Semantic Segmentation with Unsupervised Domain Adaptation Under Varying Weather Conditions for Autonomous Vehicles

Özgür Erkent[1] and Christian Laugier[1]

Abstract—**Semantic information provides a valuable source for scene understanding around autonomous vehicles in order to plan their actions and make decisions; however, varying weather conditions reduce the accuracy of the semantic segmentation. We propose a method to adapt to varying weather conditions without supervision, namely without labeled data. We update the parameters of a deep neural network (DNN) model that is pre-trained on the known weather condition (source domain) to adapt it to the new weather conditions (target domain) without forgetting the segmentation in the known weather condition. Furthermore, we don't require the labels from the source domain during adaptation training. The parameters of the DNN are optimized to reduce the distance between the distribution of the features from the images of old and new weather conditions. To measure this distance, we propose three alternatives: W-GAN, GAN and maximum-mean discrepancy (MMD). We evaluate our method on various datasets with varying weather conditions. The results show that the accuracy of the semantic segmentation is improved for varying conditions after adaptation with the proposed method.**

Index Terms—**Intelligent Transportation Systems; Semantic Scene Understanding; Learning and Adaptive Systems**

I. Introduction

Autonomous vehicles need to recognize their surroundings in varying weather conditions to be able to drive safely and navigate successfully. We focus on the semantic understanding of the environment of a vehicle in varying weather conditions by using the images from the RGB camera without supervision during adaptation training.

Deep Neural Networks (DNNs) proved to have a high accuracy and fast implementation in semantic segmentation such as MobileNetV2 [1] and SegNet [2]. However, a tremendous amount of labeled data would be necessary for each weather condition with supervised learning; therefore, we perform unsupervised domain adaptation (UDA) so that labeled images are not required for adaptation. The old weather condition is called as the "*source domain*" in UDA, while the new weather condition is called as the "*target domain*".

Although it would be possible to detect the weather type and use a separate model for each condition, the detection and the selection of the appropriate segmentation model would be time consuming and require extra computation power. Furthermore, the weather can vary during the same ride; therefore, we use the same single recognition model for both domains. Additionally, it may not always be possible to have access to the image-label pairs of the "*source domain*". Therefore we use random RGB images from both domains for adaptation training (Fig. 1).

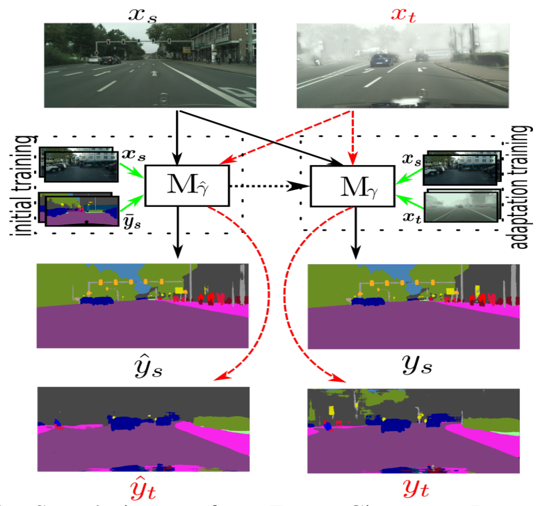


Fig. 1: Sample images from Foggy Cityscapes Dataset (CS) [3]. The semantic segmentation model has the architecture of MobileNetV2. More detailed segmentation is available after segmentation with adapted model M_γ while the segmentation for the old condition x_s remains intact. x_t: new weather condition, $\bar{y}_s$: Ground truth for old weather, $M_{\hat{\gamma}}$: Initial segmentation model, $\hat{y}_s$: segmentation of old weather with initial model, y_s: segmentation of old weather with adapted model, $\hat{y}_t$: segmentation of new weather with initial model, y_t: segmentation of new weather with adapted model.

We assume that we already have a pre-trained DNN model and we modify the parameters of the DNN in the initial layers. We want to obtain similar features in distribution in the initial layers of the network for different weather conditions with a high estimation accuracy for segmentation in both domains. To measure the distance in distribution in between the features of different weather conditions, we investigate three alternative distance measures: Wasserstein distance [4], Jensen-Shannon divergence [5] and maximum mean discrepancy (MMD) [6]. Furthermore, to sustain the accuracy for the segmentation similar to the original DNN, we use the difference between the features of the "*source domain*" obtained from the original DNN and the modified DNN as an additional loss for the optimization. Briefly, the advantages and contributions of the proposed method can be listed as:

- Accurate semantic segmentation for different weather

[1] Authors are with Univ. Grenoble Alpes, INRIA, Chroma Team, France INRIA, Rhône-Alpes, France. Correspondence: `ozgur.erkent@inria.fr`

conditions can be achieved with the same model;
- The UDA training does not require the labeled images of the *source domain* or any pairing between the source and target domain images;
- The same DNN architecture is used for both of the domains after the parameters are modified, which results in the same inference time with the original model.

