Unsupervised Domain Adaptation to Improve Image Segmentation Quality *Both* in the Source and Target Domain

Jan-Aike Bolte[1] Markus Kamp[1] Antonia Breuer[2] Silviu Homoceanu[2]
Peter Schlicht[2] Fabian Hüger[2] Daniel Lipinski[2] Tim Fingscheidt[1]

{j.bolte, m.kamp, t.fingscheidt}@tu-bs.de

{antonia.breuer, silviu.homoceanu, peter.schlicht, fabian.hueger, daniel.lipinski}@volkswagen.de

[1] Technische Universität Braunschweig [2] Volkswagen Group Research

Abstract

Domain adaptation is becoming more and more important with the advancing development of machine learning and the ever-increasing diversity of available data. The advancement of autonomous driving depends very much on progress in machine learning, which relies heavily on vast amounts of training data. It is well known that the performance of such models drops, as soon as the data used during inference stems from a different domain as the training data. To avoid the need to label a separate dataset for each new domain, e.g., each new camera sensor, methods for domain adaptation are necessary. Most interesting are unsupervised domain adaptation approaches since they do not require costly labels for the target domain. In this paper we adapt a known domain adaptation approach to work in an unsupervised fashion for semantic segmentation on high resolution data and provide some analysis of the learned representations. With our domain-adapted semantic segmentation we were able to achieve a significant 15 *% absolute increase in mean intersection over union (mIoU), securing a surprisingly good 5th rank on the target domain KITTI test set without having used any KITTI labels during training. In addition to that, we even improved quality on the source domain data.*

1. Introduction

Machine learning is used in more and more applications, e.g., in autonomous vehicles, and typically requires vast amounts of annotated data for training. However, the process of labeling a sufficient amount of data for the desired task is often tedious and costly. For many tasks there are datasets openly available that can be used for training, but in many cases the domain of the training data does not match the domain of the target data on which the model will be evaluated during inference. This may be due to the fact that the used sensors are different or the data was recorded under different conditions, e.g., different illumination, season, country, and many more. Domain adaptation aims at adjusting either the model to perform its tasks independently of the target domain or at adjusting the data from the new domain to match the distribution from the source domain, on which the model was trained on. With both approaches it is possible to train a model on existing datasets, leveraging on the already available labels. Domain adaptation has therefore become one very important research field in machine learning. Formally spoken, domain adaptation transfers the knowledge about the source domain $\mathcal{D}_S$ to the target domain $\mathcal{D}_T$, between which there is a domain shift.

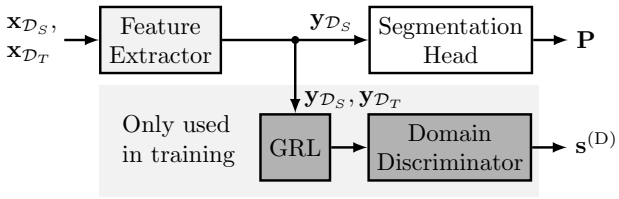

Figure 1. High-level overview of the **segmentation framework** with the **segmentation head**. The feature extractor is illustrated in detail in Figure 2. The domain discriminator and the gradient reversal layer (GRL) are illustrated in Figure 4, both only being necessary in training. Input data $\mathbf{x}_{\mathcal{D}_S}, \mathbf{x}_{\mathcal{D}_T}$ and feature data $\mathbf{y}_{\mathcal{D}_S}, \mathbf{y}_{\mathcal{D}_T}$ are shown as used in training. During inference, input and feature data is from the target domain, in this work, however, not necessarily so.

Following the classification of [40] we investigate an approach for *homogeneous* domain adaptation in this work, as the feature spaces between the source and the target domain are identical with the same feature dimension. Domain adaptation can further be classified either as supervised, semi-supervised or unsupervised. In contrast to (semi-)supervised approaches, where there are still (small)

amounts of labeled images of the target domain needed, unsupervised domain adaptation only uses the labels of the source domain and can therefore be used with no further labeling costs on any new domain. This makes unsupervised approaches especially interesting for tasks, where the labeling is very time-consuming and therefore costly, e.g., semantic image segmentation, where manual labeling can take up to 90 minutes per image [9].

Unsupervised domain adaptation is also of great interest in the field of automated driving, as it allows the simple transferability of models to new sensor setups, e.g., new camera sensors. This is especially important for machine learning models in the perception layer of automated vehicles, as these models directly extract information about the car's surroundings. A typical application here is again semantic segmentation, which outputs the semantic information for each pixel in an image, such as the location of other traffic participants or the location of the road itself. As exemplary datasets we will use Cityscapes [9], KITTI [15] and Berkeley DeepDrive [44], which are all well-known in the research field of automated driving. As the task used in this work we select semantic segmentation, which aims to assign each pixel in an image to one predefined object class. As already mentioned, it is as well a task which needs time-consuming and costly labeling as it is important for the perception in automated vehicles, which makes it an interesting subject of unsupervised domain adaptation research.

Our contribution with this work is threefold. Firstly, we adapt a known technique for unsupervised domain adaptation [13] to the task of semantic segmentation. Secondly, we show that this approach significantly improves the accuracy in the target domain, but surprisingly also in the source domain. Thirdly, we provide an analysis indicating that the used approach does indeed perform a domain adaptation and forces our model to learn domain-invariant features.

The paper is structured as follows. In Section 2 we provide related work in the field of domain adaptation for visual tasks. In Section 3 we present the methods and measures used for semantic segmentation and domain adaptation. Section 4 introduces the employed datasets and describes the training of our model. Finally we will present results and some further analysis in Section 5, before providing conclusions in Section 6.

2. Related Work

As already mentioned, domain adaptation can be achieved with different approaches. Typically, the approaches can be arranged into (semi-)supervised and unsupervised approaches. Recently, several publications provided good overviews on the entire topic of domain adaptation for vision tasks regarding both shallow methods and deep methods for deep neural networks [39, 10, 40]. In the following section, the related work for visual domain adaptation in neural networks will be presented.

(Semi-)Supervised Domain Adaptation aims at adapting a machine learning system to a new domain for which there are (some) labels given. Due to the ever improving techniques for unsupervised domain adaptation and the above-mentioned advantages, supervised approaches [29, 35, 43] are becoming less popular. Aiming at domain adaptation generalization [27], supervised approaches have been used lately. However, we decide to employ an unsupervised approach, since we see greater potential in using vast amounts of (cheap) unlabeled data.

