DNN in a vehicle.

[2]For the first layer ($\ell = 1$), there is obviously no previous BN layer and therefore also no dependency of its statistics.

TABLE I
AVAILABLE DATABASES AND THEIR CORRESPONDING NUMBER OF IMAGES USED FOR TRAINING AND FOR EVALUATION

| Dataset | Domain | pre-training | adaptation & validation | adaptation & test |
|---|---|---|---|---|
| GTA-5 [58] | $\mathcal{D}^{\mathrm{S}}$ | 24,966 | - | - |
| SYNTHIA [59] | $\mathcal{D}^{\mathrm{S}}$ | 9,400 | - | - |
| Cityscapes [10] | $\mathcal{D}^{\mathrm{S}}$ | 2,975 | - | - |
| KITTI [9], [60] | $\mathcal{D}^{\mathrm{S}}$ | 200 | - | - |
| Cityscapes [10] | $\mathcal{D}^{\mathrm{T}}$ | - | 500 | 2,975 |
| KITTI [9], [60] | $\mathcal{D}^{\mathrm{T}}$ | - | 200 | - |
| BDD [61] | $\mathcal{D}^{\mathrm{T}}$ | - | 1,000 | - |
| Mapillary [62] | $\mathcal{D}^{\mathrm{T}}$ | - | 2,000 | - |

## IV. EXPERIMENTAL AND EVALUATION SETUP

In this section, we first describe our used databases. Then, we explain our training procedures resulting in the given models for adaptation. Finally, we introduce evaluation metrics.

### A. Databases

We carry out experiments across a variety of datasets used for training the given models (top part of Tab. I) and for evaluation of the CBNA method (bottom part of Tab. I). Our main experiments use pre-trained models from the synthetic datasets GTA-5 ($\mathcal{D}^{\mathrm{S}}$) [58] and SYNTHIA ($\mathcal{D}^{\mathrm{S}}$) [59], which are commonly used in other UDA works [12], [16]–[18]. To show the applicability of CBNA to real-to-real adaptation settings we alternatively use the real dataset Cityscapes ($\mathcal{D}^{\mathrm{S}}$) [10] for training. The real dataset KITTI ($\mathcal{D}^{\mathrm{S}}$) [9] utilizing the 200 training images from the KITTI 2015 dataset [60] is used as additional source-domain training material throughout the later described domain generalization (DG-Init) experiments.

Although CBNA is meant to be applied to image sequences (i.e., a video) during deployment, there are no well-established video benchmarks for UDA of semantic segmentation. However, as CBNA and all C-X reference methods are applicable on a single-image basis, the evaluation can be carried out equivalently on single images of a validation/test set containing uncorrelated images. For our main experiments (based on GTA-5 and SYNTHIA training) we use the target domain Cityscapes ($\mathcal{D}^{\mathrm{T}}$) with 500 validation images to optimize our method's hyperparameters and 2,975 test images (official training images) to show their generalizability. Note that we use the official Cityscapes training images in our test set, as the official test set has no publicly available labels. Moreover, we use the target domains KITTI ($\mathcal{D}^{\mathrm{T}}$) [9], [60], BDD ($\mathcal{D}^{\mathrm{T}}$) [61], and Mapillary ($\mathcal{D}^{\mathrm{T}}$) [62] (further details in Appendix A) during ablation experiments. Whenever the domains Cityscapes ($\mathcal{D}^{\mathrm{S}}$) or KITTI ($\mathcal{D}^{\mathrm{S}}$) have been used during training, we do not employ the respective target domains during evaluation.

### B. Training of the "Given" Source-Domain Models

We use the same network architecture as in [19] relying on the widely used `VGG` [63] and `ResNet` [5] network architectures (further details in Appendix B). The input to the network is an RGB image $\boldsymbol{x}_t^{\mathcal{D}^{\mathrm{S}}} \in \mathbb{I}^{H\times W\times C}$ from the source domain $\mathcal{D}^{\mathrm{S}}$ with height $H$, width $W$, and number of channels $C = 3$. The image is normalized to the range $\mathbb{I} = [0, 1]$. The network predicts a posterior probability tensor $\boldsymbol{y}_t^{\mathcal{D}^{\mathrm{S}}} = (y_{t,i,s}^{\mathcal{D}^{\mathrm{S}}}) \in \mathbb{I}^{H\times W\times|\mathcal{S}|}$, where $y_{t,i,s}^{\mathcal{D}^{\mathrm{S}}}$ is the probability that a pixel $\boldsymbol{x}_{t,i}^{\mathcal{D}^{\mathrm{S}}} \in \mathbb{I}^C$ with $i \in \mathcal{I} = \{1, \ldots, H \cdot W\}$ belongs to class $s \in \mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$. For simplicity, we henceforth set $\boldsymbol{x}_t^{\mathcal{D}^{\mathrm{S}}} = \boldsymbol{x}_t$ and $\boldsymbol{y}_t^{\mathcal{D}^{\mathrm{S}}} = \boldsymbol{y}_t$ in this section. During inference, the final class can be determined through $m_{t,i} = \mathrm{argmax}_{s\in\mathcal{S}} y_{t,i,s}$, yielding a pixel-wise semantic segmentation map $\boldsymbol{m}_t = (m_{t,i}) \in \mathcal{S}^{H\times W}$. During training, the network is optimized using ground truth labels $\overline{\boldsymbol{m}}_t \in \mathcal{S}^{H\times W}$, which are one-hot encoded such that $\overline{m}_{t,i} = \mathrm{argmax}_{s\in\mathcal{S}} \overline{y}_{t,i,s}$, yielding a ground truth tensor $\overline{\boldsymbol{y}}_t = (\overline{y}_{t,i,s}) = \{0, 1\}^{H\times W\times|\mathcal{S}|}$. For optimization, we use the weighted cross-entropy loss

$$J_t^{\mathrm{seg}} = -\frac{1}{H \cdot W} \sum_{i\in\mathcal{I}} \sum_{s\in\mathcal{S}} w_s \overline{y}_{t,i,s} \cdot \log\left(y_{t,i,s}\right), \qquad (13)$$

where the class-wise weights $w_s$ are determined as in [64].

During optimization with (13) as loss function, we resize the images from GTA-5, SYNTHIA, Cityscapes, and KITTI to resolutions of $1024 \times 576$, $1024 \times 608$, $1024 \times 512$, and $1024 \times 320$, respectively. Subsequently, these resized images are randomly cropped to a resolution of $640 \times 192$. As data augmentations we use horizontal flipping, random brightness ($\pm 0.2$), contrast ($\pm 0.2$), saturation ($\pm 0.2$), and hue ($\pm 0.1$). We optimize our segmentation models for 20 epochs ($10,000$ training steps approximately comprise one epoch), using the Adam optimizer [65] and a batch size of $B = 12$ if we only use a single dataset. As a simple DG method we use mixed batches from two datasets (6 images from each dataset), which we mark by (DG-Init) during evaluation. This may not be the latest state-of-the-art DG method, however, the scope of this work is not to optimize a DG method but merely to show that CBNA can be applied to given models that were trained by DG methods. The learning rate is initially set to $10^{-4}$ and reduced to $10^{-5}$ for the last 5 epochs.

### C. Evaluation Metrics

We evaluate the semantic segmentation output by calculating the mean intersection-over-union (mIoU) [66]

$$\mathrm{mIoU} = \frac{1}{|\mathcal{S}|} \sum_{s\in\mathcal{S}} \mathrm{IoU}_s = \frac{1}{|\mathcal{S}|} \sum_{s\in\mathcal{S}} \frac{\mathrm{TP}_s}{\mathrm{TP}_s + \mathrm{FP}_s + \mathrm{FN}_s} \qquad (14)$$

over all $|\mathcal{S}| = 19$ classes as defined in [10], except for models trained on SYNTHIA, where we follow common practices [67], [68] in evaluating over subsets of 13 and 16 classes. For each class $s$ the number of true positives ($\mathrm{TP}_s$), false negatives ($\mathrm{FN}_s$), and false positives ($\mathrm{FP}_s$), calculated between predictions $\boldsymbol{m}_t$ and ground truths $\overline{\boldsymbol{m}}_t$, are accumulated over all $T$ images of the validation/test set. For adaptation and evaluation (with (14) being used as metric) we resize the images from Cityscapes, KITTI, BDD, and Mapillary to resolutions of $1024\times512$, $1024\times320$, $1024\times576$, and $1024 \times 576$, respectively.