Unsupervised Domain Adaptation only relies on the labels of the source domain and unlabeled data for the target domain. The tasks performed by such a model can be diverse. There are typically two approaches that are used for unsupervised domain adapation. The *first approach* uses a style-transfer of the input images. The authors of [2] performed a domain adaptation by using an image style transfer to learn a depth estimation network on video game images that can afterwards be used for real world data. Many methods use generative adversarial networks (GANs) for such a transfer [5], e.g., for person re-identification under diverse lighting conditions [3, 11]. Also autoencoder networks can be used for this kind of domain adaptation [25]. So-called domain separation networks to perform an unsupervised domain adaptation have been proposed in [6] on low resolution images.

A *second approach* would be to learn domain-invariant features from the data. Ganin et al. introduced this approach with their additional domain discriminator network and the gradient reversal layer (GRL) that forces the feature extractor to produce domain-invariant representations on low resolution images [13]. They showed later that this approach also works for person re-identification [14]. These methods that try to learn domain-invariant features or feature projections [23, 24, 30, 31, 37, 42] belong to the second approach. A hybrid method was introduced with the `DupGAN` that forces domain-invariant representation and performs a style transfer, using an encoder, a generator, and two discriminators [20]. Another method tries to detect "landmarks" in the source dataset that are distributed most similar to the target dataset [16]. Although most approaches only use domain adaptation for two domains, it is possible to employ multiple domains for domain adaptation [46]. Rozantsev et al. introduced an approach that can be used semi- and unsupervised, employing auxiliary residual networks predicting parameters for the target domain network [32]. Another combination of methods combines a discriminative model and weight sharing with a GAN loss [38].

All of the above-mentioned approaches are used for visual domain adaptation, but only on toy examples and low-resolution datasets and mostly for classification tasks. In the field of automated driving datasets a much higher resolu-

tion is used and the tasks, e.g., object detection and domain adaptation, are much more complex. Chen et al. [8] built upon the well-known `Faster R-CNN` with adaptation on image and instance level, but also upon the work of Ganin et al. [13]. Another approach that learns feature transformations using a conditional generator network was introduced by Hong et al. [19], while the authors of [21] use a simple activation matching. A hybrid approach that uses an image space and feature space alignment by using a GAN approach was presented in [17], one that employs a single network for each of the two tasks in [45]. Domain adversarial training in combination with weight sharing was used by Hoffmann et al. [18]. GAN approaches are employed either for style transfer [22], or to bring the feature representations close to the source domain [34]. Murez et al. [28] also use a network for style transfer but add extra losses to be able to reconstruct both source and target images from the feature representations and enforce the extracted features to be indistinguishable. An approach to improve domain adaptation is to utilize task-specific decision boundaries of the used network and align the distributions of the source and target domain [33].

We decide to build upon the approach by Ganin et al. [13] for two reasons. Firstly, it is a simple approach that can be used to extend any network that extracts features, regardless of the intended application, e.g., semantic segmentation or object detection. Secondly, it is an approach that does not use style transfer and does therefore not alter the input images, but forces the network to learn domain-invariant, better generalizing features. It seems much safer to have a model trained directly on unaltered images, than to perform style transfer first and then use a network that has never seen the unaltered images.

3. Method and Measures

In this section, the methods for semantic segmentation, domain adaptation, and the network architectures are presented. Furthermore, a measure for the domain shift is introduced that is later used for further analysis of our results in Section 5.2.

3.1. Semantic Segmentation

Semantic image segmentation is the task of finding a transformation of an input image into an image of semantically-related parts, known as a segmentation mask. To accomplish this, a neural network assigns each pixel to a specific predefined class. In our work we adopt a segmentation network based on the `DeepLabv3` [7] with some improvements by using the `WideResNet38` [41] as a feature extractor, which is pre-trained on the ImageNet corpus. The architecture can be thought of as a two-part network with a *feature extractor* (Figure 2), which extracts meaningful features from the images, and a *segmentation head*, which performs the transformation of these features into a segmentation map corresponding to the input image [4]. The dataflow is depicted in the upper path of Figure 1. We control the output stride (ratio between input resolution and output resolution) by decreasing the stride of several convolutions from two to one in a bottom-up fashion and increasing the dilation rate instead. Contrary to what was proposed by [41], we do not incorporate dropout in our network, since it led to slightly worse segmentation results.

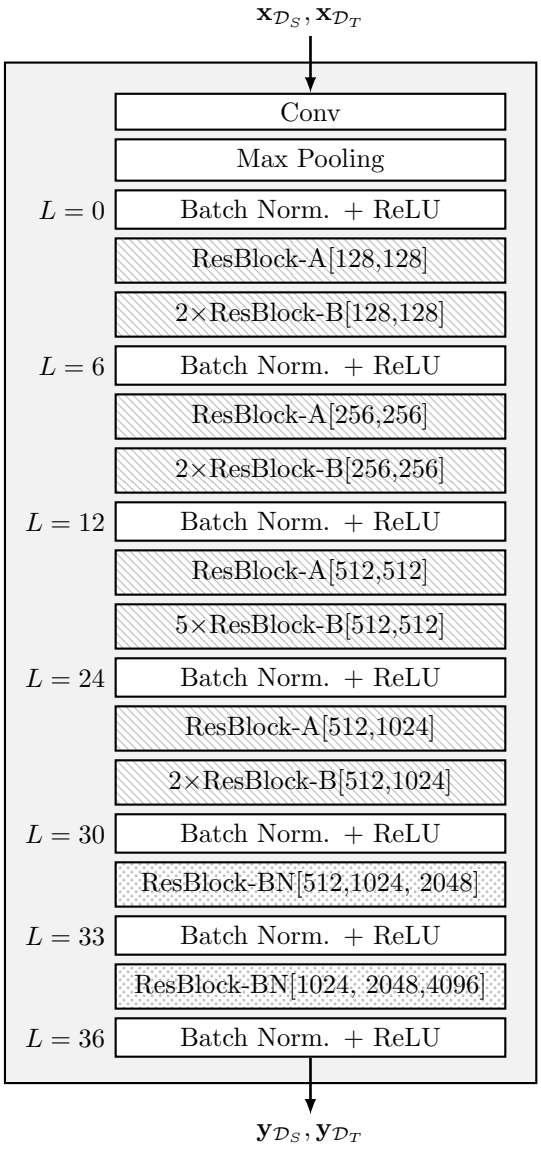

Figure 2. Architecture of the `WideResNet-38` **feature extractor** used in our semantic segmentation framework. The feature extractor is provided with the images $\mathbf{x}$ from either domain and extracts corresponding feature maps $\mathbf{y}$. The three types of ResBlocks are illustrated in Figure 3.