## V. EXPERIMENTAL EVALUATION

To evaluate our method, we first give an ablation on how the hyperparameters of CBNA influence the method's

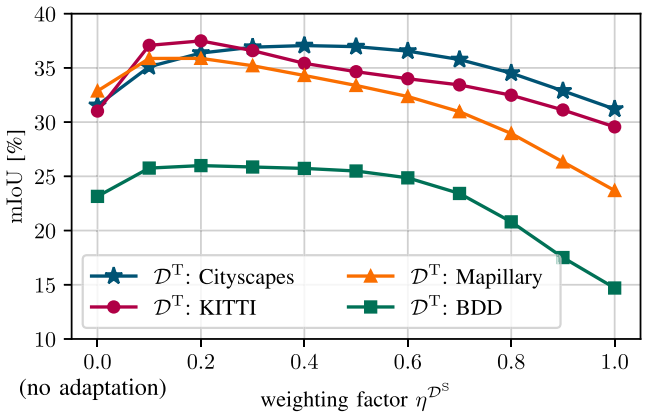



Fig. 5. **CBNA**: Influence of the weighting factor $\eta^{\mathcal{D}^S}$ (11), (12) for the `VGG-16`-based model, when performing an adaptation from GTA-5 ($\mathcal{D}^S$) to **several target domain validation sets** (cf. Table I).

performance. The final chosen model is then compared to several re-implemented reference methods (C-X) and to the only offline-capable UBNA from [19]. Finally, we compare our method on standard UDA benchmarks and give some qualitative results.

### *A. CBNA Method Design and Ablation*

When applying CBNA, first the question arises on how to weigh the influence of the source domain statistics and the statistics of the target domain image in (11) and (12), which is determined by the weighting factor $\eta^{\mathcal{D}^S}$. The analysis in Fig. 5 shows this influence for a `VGG-16`-based model being adapted from GTA-5 to several target domains, where $\eta^{\mathcal{D}^S} = 0$ represents using only the source domain statistics (no adaptation), and $\eta^{\mathcal{D}^S} = 1$ represents using only the target domain image's statistics (i.e., the C-Li method). We observe that the mIoU performance can be improved by approximately $3\% \ldots 5\%$ absolute (depending on the target domain) when mixing source and target domain statistics. However, if the influence from the target domain becomes too large, the performance decreases again. This is expected to some degree, as a high weight on the target domain image's BN statistics means that the network is strongly influenced by presumably rather unstable (i.e., highly time-variant) statistics of just a single image, compared to the statistics of many images from the source domain.

For the considered source domain model in Fig. 5, different optimal weightings $\eta^{\mathcal{D}^S}$ would be obtained for different target domains. However, in practice, we cannot assume prior knowledge about the target domain. Accordingly, each source domain model should only use a single weighting factor for all considered target domains and for any considered target image in general. Therefore, we choose the following strategy to obtain just a single weighting for all considered target domains: From the set $\mathcal{E} = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ we first take the best weighting for each target domain validation set. Then we average the target domain-specific weighting factors. Finally, we round to the next best weighting factor from $\mathcal{E}$. By this strategy we obtain a weighting factor of $\eta^{\mathcal{D}^S} = 0.2$ for models trained on GTA-5 and SYNTHIA, and $\eta^{\mathcal{D}^S} = 0.1$ for models

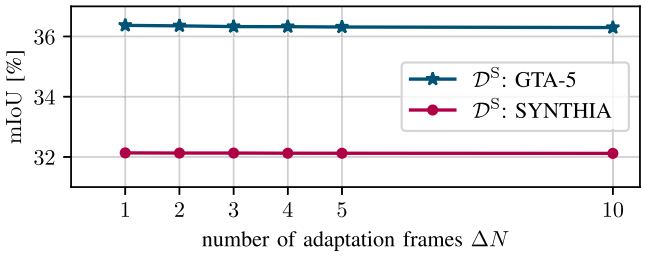



Fig. 6. **CBNA**: Influence of considering $\Delta N - 1$ preceding video frames for the `VGG-16`-based model, when performing an adaptation from **several source domains** to the Cityscapes ($\mathcal{D}^T$) validation set.

trained on Cityscapes. In the following experiments we will use these weighting factors for all experiments. To ensure fairness, we optimize the reference method C-Klingner in the same fashion, while the reference methods C-Li and C-Zhang do not contain such hyperparameters and therefore do not need to be optimized.

It is further of interest, whether the performance can be improved by considering the BN statistics from additional target domain images, which is investigated in Fig. 6. Here, for each sample, we additionally consider preceding image frames from its corresponding video (available in Cityscapes ($\mathcal{D}^T$)). Interestingly, there is no gain in performance, indicating that the statistics of a single target domain image in combination with the source domain statistics is already sufficient for a stable adaptation. One could argue that this behavior could also be expected due to the high correlation between consecutive images. However, side experiments show that using random uncorrelated frames from the target domain yields essentially the same behavior as observed for highly correlated preceding video frames in Fig. 6. Interestingly, this is not in contrast to [19], where additional images improved the adaptation performance, as in [19] a single offline adaptation was used to adapt to the entire target domain (which can of course be better represented by statistics from several images), whereas CBNA adapts to each single image separately. Accordingly, we can draw the conclusion that for the scope of our method the adaptation can be done on a single image basis during inference, which is a huge advantage in terms of applicability and latency (reaction time to domain shift).

### *B. Comparison to Reference Continual UDA Methods*

After having found suitable method hyperparameters, we facilitate a comparison to other possible approaches. As no continual UDA approaches for semantic segmentation exist so far, we reimplemented current related approaches and transferred them to our continual setting as described in Sec. III-B. We compare the results to our CBNA method in Table II for `VGG-16`-based and `ResNet-50`-based network architectures, with the adaptations from GTA-5 to Cityscapes and SYNTHIA to Cityscapes. The hyperparameters of CBNA and C-Klingner have been optimized for applicability to many target domains, as described in Section V-A on the validation sets of all target domains (cf. Table I). We test their generalization to unseen data by utilizing the official Cityscapes training set as our test set. In Table II we observe that just using the target domain statistics (C-Li and C-Zhang) does

TABLE II

PERFORMANCE COMPARISON OF CBNA TO RE-SIMULATED METHODS MODIFIED TO BECOME CONTINUAL SINGLE-IMAGE SOURCE-FREE UDA REFERENCE METHODS. RESULTS ARE REPORTED ON CITYSCAPES($\mathcal{D}^{\mathrm{T}}$)

| | Method | forward passes | $\mathcal{D}^{\mathrm{S}}$: SYNTHIA mIoU (%) (16 classes) | | $\mathcal{D}^{\mathrm{S}}$: GTA-5 mIoU (%) (19 classes) | |
|---|---|---|---|---|---|---|
| | | | validation | test | validation | test |
| ResNet-50 | No adaptation | 1 | 29.2 | 30.0 | 33.6 | 35.0 |
| | **C-Li** (∼[23]) | 1 | 28.8 | 28.9 | 34.3 | 35.6 |
| | **C-Zhang** (∼[15]) | 1 | 28.8 | 28.9 | 34.3 | 35.6 |
| | **C-Klingner** (∼[19]) | 2 | **32.7** | **33.4** | **37.3** | **38.8** |
| | **CBNA** | 1 | 32.5 | 33.2 | 36.7 | 38.3 |
| VGG-16 | No adaptation | 1 | 30.0 | 30.5 | 31.5 | 33.6 |
| | **C-Li** (∼[23]) | 1 | 28.8 | 28.9 | 31.2 | 33.3 |
| | **C-Zhang** (∼[15]) | 1 | 28.8 | 28.9 | 31.2 | 33.3 |
| | **C-Klingner** (∼[19]) | 2 | **33.4** | **33.7** | **36.7** | **39.0** |
| | **CBNA** | 1 | 32.1 | 32.4 | 36.4 | 38.9 |

not improve the results significantly and even reduces the performance in some cases, e.g., for the adaptation of both architectures from SYNTHIA to Cityscapes. This is consistent with the observations from Fig. 5, where $\eta^{\mathcal{D}^{\mathrm{S}}} = 1$ decreased the performance in all cases.

The reference method C-Klingner, mixing source and target domain statistics, improves significantly over the "no adaptation" baseline, however, it involves the execution of a second forward pass, adding a lot of computations during inference (cf. Table III). A detailed analysis is given in the Appendix. In total, our CBNA method always performs second-ranked, close after C-Klingner, requiring only a single forward pass with almost no computational overhead. On the used Tesla P100 graphics card, the VGG-16-based architecture with CBNA can be executed at 20fps (same as the "no adaptation" baseline), while C-Klingner reaches only 10fps. Compared to the source domain model, our CBNA method yields 3.2% and 3.3% absolute mIoU improvement for the adaptations from SYNTHIA to Cityscapes and GTA-5 to Cityscapes (test set), respectively, when applied to the ResNet-50-based model. Consistent improvements are also achieved for VGG-16-based models in these adaptation settings.

### C. Comparison to Offline Methods

To better understand the advantages that our method offers, we also compare to the offline-capable UBNA method from [19] in Tables IV and V. Notably, UBNA is applied on 50 random images of the target domain, meaning that in order to apply this method in a vehicle, the time of domain switch needs to be known (otherwise complexity would be dramatically too high). In contrast, CBNA is applied on a single-image basis during inference, thus an adaptation to the current domain is applied in a continuous fashion. We therefore take a model from GTA-5 and adapt it to 4 different target domains with UBNA (cf. the right-hand side of Table IV). It is observable that UBNA improves the behavior on the domain it adapts to, e.g., from 33.6% to 37.5% for the ResNet-50 model on Cityscapes. However, on other domains the performance often decreases, e.g., the ResNet-50 model exhibits

TABLE III

ADDITIONAL COMPLEXITY IN $10^9$ FLOPS/IMAGE FOR ONLINE ADAPTATION IN INFERENCE; IMAGE RESOLUTION OF $512 \times 1024$

| Network | No adaptation | C-Li | C-Zhang | C-Klingner | CBNA |
|---|---|---|---|---|---|
| ResNet-50 | 0 | 0.30 | 0.30 | 43 | 0.30 |
| VGG-16 | 0 | 0.43 | 0.43 | 161 | 0.43 |

decreased performance on KITTI, BDD, and Mapillary when being adapted to Cityscapes. Similar behavior can also be observed for the other UBNA adaptations, when using a model pre-trained on GTA-5. In the same source domain condition (GTA-5), CBNA improves the performance for both ResNet-50 and VGG-16 in *all* target domains.

In total, Table IV shows 16 adaptation conditions (2 source domains, 4 target domains, 2 network architectures). Here, our novel CBNA (one method!) secured in total twelve $1^{\mathrm{st}}$ or $2^{\mathrm{nd}}$ ranks, without any need of target domain data beforehand, while none of the four UBNA settings could achieve more than five such ranks. Presuming that the time of domain switch is always known, then all four methods "UBNA adapted to X" may be combined, achieving 16 $1^{\mathrm{st}}$ or $2^{\mathrm{nd}}$ ranks, which performs comparable to our proposed CBNA, however, if the time of domain switch is not detected, then drastic performance decreases may occur. It is important to note that CBNA solves this issue with excellent computational efficiency, and does not suffer from adaptation mismatch. Accordingly, in only 3 out of 16 cases performance slightly decreased, while for UBNA there are many cases, where an adaptation mismatch leads to drastic decreases in performance.

To also answer the question, whether CBNA works when being applied to a model with significantly higher initial performance (real-to-real adaptation), we also experiment with models pre-trained on Cityscapes, as shown in Table V. Here, we again observe significant gains in performance, e.g., an absolute 7.0%, 1.7%, and 2.8% on KITTI, BDD, and Mapillary, respectively, for the ResNet-50 model. In comparison to the three UBNA methods, CBNA is always first or second ranked (6 such ranks) in any case better than the baseline without adaptation. The three UBNA methods *together* only achieve 3 such ranks, often exhibiting an even decreased performance in the target domain.

### D. Method Performance Analysis

While all presented results up to now can be applied without making use of source data, the question arises how CBNA, using only the source domain model and target data, compares to standard UDA methods, which make use of source and target domain data at once (no "source-free adaptation", not online capable). We provide such a comparison in Tables VI and VII for the commonly used benchmarks GTA-5 to Cityscapes and SYNTHIA to Cityscapes, respectively. We compare to some of the latest state-of-the-art methods, where we observe that CBNA—as expected—performs worse than these UDA methods due to our much more constrained task definition. In a practical scenario, data often cannot be transferred from the model supplier, making CBNA the only method applicable to improve the model in such cases.

TABLE IV

COMPARISON TO OFFLINE METHODS (UBNA [19]) ACROSS VARIOUS SYNTHETIC SOURCE DOMAINS AND REAL TARGET DOMAIN DATASETS SHOWING THE STRONG ONLINE CAPABILITY OF OUR CBNA ALGORITHM. MIOU VALUES IN %; BEST RESULTS WRITTEN IN **BOLDFACE**

| | Method | $\mathcal{D}^S$: SYNTHIA; mIoU (16 classes) | | | | $\mathcal{D}^S$: GTA-5; mIoU (19 classes) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\mathcal{D}^T$: Cityscapes | $\mathcal{D}^T$: KITTI | $\mathcal{D}^T$: BDD | $\mathcal{D}^T$: Mapillary | $\mathcal{D}^T$: Cityscapes | $\mathcal{D}^T$: KITTI | $\mathcal{D}^T$: BDD | $\mathcal{D}^T$: Mapillary |
| ResNet-50 | No adaptation | 29.1 | **31.7** | 19.9 | **28.4** | 33.6 | 36.9 | 30.0 | 34.4 |
| | UBNA [19] (adapted to Cityscapes) | **33.8** | 31.2 | 20.6 | 27.9 | **37.5** | 33.3 | 29.4 | 32.5 |
| | UBNA [19] (adapted to KITTI) | 30.0 | 30.9 | 19.1 | 26.5 | 32.0 | 36.3 | 29.2 | 33.2 |
| | UBNA [19] (adapted to BDD) | 31.1 | 30.9 | 22.4 | 27.1 | 37.2 | 32.2 | **32.8** | 36.3 |
| | UBNA [19] (adapted to Mapillary) | 31.6 | 31.6 | 20.3 | 26.8 | 36.6 | 33.3 | 31.7 | **39.0** |
| | **CBNA** | 32.5 | 31.2 | **22.6** | 28.3 | 36.7 | **38.8** | 31.2 | 37.8 |
| VGG-16 | No adaptation | 30.0 | 27.5 | 19.1 | **27.6** | 31.5 | 31.0 | 23.1 | 32.9 |
| | UBNA [19] (adapted to Cityscapes) | **34.4** | **29.5** | 17.4 | 26.5 | 36.1 | 28.5 | 21.2 | 28.0 |
| | UBNA [19] (adapted to KITTI) | 30.3 | 28.9 | 15.8 | 25.2 | 31.5 | 32.5 | 21.7 | 27.9 |
| | UBNA [19] (adapted to BDD) | 32.3 | 27.7 | **19.7** | 26.1 | 33.6 | 26.3 | 25.0 | 30.8 |
| | UBNA [19] (adapted to Mapillary) | 31.1 | 28.5 | 17.7 | 26.2 | 33.9 | 29.9 | 25.5 | 34.8 |
| | **CBNA** | 32.1 | 29.4 | 19.4 | 27.0 | **36.4** | **37.5** | **26.0** | **35.9** |