The image $\mathbf{x} \in \mathbb{G}^{H \times W \times C}$ with image pixel $x(i) \in \mathbb{G}$,

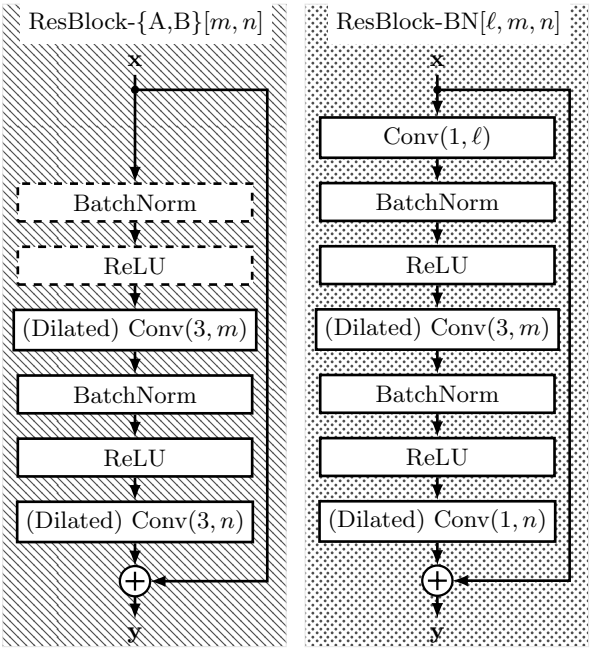

Figure 3. Structure on the **ResBlocks** used in the feature extractor. On the left, the ResBlock-A and -B are depicted. The only difference between these blocks is that the first BatchNorm and ReLU (dashed) are omitted for ResBlock-A. On the right side, the ResBlock-BN is depicted which uses an additional convolution layer and other kernel sizes for the first and last convolution. The parameters m and n denote the number of feature maps produced by the two convolutional layers in the ResBlock-A and -B and ℓ denotes the number of feature maps for the additional convolutional layer in the ResBlock-BN. The first number in the parentheses of each convolutional layer denotes the used kernel size.

where $\mathbb{G}$ is the set of gray values, i is the pixel index, H and W are the image height and width in pixels, and $C = 3$ is the number of color channels from set $\mathcal{C} = \{1, 2, 3\}$, is used as the input to the neural network. The feature extractor extracts 4096 feature maps $\mathbf{y}$ with a resolution dependent on the resolution of the input image. During the training the network is provided with random crops of the input image that all have the same resolution. In the standard approach a crop size of 700×700 is used, but due to the lower resolution of the KITTI and BDD100K dataset we adapted the crop size to 250×500. Naturally, this leads to a drop in absolute performance, which, for the purpose of this work, is not a considered problem. With the fixed crop size during the training the extracted feature maps have a resolution of $16\times$ 32 pixels.

The *segmentation head* then transforms these feature maps into output scores $\mathbf{P} \in \mathbb{I}^{H \times W \times |\mathcal{S}|}$, where $\mathcal{S} = \{1, 2, ..., 19\}$ denotes the set of classes with cardinality $|\mathcal{S}| = 19$ and $\mathbb{I} = [0, 1]$. These output scores can be thought of as a posterior probability (score) $P(i, s)$ for each class $s \in \mathcal{S}$ at pixel index i. After searching for the maximum, the segmentation mask $\mathbf{m} = \text{argmax}_{s \in \mathcal{S}} \mathbf{P} \in \mathcal{S}^{H \times W}$ is provided. To measure the accuracy of this map with respect to the ground truth labels, the intersection over union (IoU) is computed [12] as follows:

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}, \tag{1}$$

where TP, FP, and FN are the numbers of true positive, false positive, and false negative pixels for one corresponding label and segmentation mask pair. The mean IoU (mIoU) is then computed as the average IoU over a given set of images.

3.2. Domain Adaptation

Unsupervised domain adaptation in the context of semantic segmentation aims at keeping the mIoU at a high level independently from the used data set. As already mentioned, there are mainly two different approaches for domain adaptation for visual tasks. Either the images are adjusted to fit the distribution of the source domain before being entered into the network, or the network is trained to learn domain-invariant feature maps $\mathbf{y}$ that produce accurate segmentations on both datasets. Our system for domain adaptation is built upon the idea published by Ganin et al. [13] where they trained a neural network in a multi-task fashion. Similar to their work we use the *feature extractor* jointly for both tasks. After the *feature extractor*, the feature maps are split into two branches as seen in Figure 1. The *segmentation head* (upper part of the network) only receives the feature maps of the source domain $\mathcal{D}_S$ for which the labels are available. The *domain discriminator* (lower part of the network) receives both the source and target domain feature maps. After bringing the number of feature maps gradually down from 4096 to 2 by 1×1 convolutions, the domain discriminator tries to predict the domain from which the given feature maps came from, providing an output class map $\mathbf{s}^{(\text{D})} \in \mathbb{I}^{h_D \times w_D \times |\mathcal{D}|}$, with h_D and w_D being the height and width of the feature maps that are fed into the domain discriminator and $|\mathcal{D}|$ being the number of datasets used during training. The resolution of the feature maps is not reduced during the reduction of feature maps such that a patch-wise classification error can be used, which was proposed by [36] as a local adversarial loss in their GAN network. Just ahead of the domain discriminator there is the gradient reversal layer (GRL) [13]. During the forward-pass of the training this layer behaves as the identity function just passing the data through, while during the backward-pass the gradient gets reversed. Following Ganin et al. [13], this can be formalized as follows:

$$\begin{aligned} \text{Forward-pass: } & R_\lambda(\mathbf{y}) = \mathbf{y} \\ \text{Backward-pass: } & \frac{dR_\lambda}{d\mathbf{y}} = -\lambda(\tau)\mathbf{I}, \end{aligned} \tag{2}$$

where $\mathbf{I}$ is the identity matrix and λ is a weighting factor that is used in order to limit the influence of the domain discriminator. Here, $\mathbf{y}$ denotes the feature map that is passed through the layer $R_\lambda(\mathbf{y})$ during the forward pass. The gradients calculated by the backpropagation algorithm are denoted as $\frac{dR_\lambda}{d\mathbf{y}}$. The weighting factor λ is computed by

$$\lambda(\tau) = \frac{2}{1 + \exp\left(-10\ \alpha(\tau)\right)} - 1 \tag{3}$$

and changes for each epoch τ, with $\alpha(\tau)$ being defined as

$$\alpha(\tau) = \begin{cases} \frac{\tau}{\tau_{\max}} & \tau < \tau_{\max} \\ 1 & \tau \geq \tau_{\max}, \end{cases} \tag{4}$$

with $\tau_{\max}$ being the iteration number, from which on the influence of the domain discriminator should not be limited anymore. One iteration equals one processed minibatch.