TABLE V

COMPARISON TO OFFLINE METHODS (UBNA [19]) ACROSS FOR ONE REAL SOURCE DOMAIN AND VARIOUS REAL TARGET DOMAIN DATASETS SHOWING THE STRONG ONLINE CAPABILITY OF OUR CBNA ALGORITHM. MIOU VALUES IN %; BEST RESULTS WRITTEN IN **BOLDFACE**

| | Method | $\mathcal{D}^S$: Cityscapes; mIoU (19 classes) | | |
|---|---|---|---|---|
| | | $\mathcal{D}^T$: KITTI | $\mathcal{D}^T$: BDD | $\mathcal{D}^T$: Mapillary |
| ResNet-50 | No adaptation | 46.9 | 36.6 | 43.0 |
| | UBNA [19] (adapted to KITTI) | **56.4** | 33.9 | 44.0 |
| | UBNA [19] (adapted to BDD) | 47.6 | 35.9 | 40.4 |
| | UBNA [19] (adapted to Mapillary) | 48.4 | 33.2 | 44.3 |
| | **CBNA** | 53.9 | **38.3** | **45.8** |
| VGG-16 | No adaptation | 51.1 | 32.7 | 43.2 |
| | UBNA [19] (adapted to KITTI) | **57.1** | 28.5 | 37.8 |
| | UBNA [19] (adapted to BDD) | 45.6 | 28.7 | 36.4 |
| | UBNA [19] (adapted to Mapillary) | 46.7 | 28.5 | 36.1 |
| | **CBNA** | 57.0 | **33.3** | **43.7** |

Here, for a VGG-16-based model adaptation to Cityscapes ($\mathcal{D}^T$), we observe improvements from 31.5% to 36.4% ($\mathcal{D}^S$: GTA-5) and from 30.0% to 32.1%/from 35.2% to 37.7% ($\mathcal{D}^S$: SYNTHIA). Similar improvements are achieved with a ResNet-50 backbone.

A particular advantage is that CBNA can be combined with any domain generalization (DG) pre-training, which is not necessarily the case for standard UDA methods. This would allow a supplier to improve the model, while the car manufacturer can still improve the final model performance on the target domain during vehicle operation using CBNA. Exploiting this advantage, we also present results for such a DG-initialized model, where for VGG-16-based models and the adaptation from GTA-5 or SYNTHIA to Cityscapes, we achieve significantly higher performances of 43.1% and 44.7%/51.3%, respectively, than without DG initialization (36.4% and 32.1%/37.7%). This reduces the gap to UDA methods, which are sometimes even outperformed by the combination of DG pre-training and CBNA as, e.g., for a VGG-16-based model and the adaptation from SYNTHIA to Cityscapes. In other cases, the final performance of CBNA with DG pre-training is still slightly worse than UDA methods, but offers a good alternative to UDA methods, when simultaneous access to source and target domain data is not possible.

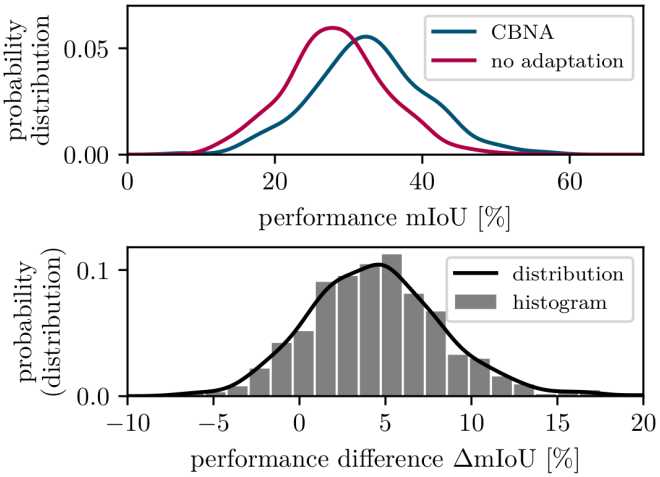



Fig. 7. Probability distributions over the absolute single-image performances (top) and the single-image performance difference before and after application of CBNA (bottom) when adapting from GTA-5 ($\mathcal{D}^S$) to Cityscapes ($\mathcal{D}^T$) using a VGG-16 backbone.

As our method is applicable on a single-image basis, we further analyze the single-image performance in Fig. 7. In the top part, we plot the distributions over the absolute performance for different models. We observe a clear shift towards a higher performance for CBNA compared to the no adaptation model. We further compare the performance before and after application of CBNA for single images and plot the distribution over this performance difference in the bottom part of Fig. 7. We can see that for the large majority of images CBNA improves performance, but for some images the performance also decreases slightly.

The mentioned improvements are also illustrated in Fig. 8, where the segmentation masks generated by CBNA contain much fewer artifacts than the ones of the "no adaptation" baseline. Using a model that has been pre-trained using a DG pre-training improves the results even further, which is consistent with the results from Tables VI and VII.

## VI. CONCLUSION

We presented a continual domain adaptation method for semantic segmentation in constrained (practical) scenarios,

TABLE VI

COMPARISON TO UDA METHODS ON THE CITYSCAPES VALIDATION SET FOR THE ADAPTATION FROM **GTA-5** ($\mathcal{D}^{\mathrm{S}}$) TO CITYSCAPES ($\mathcal{D}^{\mathrm{T}}$). BEST UDA RESULTS AND BEST SOURCE-FREE UDA RESULTS IN BOLDFACE; RESULTS MARKED WITH $*$ ARE TAKEN FROM THE RESPECTIVE PUBLICATIONS