3.3. Measures for the Domain Shift

With the employed approach, it is initially not unambiguous that the increase in accuracy in the source and target domain is actually due to the fact that domain-invariant features were learned. It would also be plausible that each additional branch after the feature extractor, which is trained on a different task, increases the magnitude of gradients in the feature extractor and thus enables more efficient learning, e.g., by preventing a vanishing gradient.

To analyze the hypothesis of domain-invariant features, we examined the feature maps of the feature extractor for the source and target dataset of both the network trained without the domain discriminator and the network trained with the domain discriminator. The feature map tensor after each activation layer has the size $(h_L \times w_L \times f_L)$, with h_L and w_L being the height and width of the feature maps in layer L, which itself is dependent on the input resolution that is different for all datasets used during inference. The term f_L denotes the number of feature maps in layer L. For the analysis we average the feature maps over h_L and w_L during inference, leaving us with a vector $\boldsymbol{\phi}_L$ of length f_L, that holds the mean activation value for each feature map in that layer. We can then compute the mean-squared error (MSE) distance between two of those vectors as follows:

$$D_{\text{MSE}}(\boldsymbol{\phi}_L^a, \boldsymbol{\phi}_L^b) = ||\boldsymbol{\phi}_L^a - \boldsymbol{\phi}_L^b||^2, \tag{5}$$

where a and b denote the different datasets the activation vector was produced by. We assume that the MSE D_{MSE} is related to the domain shift (higher D_{MSE} means higher domain shift), if dataset a comprises the training data and dataset b comprises the inference data.

4. Databases and Training

In this section, the databases used for training and evaluation are introduced. In the field of domain adaptation, there are several known databases for different applications. In the context of this work we limit ourselves to real-world (not simulated) automotive datasets. Subsequently, we describe the training method and parameters used for training of the segmentation network with and without domain adaptation.

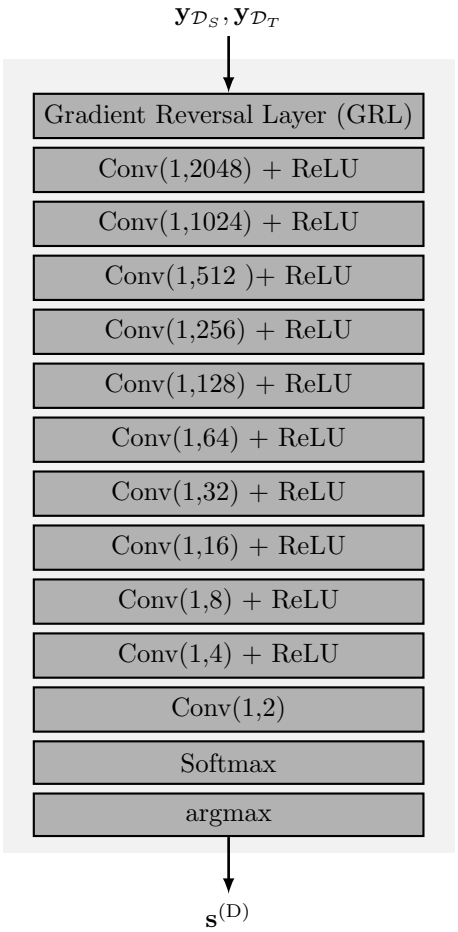

Figure 4. Architecture of the **gradient reversal layer (GRL)** with the **domain discriminator** used for domain adaptation in Figure 1. The network is provided with feature maps from both domains and tries to predict an output class map $\mathbf{s}^{(\mathrm{D})}$ that simply discriminates between both domains.

4.1. Databases

In this work we conducted experiments with three different automotive datasets, all providing labels for semantic segmentation and all being recorded under diverse conditions with a different sensor. For our experiments different subsets of the three datasets are used. Table 1 provides a description and composition of the individual subsets.

Cityscapes is used as the *source dataset* in all experiments. It consists of 2975 training images $\mathcal{D}_{\text{CS}}^{\text{train}}$ and 500

Table 1. The names and number of files for the subsets we used in our experiments. The KITTI train subsets are randomly sampled from the raw dataset.

Dataset	Data Subset	# Files	Labels Used
Cityscapes	$\mathcal{D}_{\text{CS}}^{\text{train}}$	2975	✓
	$\mathcal{D}_{\text{CS}}^{\text{val}}$	500	✓
	$\mathcal{D}_{\text{CS}}^{\text{test}}$	1525	✓
KITTI	$\mathcal{D}_{\text{KITTI}}^{\text{train,2975,c}}$	2975	
	$\mathcal{D}_{\text{KITTI}}^{\text{train,2975}}$	2975	
	$\mathcal{D}_{\text{KITTI}}^{\text{train,5000}}$	5000	
	$\mathcal{D}_{\text{KITTI}}^{\text{train,10000}}$	10000	
	$\mathcal{D}_{\text{KITTI}}^{\text{val}}$	200	✓
	$\mathcal{D}_{\text{KITTI}}^{\text{test}}$	200	✓
BDD100K	$\mathcal{D}_{\text{BDD}}^{\text{train}}$	7000	
	$\mathcal{D}_{\text{BDD}}^{\text{val}}$	1000	✓

validation images $\mathcal{D}_{\text{CS}}^{\text{val}}$, for each of which there are finely annotated labels. The 1525 test images $\mathcal{D}_{\text{CS}}^{\text{test}}$ can only be evaluated on the Cityscapes benchmark server. The dataset was recorded in 50 different cities in Germany and cities close to Germany using an automotive-grade camera with an OnSemi AR0331 sensor. The recordings were made under moderate to good weather conditions throughout spring, summer and autumn, so there are no recordings during bad weather or winter. All recordings were made during the day. The provided color images have a resolution of 1024×2048 pixels. The dataset is labeled with 19 classes that are used during training and inference [9].

KITTI is used as the *first target dataset* in all experiments. Originally, the KITTI dataset did not provide semantically segmented labels, but only labels for the object detection. However, a subset for which semantic labels have been available exists since 2015, which is called KITTI-15 [1]. There are 200 training images, which we use for validation $\mathcal{D}_{\text{KITTI}}^{\text{val}}$, and 200 test images $\mathcal{D}_{\text{KITTI}}^{\text{test}}$, but similar to the Cityscapes dataset there are only labels for the test images on the KITTI benchmark server. The labeling followed the same policy as in Cityscapes. The raw dataset consists of 48893 unlabeled images from which we randomly sampled images used for the unsupervised domain adaptation. For our initial experiments with two domains we sampled the same amount of images $\mathcal{D}_{\text{KITTI}}^{\text{train,2975,c}}$ as there are in the Cityscapes training set. For the subsequent experiments we sampled 10000 images $\mathcal{D}_{\text{KITTI}}^{\text{train,10000}}$, as well as 5000 images $\mathcal{D}_{\text{KITTI}}^{\text{train,5000}}$ and 2975 images $\mathcal{D}_{\text{KITTI}}^{\text{train,2975}}$, with the smaller sets being a subset of the bigger sets $\mathcal{D}_{\text{KITTI}}^{\text{train,10000}} \supset \mathcal{D}_{\text{KITTI}}^{\text{train,5000}} \supset \mathcal{D}_{\text{KITTI}}^{\text{train,2975}}$. The dataset was recorded in and around the German city Karlsruhe using a Point Grey FL2-14S3C-C camera with a Sony ICX267 sensor. The provided color images have a resolution of 375×1250 pixels [15].

Berkeley DeepDrive (BDD100K) is used as an *additional target dataset* in experiments with more than two domains. The dataset consists of 100000 videos each with a length of 40 seconds. The videos were gathered via crowd sourcing, which means that volunteers contributed the videos recorded by their dashcam. Therefore, no single sensor can be specified for this dataset. The videos were recorded on the entire territory of the U.S.A. Cities, highways and rural regions were captured under various weather conditions and times of the day. The BDD100K also offers a subset with pixel labels, as with the KITTI dataset. 8000 images were annotated, 7000 of them as training images $\mathcal{D}_{\text{BDD}}^{\text{train}}$ and 1000 validation images $\mathcal{D}_{\text{BDD}}^{\text{val}}$. Although there is a test set with 2000 images, no labels are provided and there is currently no evaluation server available. The labeling also followed the same policy as in Cityscapes. The images have a resolution of 720×1280 pixels [44].

4.2. Training and Domain Adaptation Stages 1 & 2

The training of the segmentation network follows a *two-stage* training based on [7], which is also described in [4] and [26]. Due to the different resolutions of the used datasets the random scaling is performed in a range of 100 % to 200 % and the crop size is set to 250×500. The images from all data sets were cropped to an aspect ratio of 2:1 before training. The domain discriminator is initialized with random weights. In the discriminator network (index "D") we use a dropout value $p_{\text{D}} = 0.7$ for stages 1 and 2. For stage 1 the weighting factor λ follows (4) with $\tau_{\max} = 18000$. During stage 2 we use a weighting factor $\lambda = 1$. The initial learning rate for the domain discriminator used for stage 1 is $\eta_{\text{D},0} = 0.01$, which is halved in the 2nd stage to $\eta_{\text{D},0} = 0.005$. The learning rate schedule follows a power scheduling:

$$\eta_{\text{D}}(\tau) = \frac{\eta_{\text{D},0}}{(1 + 10\,\gamma(\tau)^{0.75})}, \tag{6}$$

where τ is the iteration index and $\gamma = \frac{\tau}{\tau_{\text{end}}}$ with τ_{end} being the maximum number of iterations. The initial learning rate for the feature extractor (index "F") used for stage 1 is $\eta_{\text{F},0} = 0.001$, which is halved in stage 2 to $\eta_{\text{F},0} = 0.0005$. The learning rate follows a polynomial scheduling:

$$\eta_{\text{F}}(\tau) = \eta_{\text{F},0}(1-\gamma)^c, \tag{7}$$

with $c = 0.9$. The learning rate for the weights of the segmentation head follows the same schedule, but is multiplied by a factor of 10 and the learning rate for the bias weights in the segmentation head is multiplied by 20. Both the feature extractor and segmentation head also use L2 regularization with a weighting value of $1 \cdot 10^{-4}$. For stage 1 we chose $\tau_{\text{end}} = 90000$ and for stage 2 we chose $\tau_{\text{end}} = 120000$. We use a momentum of $\beta = 0.9$. The segmentation head and the domain discriminator both employ a cross entropy loss during training.

Table 3. **Validation set mIoU** for multiple datasets without and with unsupervised domain adaptation. The results are obtained by networks trained for stages 1 and 2. Only $\mathcal{D}_{\text{CS}}^{\text{train}}$ is used as the source domain $\mathcal{D}_S$ and all other training subsets are used as the unlabeled target domain $\mathcal{D}_T$.

Training Stage	Trained on	Evaluation Results (mIoU)		
		$\mathcal{D}_{\text{CS}}^{\text{val}}$	$\mathcal{D}_{\text{KITTI}}^{\text{val}}$	$\mathcal{D}_{\text{BDD}}^{\text{val}}$
1	$\mathcal{D}_{\text{CS}}^{\text{train}}$	51.2 %	40.1 %	25.2 %
	$\mathcal{D}_{\text{CS}}^{\text{train}} \cup \mathcal{D}_{\text{KITTI}}^{\text{train,2975}} \cup \mathcal{D}_{\text{BDD}}^{\text{train}}$	54.7 %	46.4 %	48.1 %
	$\mathcal{D}_{\text{CS}}^{\text{train}} \cup \mathcal{D}_{\text{KITTI}}^{\text{train,5000}} \cup \mathcal{D}_{\text{BDD}}^{\text{train}}$	**55.3 %**	**49.0 %**	**48.9 %**
	$\mathcal{D}_{\text{CS}}^{\text{train}} \cup \mathcal{D}_{\text{KITTI}}^{\text{train,10000}} \cup \mathcal{D}_{\text{BDD}}^{\text{train}}$	54.1 %	47.5 %	47.3 %
2	$\mathcal{D}_{\text{CS}}^{\text{train}}$	51.5 %	40.5 %	26.1 %
	$\mathcal{D}_{\text{all}}^{\text{train}} = \mathcal{D}_{\text{CS}}^{\text{train}} \cup \mathcal{D}_{\text{KITTI}}^{\text{train,5000}} \cup \mathcal{D}_{\text{BDD}}^{\text{train}}$	**57.2 %**	**56.7 %**	**49.9 %**