| Network | Method | Source-free adaptation | Online capable | road | sidewalk | building | wall | fence | pole | traffic light | traffic sign | vegetation | terrain | sky | person | rider | car | truck | bus | on rails | motorbike | bike | mIoU (19 cl.) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ResNet-50 | Dong et al. [32]* | no | no | 89.6 | 50.4 | 83.0 | 35.6 | 26.9 | 31.1 | 37.3 | 35.1 | 83.5 | **40.6** | 84.0 | 60.6 | **34.3** | 80.9 | 35.1 | 47.3 | 0.5 | 34.5 | 33.7 | 48.6 |
| | Kim et al. [69]* | no | no | 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 |
| | Mei et al. [42]* | no | no | **94.1** | **58.8** | **85.4** | **39.7** | 29.2 | 25.1 | **43.1** | **34.2** | 84.8 | 34.6 | **88.7** | **62.7** | 30.3 | **87.6** | **42.3** | **50.3** | **24.7** | 35.2 | 40.2 | **52.2** |
| | No adaptation | - | - | 58.1 | 23.8 | 70.5 | 14.8 | 19.2 | 30.5 | 29.0 | 17.7 | 79.1 | 21.8 | 83.1 | 56.4 | 14.8 | 72.3 | 19.5 | 4.5 | 0.9 | 16.5 | 5.6 | 33.6 |
| | Termöhlen et al. [48]* | no | yes | 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 |
| | Klingner et al. [19]* | yes | no | 81.8 | 32.3 | 79.5 | 18.2 | 23.8 | 34.9 | 29.5 | 19.8 | 74.2 | 17.9 | 82.4 | 57.5 | 11.1 | 81.6 | 16.1 | 19.0 | 2.5 | 21.3 | 9.8 | 37.5 |
| | Liu et al. [56]* | yes | no | 84.2 | 39.2 | 82.7 | 27.5 | 22.1 | 25.9 | 31.1 | 21.9 | 82.4 | 30.5 | 85.3 | 58.7 | 22.1 | 80.0 | **33.1** | 31.5 | 3.6 | **27.8** | 30.6 | 43.2 |
| | Teja et al. [54]* | yes | no | **92.3** | **55.2** | 81.6 | **30.8** | 18.8 | 37.1 | 17.7 | 12.1 | 84.2 | **35.9** | 83.8 | 57.7 | 24.1 | 81.7 | 27.5 | **44.3** | 6.9 | 24.1 | **40.4** | **45.1** |
| | **CBNA** | yes | yes | 69.9 | 25.8 | 78.8 | 20.9 | 23.5 | 34.1 | 27.5 | 16.0 | 79.6 | 23.5 | 82.8 | 56.7 | 12.6 | 81.3 | 20.9 | 16.6 | 0.5 | 17.5 | 8.2 | 36.7 |
| | No adaptation (DG-Init) | - | - | 75.9 | 40.7 | 74.4 | 12.0 | 20.9 | 35.6 | 31.1 | 42.5 | 84.0 | 17.3 | 87.4 | 56.9 | 14.7 | 78.4 | 24.3 | 2.8 | 0.1 | 11.4 | 18.1 | 38.3 |
| | **CBNA** (DG-Init) | yes | yes | 89.4 | 48.4 | **83.8** | 21.1 | **26.1** | **42.8** | **35.5** | **45.0** | **85.3** | 32.2 | **88.9** | **60.1** | 21.3 | **85.9** | 25.3 | 6.2 | **10.9** | 14.6 | 27.7 | 44.8 |
| VGG-16 | Dong et al. [32]* | no | no | 89.8 | 46.1 | 75.2 | 30.1 | **27.9** | 15.0 | 20.4 | 18.9 | 82.6 | 39.1 | 77.6 | 47.8 | 17.4 | 76.2 | 28.5 | **33.4** | 0.5 | 29.4 | **30.8** | 41.4 |
| | Kim et al. [69]* | no | no | **92.5** | **54.5** | **83.9** | **34.5** | 25.5 | **31.0** | 30.4 | 18.0 | **84.1** | **39.6** | **83.9** | 53.6 | 19.3 | 81.7 | 21.1 | 13.6 | **17.7** | 12.3 | 6.5 | 42.3 |
| | Yang et al. [70]* | no | no | 90.1 | 41.2 | 82.2 | 30.3 | 21.3 | 18.3 | **33.5** | **23.0** | **84.1** | 37.5 | 81.4 | **54.2** | **24.3** | **83.0** | **27.6** | 32.0 | 8.1 | **29.7** | 26.9 | **43.6** |
| | No adaptation | - | - | 55.8 | 21.9 | 65.9 | 15.2 | 14.7 | 27.5 | 31.0 | 17.9 | 77.8 | 19.5 | 74.4 | 55.2 | 12.1 | 71.7 | 11.9 | 3.3 | 0.5 | 13.2 | 9.6 | 31.5 |
| | Klingner et al. [19]* | yes | no | 80.8 | 29.4 | 77.6 | 19.8 | 17.1 | 33.9 | 29.3 | 20.5 | 73.9 | 16.8 | 76.7 | 58.3 | **15.2** | 79.1 | 13.6 | 12.5 | 5.7 | **14.1** | 10.8 | 36.1 |
| | Liu et al. [56]* | yes | no | **81.8** | 35.4 | **82.3** | **21.6** | 20.2 | 25.3 | 17.8 | 4.7 | 80.7 | 24.6 | 80.4 | 50.5 | 9.2 | 78.4 | **26.3** | **19.8** | **11.1** | 6.7 | 4.3 | 35.9 |
| | **CBNA** | yes | yes | 75.8 | 31.9 | 75.5 | 17.2 | 17.9 | 34.4 | 30.0 | 18.9 | 80.5 | 22.7 | 78.0 | 58.3 | 14.0 | **82.6** | 15.2 | 10.4 | 1.5 | 13.2 | 13.0 | 36.4 |
| | No adaptation (DG-Init) | - | - | 64.7 | 32.8 | 73.6 | 16.5 | 22.8 | 39.4 | 37.0 | 44.6 | 85.9 | 30.8 | 83.9 | 58.1 | 07.2 | 68.3 | 18.3 | 8.1 | 5.2 | 9.7 | 13.8 | 37.9 |
| | **CBNA** (DG-Init) | yes | yes | 81.4 | **41.5** | 81.8 | 21.1 | **26.0** | **44.2** | **41.3** | **45.0** | **86.5** | **35.0** | **87.1** | **60.6** | 14.8 | 80.7 | 22.9 | 12.4 | 5.8 | 12.6 | **19.3** | **43.1** |

TABLE VII

COMPARISON TO UDA METHODS ON THE CITYSCAPES VALIDATION SET FOR THE ADAPTATION FROM **SYNTHIA** ($\mathcal{D}^{\mathrm{S}}$) TO CITYSCAPES ($\mathcal{D}^{\mathrm{T}}$). BEST UDA RESULTS AND BEST SOURCE-FREE UDA RESULTS IN BOLDFACE; RESULTS MARKED WITH $*$ ARE TAKEN FROM THE RESPECTIVE PUBLICATIONS

| Network | Method | Source-free adaptation | Online capable | road | sidewalk | building | wall | fence | pole | traffic light | traffic sign | vegetation | terrain | sky | person | rider | car | truck | bus | on rails | motorbike | bike | mIoU (16 cl.) | mIoU (13 cl.) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ResNet-50 | Yang et al. [70]* | no | no | 85.1 | 44.5 | 81.0 | - | - | - | 16.4 | 15.2 | 80.1 | - | 84.8 | 59.4 | 31.9 | 73.2 | - | 41.0 | - | 32.6 | 44.7 | - | 53.1 |
| | Dong et al. [32]* | no | no | 80.2 | 41.1 | 78.9 | **23.6** | 0.6 | 31.0 | 27.1 | **29.5** | 82.5 | - | 83.2 | 62.1 | 26.8 | 81.5 | - | 37.2 | - | 27.3 | 42.9 | 47.2 | - |
| | Mei et al. [42]* | no | no | 81.9 | 41.5 | **83.3** | 17.7 | **4.6** | **32.3** | **30.9** | 28.8 | **83.4** | - | 85.0 | **65.5** | 30.8 | **86.5** | - | 38.2 | - | **33.1** | **52.7** | **49.8** | **57.0** |
| | No adaptation | - | - | 36.5 | 18.6 | 68.3 | 2.0 | 0.2 | 30.3 | 6.0 | 10.2 | 74.5 | - | 81.6 | 51.9 | 10.6 | 41.3 | - | 9.5 | - | 2.2 | 22.6 | 29.1 | 34.1 |
| | Termöhlen et al. [48]* | no | yes | 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 |
| | Klingner et al. [19]* | yes | no | 62.5 | 22.8 | 75.6 | 3.1 | 0.5 | 32.5 | 8.6 | 11.3 | 73.0 | - | 82.7 | 42.5 | 12.5 | 67.1 | - | 12.5 | - | 5.7 | 27.8 | 33.8 | 39.7 |
| | Teja et al. [54]* | yes | no | 59.3 | 24.6 | 77.0 | **14.0** | 1.8 | 31.5 | 18.3 | 32.0 | 83.1 | - | 80.4 | 46.3 | 17.8 | 76.7 | - | 17.0 | - | **18.5** | 34.6 | 39.6 | 45.0 |
| | Liu et al. [56]* | yes | no | **81.9** | 44.9 | **81.7** | 4.0 | 0.5 | 26.2 | 3.3 | 10.7 | **86.3** | - | 89.4 | 37.9 | 13.4 | **80.6** | - | **25.6** | - | 9.6 | 31.3 | 39.2 | 45.9 |
| | **CBNA** | yes | yes | 53.9 | 21.6 | 74.5 | 1.2 | 0.2 | 33.4 | 7.9 | 12.4 | 77.4 | - | 81.5 | 42.7 | 11.7 | 57.0 | - | 12.2 | - | 4.9 | 27.6 | 32.5 | 38.2 |
| | No adaptation (DG-Init) | - | - | 70.6 | 41.0 | 71.9 | 10.2 | **14.6** | 40.6 | 26.4 | **40.0** | 85.6 | - | **90.4** | **60.0** | 17.5 | 54.0 | - | 10.3 | - | 4.2 | 36.3 | 42.1 | 47.7 |
| | **CBNA** (DG-Init) | yes | yes | 79.9 | **46.7** | 74.5 | 10.5 | 10.2 | **41.3** | **28.3** | 39.1 | 84.3 | - | 88.6 | 59.0 | 18.5 | 75.8 | - | 14.5 | - | 4.1 | **37.0** | **44.5** | **51.1** |
| VGG-16 | Lee et al. [67]* | no | no | 71.1 | 29.8 | 71.4 | 3.7 | 0.3 | **33.2** | 6.4 | 15.6 | **81.2** | - | 78.9 | 52.7 | 13.1 | 75.9 | - | 25.5 | - | 10.0 | 20.5 | 36.8 | **42.4** |
| | Dong et al. [32]* | no | no | 70.9 | **30.5** | **77.8** | **9.0** | **0.6** | 27.3 | 8.8 | 12.9 | 74.8 | - | 81.1 | 43.0 | **25.1** | 73.4 | - | **34.5** | - | **19.5** | 38.2 | 39.2 | - |
| | Yang et al. [70]* | no | no | **73.7** | 29.6 | 77.6 | 1.0 | 0.4 | 26.0 | **14.7** | **26.6** | 80.6 | - | **81.8** | **57.2** | 24.5 | **76.1** | - | 27.6 | - | 13.6 | **46.6** | **41.1** | - |
| | No adaptation | - | - | 49.4 | 20.8 | 61.5 | 3.6 | 0.1 | 30.5 | 13.6 | 14.1 | 74.4 | - | 75.5 | 53.5 | 10.6 | 47.2 | - | 4.8 | - | 3.0 | 17.1 | 30.0 | 35.2 |
| | Klingner et al. [19]* | yes | no | 72.3 | 26.6 | 73.0 | 2.3 | 0.3 | 31.5 | 12.1 | 16.6 | 72.1 | - | 75.6 | 45.4 | 13.6 | 61.2 | - | **8.5** | - | **8.5** | 30.1 | 34.4 | 40.7 |
| | **CBNA** | yes | yes | 52.9 | 25.0 | 62.4 | 2.7 | 0.2 | 32.5 | 13.3 | 16.1 | 78.8 | - | 75.8 | 46.8 | 12.8 | 55.7 | - | 6.3 | - | 6.9 | 26.0 | 32.1 | 37.7 |
| | No adaptation (DG-Init) | - | - | 72.3 | 45.3 | **78.9** | **12.6** | **15.0** | 41.1 | 28.4 | **42.7** | **85.3** | - | 87.8 | 61.5 | 20.0 | 50.5 | - | 4.5 | - | 4.0 | 40.2 | 43.1 | 48.7 |
| | **CBNA** (DG-Init) | yes | yes | **77.9** | **48.1** | 78.8 | 11.2 | 8.0 | **43.6** | **31.7** | 42.2 | 84.3 | - | **88.2** | **62.7** | **21.3** | **63.2** | - | 5.5 | - | 7.1 | **41.6** | **44.7** | **51.3** |

where one does not have simultaneous access to both source and target domain data. For these cases, the given trained deep neural network (DNN) model from the source domain can be adapted in an online fashion to single images of different target domains by using our source-free Continuous BatchNorm Adaptation (CBNA) method, which yields a significant increase in performance. This presents a clear advantage over previous offline unsupervised domain adaptation (UDA) methods, as we perform a single-image adaptation which is employed during inference, requiring only minimal computational overhead while incurring no algorithmic delay. For semantic segmentation, we presented experiments for three source domains and four target domains, showing the good generalization capability of our method. We thereby

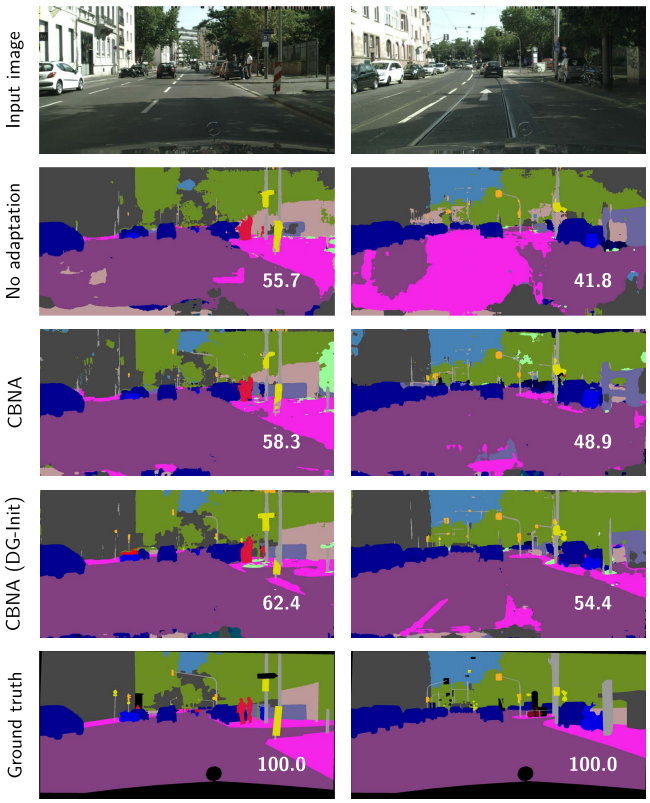



Fig. 8. Qualitative comparison when adapting from GTA-5 ($\mathcal{D}^{\mathrm{S}}$) to Cityscapes ($\mathcal{D}^{\mathrm{T}}$) between the model without adaptation and our CBNA models using a `VGG-16` backbone. Single image mIoU performance [%] in white.

offer the possibility to deploy a UDA method in an online fashion (i.e., in an operating vehicle) for continual adaptation. Future works could integrate our method in tasks such as instance segmentation or object detection as a standard normalization layer modification to improve these tasks' target domain performance. Also, transferring the advances of other source-free domain adaptation methods to the continual setting may further facilitate target domain performance gains.

## APPENDIX

### A. Mapillary Label Inconsistency

To deal with the label inconsistency between Cityscapes and Mapillary, the classes "bike-lane", "crosswalk-plain", "road", "lane marking - crosswalk", and "lane marking - general" are mapped to the "road" class, and the classes "bicyclist", "motorcyclist", and "other rider" are mapped to the "rider" class. All other classes defined in Cityscapes are also present in Mapillary and can be mapped in a straightforward fashion. All remaining additional classes defined in Mapillary are mapped to the background class.

### B. Network Architecture Details

We rely on the encoder-decoder network architecture from [19]. The encoder is a standard `ResNet-50` [5] or `VGG-16` [63] model with Imagenet-pretrained weights [71].

TABLE VIII
**ADDITIONAL FLOPS** INDUCED BY THE SINGLE EQUATIONS INVOLVED FOR CBNA AND FOR THE REFERENCE METHODS C-X

| Equations | | (2), (3) | (7), (8) | (9), (10) | (11), (12) | forward pass |
|---|---|---|---|---|---|---|
| FLOPs | Complexity | $\sim H_\ell W_\ell C_\ell$ | $\sim C_\ell$ | $\sim H_\ell W_\ell C_\ell$ | $\sim C_\ell$ | |
| | `ResNet-50` | $300 \cdot 10^6$ | $17 \cdot 10^3$ | $300 \cdot 10^6$ | $17 \cdot 10^3$ | $43 \cdot 10^9$ |
| | `VGG-16` | $430 \cdot 10^6$ | $106 \cdot 10^3$ | $430 \cdot 10^6$ | $106 \cdot 10^3$ | $161 \cdot 10^9$ |
| Methods | | C-Klingner<br>C-Li<br>C-Zhang | C-Klingner | **CBNA** | **CBNA** | C-Klingner |

The basic setup of each layer in these architectures is the use of a convolutional layer, followed by a batch normalization (BN) layer, followed by an activation function (mainly ReLU variants), where the BN layers in the encoder are essential for our adaptation method. In total, the feature resolution is downsampled five times resulting in a downsampling factor of $2^5$. The intermediate features at each resolution are passed on to the decoder, implementing a U-Net-like structure inspired by [72].