Table 2. **mIoU** for Cityscapes and KITTI **validation sets** without (1st row) and with (2nd row) unsupervised domain adaptation. The results for networks trained only for stage 1:

Trained on	Evaluation Results (mIoU)	
	$\mathcal{D}_{\text{CS}}^{\text{val}}$	$\mathcal{D}_{\text{KITTI}}^{\text{val}}$
$\mathcal{D}_{\text{CS}}^{\text{train}}$	51.2 %	40.1 %
$\mathcal{D}_{\text{CS}}^{\text{train}} \cup \mathcal{D}_{\text{KITTI}}^{\text{train,2975,c}}$	**53.6 %**	**46.3 %**

5. Experiments and Results

In this section we will evaluate the experiments on domain adaptation. We will start with experiments on two domains, using Cityscapes as our source and KITTI as our unlabeled target domain. In the subsequent experiments we will use BDD100K as an additional unlabeled target domain. Finally, an analysis of the learned domain-invariant feature maps will follow.

5.1. Discussion of Domain Adaptation

Our initial experiments were conducted using Cityscapes as our source Domain $\mathcal{D}_S$ and KITTI as out target domain $\mathcal{D}_T$. The results are shown in Table 2. The effect of the unsupervised domain adaptation can be clearly seen by the mIoU increase of 6.2 % absolute on KITTI. A surprising result is that not only in the target domain, but even in the source domain we obtain an increase in mIoU (2.4 % absolute)[1]. We cast both improvements on the mere fact of additional data for training. The important facts are: Firstly, this additional training material does not require labels and secondly, it may even stem from a different domain.

Intuitively one would use the same amount of images from each dataset per minibatch (1:1:1), but preliminary experiments have shown that a ratio of (1:2:1), i.e., two images from the KITTI data set with one image from Cityscapes and one from BDD100K, yields better results on the KITTI and BDD100K validation sets. We also observed that it is advantageous for the performance on the target domains if the domain discriminator has to distinguish between all domains and not only between source and target domain.

The results for multiple datasets are presented in Table 3, showing the mIoU for different validation sets. The first row serves as our baseline, as these results are obtained with a segmentation network that was solely trained on the Cityscapes training set $\mathcal{D}_{\text{CS}}^{\text{train}}$ and has not seen images from any other domain. *It can be seen that the mIoU for the network with the domain adaptation increases for the source domain (Cityscapes) as well as for the target domains* (KITTI and BDD100K), implying that a multi-task gain exists and the feature extractor network is forced to extract domain-invariant, better generalizing features.

It can also be seen that the usage of more unlabeled data further improves performance but has a sweet spot and decreases again for too much unlabeled data. The sweet spot is reached for 5000 images on KITTI, while both 2975 and 10000 images yield a somewhat lower performance. With the best approach from stage 1 a stage 2 training was performed, which further increased the mIoU on all validation sets. The final network yields a performance on $\mathcal{D}_{\text{KITTI}}^{\text{val}}$, which is nearly en par with results on $\mathcal{D}_{\text{CS}}^{\text{val}}$. We obtain a gain of by 16.2 % mIoU absolute on the KITTI validation set by unsupervised domain adaptation and moreover we obtain a gain of 5.7 % mIoU on the Cityscapes validation set. We also see a large gain on the BDD100K subset $\mathcal{D}_{\text{BDD}}^{\text{val}}$ where we obtain an increase of 23.8 % mIoU absolute. For the sake of clarity, the dataset that is used for training with unsupervised domain adaptation in stage 2 is referred to as $\mathcal{D}_{\text{all}}^{\text{train}}$.

For our final evaluation we tested the final stage 2 network against the stage 2 baseline on the two *test sets* $\mathcal{D}_{\text{CS}}^{\text{test}}$ and $\mathcal{D}_{\text{KITTI}}^{\text{test}}$. The results are presented in Table 4. *We obtain a gain of* 15.4 % *mIoU absolute on* $\mathcal{D}_{\text{KITTI}}^{\text{test}}$ *after unsupervised domain adaptation, and in addition we also obtain a gain of* 3.8 % *mIoU absolute on the in-domain data*

[1] It has to be noted that the KITTI subset $\mathcal{D}_{\text{KITTI}}^{\text{train,2975,c}}$ used for this experiment included images from the *calibration* and *person* categories of the KITTI dataset. However, for the following experiments we use KITTI subsets, which exclude these categories as they are originally not intended for training.

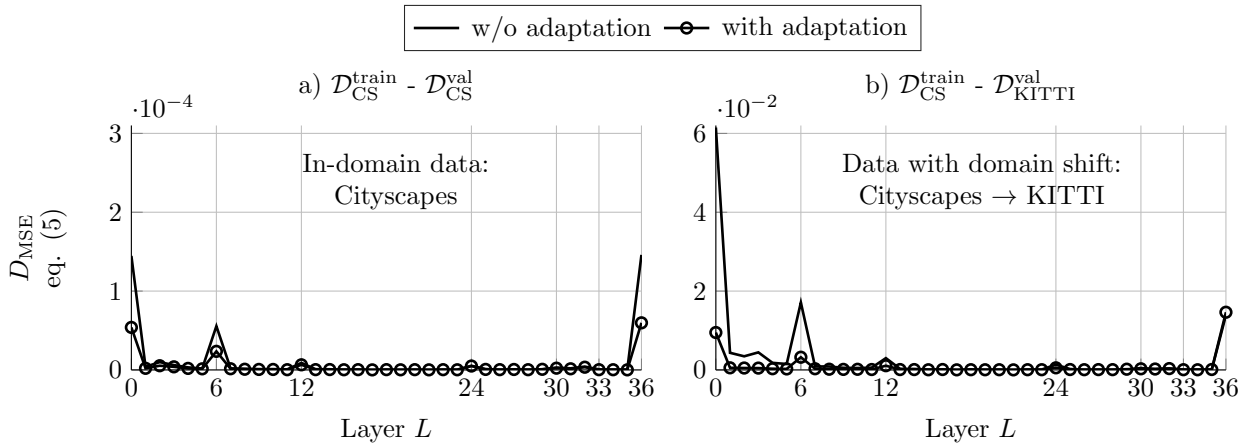

Figure 5. Results of the **feature map activation analysis** for a) in-domain data and b) data with a domain shift. Results are shown for the non-adapted network (no markers) and the adpated network (markers ∘). Please note that the scaling of the left y-axis is two orders of magnitude smaller to visualize the effect also for the in-domain data.