The decoder uses a simple fully convolutional architecture as defined in [73]. The features from the skip connections are concatenated with the features from the decoder, followed by two convolutional layers with ELU activation and nearest neighbor upsampling. Note that no BN layers are used in the decoder. The output convolution produces output logits in $S = |\mathcal{S}|$ feature maps, which are converted to posterior class probabilities for each pixel by a pixel-wise softmax function.

### C. Method Complexity Analysis

To better understand the additional computational complexity induced by CBNA and the reference methods C-X, we analyze the single involved equations in terms of their induced additional FLOPs in Table VIII. The numbers are accumulated over all BN layers in the `VGG-16` and `ResNet-50` network architectures. For the reference methods C-Li and C-Zhang, we need to apply (2) and (3), which then replace the source domain statistics. As here the mean over each feature map is computed, the additional FLOPs induced by these equations scale with the feature map resolution $H_\ell \cdot W_\ell$ and the number of feature maps $C_\ell$.

For C-Klingner, additionally (7) and (8) are applied on the first forward pass to mix source and target domain statistics. As, however, only one value per feature map is updated, these equations induce additional FLOPs in the order of $C_\ell$. However, here most additional computations are induced by the additional forward pass (cf. Table VIII), where all computations in the convolutional layers have to be recomputed, inducing much more additional FLOPs than just a few additional computations in the BN layers as in C-Li/C-Zhang.

For CBNA, on the other hand, first (9) and (10) have to be applied, which, however, induces exactly the same number of additional FLOPs as (2) and (3) for C-Li/C-Zhang, i.e., (9) and (10) also scale with $H_\ell \cdot W_\ell \cdot C_\ell$. Afterwards only (11) and (12) have to be executed, which only induces additional FLOPS in the order of $C_\ell$. For the given network architecture, the feature map resolution is up to the order of $H_\ell \cdot W_\ell \sim 10^6$ (image resolution of $512 \times 1024$), while the maximum number of channels is only in the order of $C_\ell \sim 10^3$, which is why the main complexity of CBNA is caused by (9) and (10). This

also explains, why the number of additional FLOPs of CBNA and C-Li/C-Zhang in Table III appears to be equal, as the additional complexity induced by (11) and (12) is negligible.

## REFERENCES


[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 3431–3440.

[2] L. C. Chen, G. Papandreou, and I. Kokkinos, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Jun. 2017.

[3] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proc. Adv. Neural Inf. Process. Syst.*, Montréal, QC, Canada, Jun. 2014, pp. 2366–2374.

[4] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 1851–1860.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[6] A. Kirillov, R. Girshick, K. He, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proc. CVPR*, Long Beach, CA, USA, Jun. 2019, pp. 9404–9413.

[7] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Las Condes, Chile, Dec. 2015, pp. 1440–1448.

[8] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, "Domain adaptive faster R-CNN for object detection in the wild," in *Proc. CVPR*, Salt Lake City, UT, USA, Jun. 2018, pp. 3339–3348.

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[10] M. Cordts, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jul. 2016, pp. 3213–3223.

[11] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. ICML*, Lille, France, Jul. 2015, pp. 1180–1189.

[12] Y.-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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 7472–7481.

[13] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, no. 1, pp. 1–40, Dec. 2016.

[14] X. Yue, Y. Zhang, S. Zhao, A. Sangiovanni-Vincentelli, K. Keutzer, and B. Gong, "Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data," in *Proc. ICCV*, Seoul, South Korea, Oct. 2019, pp. 2100–2110.

[15] J. Zhang, L. Qi, Y. Shi, and Y. Gao, "Generalizable semantic segmentation via model-agnostic learning and target-specific normalization," *Pattern Recognit.*, vol. 122, no. 122, Feb. 2022, Art. no. 108292.

[16] F. Pan, I. Shin, F. Rameau, S. Lee, and I. S. Kweon, "Unsupervised intra-domain adaptation for semantic segmentation through self-supervision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 3764–3773.

[17] Y. Li, L. Yuan, and N. Vasconcelos, "Bidirectional learning for domain adaptation of semantic segmentation," in *Proc. CVPR*, Long Beach, CA, USA, Jun. 2019, pp. 6936–6945.

[18] Z. Wang *et al.*, "Differential treatment for stuff and things: A simple unsupervised domain adaptation method for semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 12635–12644.

[19] M. Klingner, J.-A. Termohlen, J. Ritterbach, and T. Fingscheidt, "Unsupervised BatchNorm adaptation (UBNA): A domain adaptation method for semantic segmentation without using source domain representations," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. Workshops (WACVW)*, Waikoloa, HI, USA, Jan. 2022, pp. 1–11.

[20] R. Gong *et al.*, "Cluster, split, fuse, and update: Meta-learning for open compound domain adaptive semantic segmentation," in *Proc. CVPR*, Jun. 2021, pp. 8344–8354.

[21] J. Zhang, K. Yang, A. Constantinescu, K. Peng, K. Müller, and R. Stiefelhagen, "Trans4Trans: Efficient transformer for transparent object segmentation to help visually impaired people navigate in the real world," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 1760–1770.

[22] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, "Revisiting batch normalization for practical domain adaptation," in *Proc. ICLR*, Toulon, France, Apr. 2017, pp. 1–10.

[23] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive batch normalization for practical domain adaptation," *Pattern Recognit.*, vol. 80, pp. 109–117, Aug. 2018.

[24] T. Saikia, C. Schmid, and T. Brox, "Improving robustness against common corruptions with frequency biased models," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 1–13.

[25] X. Pan, P. Luo, J. Shi, and X. Tang, "Two at once: Enhancing learning and generalization capacities via IBN-Net," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, Sep. 2018, pp. 464–479.

[26] Q. Dou, D. C. de Castro, K. Kamnitsas, and B. Glocker, "Domain generalization via model-agnostic learning of semantic features," in *Proc. NeurIPS*, Vancouver, BC, Canada, Dec. 2019, pp. 6447–6458.

[27] D. Li, J. Zhang, Y. Yang, C. Liu, Y.-Z. Song, and T. M. Hospedales, "Episodic training for domain generalization," in *Proc. CVPR*, Seoul, South Korea, Oct. 2019, pp. 1446–1455.

[28] S. Seo, Y. Suh, D. Kim, G. Kim, J. Han, and B. Han, "Learning to optimize domain specific normalization for domain generalization," in *Proc. ECCV*, Glasgow, U.K., Aug. 2020, pp. 68–83.

[29] S. Choi, S. Jung, H. Yun, J. T. Kim, S. Kim, and J. Choo, "RobustNet: Improving domain generalization in urban-scene segmentation via instance selective whitening," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11580–11590.

[30] J.-A. Bolte *et al.*, "Unsupervised domain adaptation to improve image segmentation quality both in the source and target domain," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Long Beach, CA, USA, Jun. 2019, pp. 1404–1413.

[31] M. Chen, H. Xue, and D. Cai, "Domain adaptation for semantic segmentation with maximum squares loss," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 2090–2099.

[32] J. Dong, Y. Cong, G. Sun, Y. Liu, and X. Xu, "CSCL: Critical semantic-consistent learning for unsupervised domain adaptation," in *Proc. ECCV*, Glasgow, U.K., Aug. 2020, pp. 745–762.

[33] L. Du *et al.*, "SSF-DAN: Separated semantic feature based domain adaptation network for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 982–991.

[34] J. Hoffman, D. Wang, F. Yu, and T. Darrell, "FCNs in the wild: Pixel-level adversarial and constraint-based adaptation," 2016, *arXiv:1612.02649*.

[35] J. Huang, S. Lu, D. Guan, and X. Zhang, "Contextual-relation consistent domain adaptation for semantic segmentation," in *Proc. ECCV*, Glasow, U.K., Aug. 2020, pp. 705–722.

[36] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Perez, "ADVENT: Adversarial entropy minimization for domain adaptation in semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 2517–2526.

[37] J. Xu, L. Xiao, and A. M. López, "Self-supervised domain adaptation for computer vision tasks," *IEEE Access*, vol. 7, pp. 156694–156706, 2019.

[38] R. Gong, W. Li, Y. Chen, and L. Van Gool, "DLOW: Domain flow for adaptation and generalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 2477–2486.

[39] J. Hoffman *et al.*, "CyCADA: Cycle-consistent adversarial domain adaptation," in *Proc. ICML*, Stockholm, Sweden, Jul. 2018, pp. 1989–1998.

[40] Y. Yang and S. Soatto, "FDA: Fourier domain adaptation for semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 4085–4095.

[41] J. Choi, T. Kim, and C. Kim, "Self-ensembling with GAN-based data augmentation for domain adaptation in semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 6830–6840.

[42] K. Mei, C. Zhu, J. Zou, and S. Zhang, "Instance adaptive self-training for unsupervised domain adaptation," in *Proc. ECCV*, Glasgow, U.K., Jul. 2020, pp. 415–430.

[43] M. N. Subhani and M. Ali, "Learning from scale-invariant examples for domain adaptation in semantic segmentation," in *Proc. ECCV*, Glasgow, U.K., Aug. 2020, pp. 290–306.

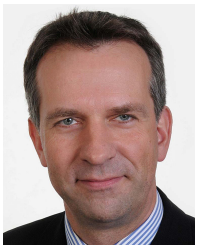
[44] Y. Zou, Z. Yu, B. V. K. V. Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *Proc. ECCV*, Munich, Germany, Sep. 2018, pp. 289–305.

[45] V. K. Kurmi, V. K. Subramanian, and V. P. Namboodiri, "Domain impression: A source data free domain adaptation method," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 615–625.

[46] M. Wulfmeier, A. Bewley, and I. Posner, "Incremental adversarial domain adaptation for continually changing environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Brisbane, QD, Australia, May 2018, pp. 4489–4495.
[47] S. Stan and M. Rostami, "Unsupervised model adaptation for continual semantic segmentation," 2020, *arXiv:2009.12518*.
[48] J.-A. Termohlen, M. Klingner, L. J. Brettin, N. M. Schmidt, and T. Fingscheidt, "Continual unsupervised domain adaptation for semantic segmentation by online frequency domain style transfer," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, Sep. 2021, pp. 2881–2888.
[49] X. Li *et al.*, "A free lunch for unsupervised domain adaptive object detection without source data," 2020, *arXiv:2012.05400*.
[50] H.-W. Yeh, B. Yang, P. C. Yuen, and T. Harada, "SoFA: Source-data-free feature alignment for unsupervised domain adaptation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 474–483.
[51] S. Li, X. Wang, Y. Cao, F. Xue, Z. Yan, and H. Zha, "Self-supervised deep visual odometry with online adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 6339–6348.
[52] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation," in *Proc. ICML*, Jul. 2020, pp. 6028–6039.
[53] R. Li, Q. Jiao, W. Cao, H.-S. Wong, and S. Wu, "Model adaptation: Unsupervised domain adaptation without source data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 9641–9650.
[54] F. Fleuret, "Uncertainty reduction for model adaptation in semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 9613–9623.
[55] J. N. Kundu, A. Kulkarni, A. Singh, V. Jampani, and R. V. Babu, "Generalize then adapt: Source-free domain adaptive semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 7046–7056.
[56] Y. Liu, W. Zhang, and J. Wang, "Source-free domain adaptation for semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 1215–1224.
[57] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, Lille, France, Jul. 2015, pp. 448–456.
[58] S. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Proc. ECCV*, Amsterdam, The Netherlands, Oct. 2016, pp. 102–118.
[59] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 3234–3243.
[60] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 3061–3070.
[61] F. Yu *et al.*, "BDD100 K: A diverse driving video database with scalable annotation tooling," 2018, *arXiv:1805.04687*.
[62] G. Neuhold, T. Ollmann, S. R. Bulo, and P. Kontschieder, "The Mapillary vistas dataset for semantic understanding of street scenes," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 4990–4999.
[63] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, San Diego, CA, USA, May 2015, pp. 1–27.
[64] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," 2016, *arXiv:1606.02147*.
[65] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, San Diego, CA, USA, May 2015, pp. 1–15.
[66] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.
[67] K.-H. Lee, G. Ros, J. Li, and A. Gaidon, "SPIGAN: Privileged adversarial learning from simulation," in *Proc. ICLR*, New Orleans, LA, USA, Apr. 2019, pp. 1–14.
[68] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. P. Perez, "DADA: Depth-aware domain adaptation in semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 7364–7373.
[69] M. Kim and H. Byun, "Learning texture invariant representation for domain adaptation of semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 12975–12984.
[70] J. Yang, W. An, S. Wang, X. Zhu, C. Yan, and J. Huang, "Label-driven reconstruction for domain adaptation in semantic segmentation," in *Proc. ECCV*, Glasgow, U.K., Aug. 2020, pp. 480–498.
[71] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[72] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, Munich, Germany, Oct. 2015, pp. 234–241.
[73] C. Godard, O. M. Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Seoul, South Korea, Oct. 2019, pp. 3828–3838.

**Marvin Klingner** (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in physics from Georg-August-Universität Göttingen, Germany, in 2016 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Faculty of Electrical Engineering, Information Technology, and Physics, Technische Universität Braunschweig, Germany. His research interests lie in self-supervised 3D geometry perception with neural networks and in multi-task learning and domain adaptation approaches for neural networks with focus on computer vision tasks. He was the recipient of the Dr. Berliner–Dr. Ungewitter Award of the Faculty of Physics at Georg-August-Universität Göttingen in 2018 and was given the CVPR Workshop Best Paper Award in 2020 and 2021.

**Mouadh Ayache** received the B.Sc. degree in industrial engineering, specialized in electrical engineering, from Technische Universität Braunschweig, Germany, in 2019, where he currently studies the M.Sc. degree in electrical engineering. Since January 2021, he has been writing his master's thesis "Adaptive Online Domain Adaption Without Source Data" under the supervision of T. Fingscheidt and M. Klingner with the Signal Processing and Machine Learning Group. He is interested in deep learning applications in autonomous driving, security, and digital hardware design.



**Tim Fingscheidt** (Senior Member, IEEE) received the Dipl.-Ing. degree in electrical engineering and the Ph.D. degree from RWTH Aachen University, Germany, in 1993 and 1998, respectively. He joined AT&T Labs, Florham Park, NJ, USA, in 1998; and Siemens AG (Mobile Devices), Munich, Germany, in 1999. With Siemens Corporate Technology, Munich, he was leading the speech technology development activities from 2005 to 2006. Since 2006, he has been a Full Professor with the Institute for Communications Technology, Technische Universität Braunschweig, Germany. His research interests include speech technology and vision for autonomous driving. He was a member of the IEEE Speech and Language Processing Technical Committee from 2011 to 2018. He was the recipient of several awards, including the Vodafone Mobile Communications Foundation Prize in 1999 and the 2002 ITG Prize of the Association of German Electrical Engineers (VDE ITG). In 2017 and 2020, he coauthored the ITG Award-winning publication. He was given the Best Paper Award of a CVPR Workshop from 2019 to 2021. He has been the Speaker of the Speech Acoustics Committee ITG AT3 since 2015. He was an Associate Editor of the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING from 2008 to 2010.