Table 4. **Test set mIoU** for Cityscapes and KITTI without and with unsupervised domain adaptation. The results are for networks trained for stages 1 and 2.

Trained on	**Evaluation Results (mIoU)**	
	$\mathcal{D}_{\text{CS}}^{\text{test}}$	$\mathcal{D}_{\text{KITTI}}^{\text{test}}$
$\mathcal{D}_{\text{CS}}^{\text{train}}$ (w/o adaptation)	56.0 %	44.1 %
$\mathcal{D}_{\text{all}}^{\text{train}}$ (with adaptation)	**59.8 %**	**59.5 %**

$\mathcal{D}_{\text{CS}}^{\text{test}}$. On the KITTI test set our domain-adapted semantic segmentation achieved a surprisingly good 5th rank with an mIoU of 59.5 % and without having used any KITTI labels during training [2].

5.2. Further Analysis

The results presented above allow for two hypotheses. The increase in accuracy could either be due to the fact that the gradients of deeper layers in the feature extractor are larger by the multi-task approach, or we actually learn domain-invariant features. We used the settings with two domains to analyze the features that are extracted by both a network trained without and a network trained with domain adaptation. We extracted activation vectors as described in Section 3.3 for the Cityscapes subsets $\mathcal{D}_{\text{CS}}^{\text{train}}$ and $\mathcal{D}_{\text{CS}}^{\text{val}}$ and for the KITTI subset $\mathcal{D}_{\text{KITTI}}^{\text{val}}$ for both network variants. Afterwards, we computed the D_{MSE} (5) for all vectors from the *validation* sets with their corresponding vector from the Cityscapes *train* set. The results are depicted in Figure 5. It can be seen that the difference for the network trained without domain adaptation is higher than for the network trained with domain adaptation. This holds even true for data from the same domain as can be seen in subfigure a), even so this effect is only noticeable on a scale which is smaller by two orders of magnitude.

What is immediately noticeable is that the differences in activations are particularly large for the layers $L \in \{0, 6, 12, 24, 36\}$. If one compares these layers with Figure 2, then we realize that these are exactly the layers which, in contrast to the residual units, are not extended by a residual path. *It seems that these layers contain domain-specific knowledge in particular.*

6. Conclusions

In this work we presented an unsupervised domain adaptation technique to semantic segmentation, which has advantages over style transfer approaches with generative models, as it does not modify the input data to the network. This is particularly important from a safety point of view, as it cannot be said with certainty how a segmentation network reacts to possible artifacts in generated images. We have shown that this approach is also very convenient, as existing networks can easily be extended by a domain discriminator and in fact domain-invariant, better generalizing features are learned. In consequence we were able to show that with our technique *both the source and the target domain take profit from additional unlabeled data of the target domain.* On the target domain KITTI test set, *performance increases by an impressive* 15 % *mIoU absolute.* In consequence, our domain-adapted semantic segmentation achieved a surprisingly good 5th rank on the KITTI test set, without having used any KITTI labels during training.

7. Acknowledgment

The authors gratefully acknowledge support of this work by Volkswagen Group Research, Wolfsburg, Germany.

[2] We could not test on $\mathcal{D}_{\text{BDD}}^{\text{test}}$, since an evaluation server does not seem to be set up yet and no test labels are provided.

References

[1] H. Abu Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes. *International Journal of Computer Vision*, 126(9):961–972, Sept. 2018. 6

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

[3] S. Bak, P. Carr, and J.-F. Lalonde. Domain Adaptation Through Synthesis for Unsupervised Person Re-identification. In *Proc. of ECCV*, pages 189–205, Munich, Germany, Sept. 2018. 2

[4] J.-A. Bolte, A. Bär, D. Lipinski, and T. Fingscheidt. Towards Corner Case Detection for Autonomous Driving. *arXiv*, (1902.09184), Feb. 2019. 3, 6

[5] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised Pixel-Level Domain Adaptation With Generative Adversarial Networks. In *Proc. of CVPR*, pages 3722–3731, Honolulu, HI, USA, July 2017. 2

[6] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain Separation Networks. In *Proc. of NIPS*, pages 343–351, Barcelona, Spain, Dec. 2016. 2

[7] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv*, (1706.05587), June 2017. 3, 6

[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. of CVPR*, pages 3339–3348, Salt Lake City, UT, USA, June 2018. 3

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of CVPR*, pages 3213–3223, Las Vegas, NV, USA, June 2016. 2, 6

[10] G. Csurka. Domain Adaptation for Visual Applications: A Comprehensive Survey. *arXiv*, (1702.05374), Mar. 2017. 2

[11] W. Deng, L. Zheng, Q. Ye, G. Kang, Y. Yang, and J. Jiao. Image-Image Domain Adaptation With Preserved Self-Similarity and Domain-Dissimilarity for Person Re-Identification. In *Proc. of CVPR*, pages 994–1003, Salt Lake City, UT, USA, June 2018. 2

[12] M. Everingham, S. M. Eslami, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision (IJCV)*, 111(1):98–136, Jan. 2015. 4

[13] Y. Ganin and V. Lempitsky. Unsupervised Domain Adaptation by Backpropagation. In *Proc. of ICML*, pages 1180–1189, Lille, France, July 2015. 2, 3, 4

[14] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-Adversarial Training of Neural Networks. *The Journal of Machine Learning Research (JMLR)*, 17(1):2096–2030, 2016. 2

[15] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision Meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, Aug. 2013. 2, 6

[16] B. Gong, K. Grauman, and F. Sha. Connecting the Dots With Landmarks: Discriminatively Learning Domain-Invariant Features for Unsupervised Domain Adaptation. In *Proc. of ICML*, pages 222–230, Atlanta, Georgia, USA, June 2013. 2

[17] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *Proc. of ICML*, pages 1989–1998, Stockholm, Sweden, July 2018. 3

[18] J. Hoffman, D. Wang, F. Yu, and T. Darrell. FCNs in the Wild: Pixel-Level Adversarial and Constraint-Based Adaptation. *arXiv*, (1612.02649), Dec. 2016. 3

[19] W. Hong, Z. Wang, M. Yang, and J. Yuan. Conditional Generative Adversarial Network for Structured Domain Adaptation. In *Proc. of CVPR*, pages 1335–1344, Salt Lake City, UT, USA, June 2018. 3

[20] L. Hu, M. Kan, S. Shan, and X. Chen. Duplex Generative Adversarial Network for Unsupervised Domain Adaptation. In *Proc. of CVPR*, pages 1498–1507, Salt Lake City, UT, USA, June 2018. 2

[21] H. Huang, Q. Huang, and P. Krahenbuhl. Domain Tansfer Through Deep Activation Matching. In *Proc. of ECCV*, pages 590–605, Munich, Germany, Sept. 2018. 3

[22] S.-W. Huang, C.-T. Lin, S.-P. Chen, Y.-Y. Wu, P.-H. Hsu, and S.-H. Lai. AugGAN: Cross Domain Adaptation With GAN-based Data Augmentation. In *Proc. of ECCV*, pages 718–731, Munich, Germany, Sept. 2018. 3

[23] G. Kang, L. Zheng, Y. Yan, and Y. Yang. Deep Adversarial Attention Alignment for Unsupervised Domain Adaptation: The Benefit of Target Expectation Maximization. In *Proc. of ECCV*, pages 401–416, Munich, Germany, Sept. 2018. 2

[24] E. Kodirov, T. Xiang, Z. Fu, and S. Gong. Unsupervised Domain Adaptation for Zero-Shot Learning. In *Proc. of ICCV*, pages 2452–2460, Las Condes, Chile, Dec. 2015. 2

[25] H. Li, S. Jialin Pan, S. Wang, and A. C. Kot. Domain Generalization With Adversarial Feature Learning. In *Proc. of CVPR*, pages 5400–5409, Salt Lake City, UT, USA, June 2018. 2

[26] J. Löhdefink, A. Bär, N. M. Schmidt, F. Hüger, P. Schlicht, and T. Fingscheidt. GAN vs. JPEG2000 Image Compression for Distributed Automotive Perception: Higher Peak SNR Does Not Mean Better Semantic Segmentation. *arXiv*, (1902.04311), Feb. 2019. 6

[27] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto. Unified Deep Supervised Domain Adaptation and Generalization. In *Proc. of ICCV*, pages 5715–5725, Venice, Italy, Oct. 2017. 2

[28] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim. Image to Image Translation for Domain Adaptation. In *Proc. of CVPR*, pages 4500–4509, Salt Lake City, UT, USA, June 2018. 3

[29] K. S. Neethu and D. Varghese. An Incremental Semi-supervised Approach for Visual Domain Adaptation. In *Proc. of ICCSP*, pages 1343–1346, Chennai, India, Apr. 2017. 2

[30] P. O. Pinheiro. Unsupervised Domain Adaptation With Similarity Learning. In *Proc. of CVPR*, pages 8004–8013, Salt Lake City, UT, USA, June 2018. 2

[31] Z. Ren and Y. Jae Lee. Cross-Domain Self-Supervised Multi-Task Feature Learning Using Synthetic Imagery. In *Proc. of CVPR*, pages 762–771, Salt Lake City, UT, USA, June 2018. 2

[32] A. Rozantsev, M. Salzmann, and P. Fua. Residual Parameter Transfer for Deep Domain Adaptation. In *Proc. of CVPR*, pages 4339–4348, Salt Lake City, UT, USA, June 2018. 2

[33] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada. Maximum Classifier Discrepancy for Unsupervised Domain Adaptation. In *Proc. of CVPR*, pages 3723–3732, Salt Lake City, UT, USA, June 2018. 3

[34] S. Sankaranarayanan, Y. Balaji, A. Jain, S. Nam Lim, and R. Chellappa. Learning From Synthetic Data: Addressing Domain Shift for Semantic Segmentation. In *Proc. of CVPR*, pages 3752–3761, Salt Lake City, UT, USA, June 2018. 3

[35] S. Saxena, S. Pandey, and P. Khanna. A Semi-Supervised Domain Adaptation Assembling Approach for Image Classification. *Pattern Analysis and Applications*, 21(3):813–827, Aug. 2018. 2

[36] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from Simulated and Unsupervised Images Through Adversarial Training. In *Proc. of CVPR*, pages 2242–2251, Honolulu, HI, USA, July 2017. 4

[37] R. Shu, H. Bui, H. Narui, and S. Ermon. A DIRT-T Approach to Unsupervised Domain Adaptation. In *Proc. of ICLR*, Vancouver, Canada, Apr. 2018. 2

[38] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial Discriminative Domain Adaptation. In *Proc. of CVPR*, Honulu, HI, USA, July 2017. 2

[39] H. Venkateswara, S. Chakraborty, and S. Panchanathan. Deep-Learning Systems for Domain Adaptation in Computer Vision: Learning Transferable Feature Representations. *IEEE Signal Processing Magazine*, 34(6):117–129, Nov. 2017. 2

[40] M. Wang and W. Deng. Deep Visual Domain Adaptation: A Survey. *arXiv*, (1802.03601), May 2018. 1, 2

[41] Z. Wu, C. Shen, and A. van den Hengel. Wider or Deeper: Revisiting the ResNet Model for Visual Recognition. *arXiv*, Nov. 2016. 3

[42] L. Yan, B. Fan, S. Xiang, and C. Pan. Adversarial Domain Adaptation With a Domain Similarity Discriminator for Semantic Segmentation of Urban Areas. In *Proc. of ICIP*, pages 1583–1587, Athens, Greece, Oct. 2018. 2

[43] T. Yao, Y. Pan, C.-W. Ngo, H. Li, and T. Mei. Semi-Supervised Domain Adaptation With Subspace Learning for Visual Recognition. In *Proc. of CVPR*, pages 2142–2150, Boston, MA, USA, June 2015. 2

[44] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell. BDD100K: A Diverse Driving Video Database With Scalable Annotation Tooling. *arXiv*, (1805.04687), Aug. 2018. 2, 6

[45] Y. Zhang, Z. Qiu, T. Yao, D. Liu, and T. Mei. Fully Convolutional Adaptation Networks for Semantic Segmentation. In *Proc. of CVPR*, pages 6810–6818, Salt Lake City, UT, USA, June 2018. 3

[46] H. Zhao, S. Zhang, G. Wu, J. M. F. Moura, J. P. Costeira, and G. J. Gordon. Adversarial Multiple Source Domain Adaptation. In *Proc. of NIPS*, pages 8568–8579, Montreal, Canada, Dec. 2018. 2