In Section. II, we discuss the literature related to the domain adaptation. In Section. III, we explain our method with three alternative distance measures. In Section. IV, we evaluate our method with three DNN models and various datasets. Finally, we conclude the paper with a summary.

II. Related Work

We will review the studies related to UDA with a focus on semantic segmentation, object detection and image classification between different weather conditions and different domains such as adaptation from simulation to real world.

A group of studies related to UDA use the incremental adaptation approach which relies on the assumption that the conditions change continuously and the estimation model should adapt to this new condition incrementally. For example Dai *et al.* [7] use different times of the twilight to adapt to darkness. First a less dark time of twilight is selected and the images are labeled with the previously trained network on daytime images. The estimated labels from the network are considered as the *new ground truth* and the network is retrained with a mixture of these *new ground truth* and previous hand labeled ground truth images. This process is applied for each incremental dark condition until the network adapts to darkness. In another study, Wulfmeier *et al.* [8] use the incremental adaptation for darkness with a generative adversarial network (GAN) [5]. The distribution of the target domain features are adapted to the source domain features at each step. One problem with such approaches is that the incremental data collection for each weather condition may not be always feasible for autonomous vehicles.

The invariance of the sensors to weather conditions is also used for accurate classification. Valada *et al.* [9] propose a network architecture that can adapt to different modalities of the sensors and integrate different networks within the same architecture. Kim *et al.* [10] use the invariance of laser range sensors to different weather conditions and train a network to fuse the Lidar and RGB camera sensor data. In this way, they obtain an accurate segmentation of the scene in different weather conditions. Wu *et al.* [11] adapt Squeeze Segmentation for road-object segmentation by using Lidar sensor data for synthetic and real data. Chen *et al.* [12] use the depth information of the source domain to transform the source domain data into the target domain and the network is trained with the transformed images. In another study, Erkent *et al.* [13] obtain the semantic 2D maps of the environment by combining the occupancy grids with the semantic segmented images. These approaches require the usage of multiple calibrated sensors and a depth sensor whereas we are interested in adaptation to data from a single sensor under a new condition.

Some studies use special sensors to mimic the weather conditions to have similar images at the input of the network. For example, Porav *et al.* [14] tackle the problem of domain adaptation by generating input images that are similar to each other in both domains. They obtain a dataset with a special stereo camera that has water droplets on its lenses for simulating appearances of rainy and un-rainy weather conditions and train a network by using these image pairs to de-rain the images in rainy weather. It should be noted that it would be costly to produce a special generator for each weather condition. On the other hand, some studies modify the networks to have similar distributions for both domains in the output. For example, Vu *et al.* [15] apply GAN training [5] on the entropy image of the output, where the distribution of the entropies of the output are required to be similar. Some studies ([16], [17]) also measure the similarity between the distributions of the outputs of the network for source and the target domain by using a GAN [5] and update the network to make these distributions similar to each other. To be able to measure the similarity in the network output distribution, the outputs generally need to be converted into a higher-dimensional space, which requires additional computational time and power.

The similarity between the distributions of the features of the source and target domain are also considered. In an early work in UDA with DNNs, Tzeng *et al.* [18] adapt a DNN by inserting a generator into the network and train it by using the MMD [19] to measure the discrepancy between the source and target domain features. Long *et al.* [20] propose UDA in DNNs by updating the parameters of the later layers of the network with multiple MMD kernels. One of the earliest works that used GANs [5] for domain adaptation to find the similarity between source and target domain data at different levels of the network was proposed by Ganin *et al.* [21]. The usage of GANs for domain adaptation is applied in semantic segmentation where the discriminators are deployed at various locations of the network to measure the differences between the distributions of the features of source and target domains [22], [23], [24]. Bolte *et al.* [25] use GANs and obtain a single model for both domains; however, the adaptation is applied between datasets with similar appearances under similar weather conditions and all parameters of the network are updated by using the source domain labels. Adaptation is not limited to GANs, Peng *et al.* [26] use Wasserstein GANs (W-GANs) [4] with penalty gradient [27] with an additional generator in the network to measure the distribution similarity.

The similarity can also be measured as a combination of input, output and feature distributions. For example, Cycada [28] transforms the data in between two domains by measuring the distributions of both input and features of the DNN by using GAN [5] and uses generators to transform the data. However, the complexity of the network due to the number of discriminators and generators can be problematic for large DNNs. Other studies which measure the similarities between the features and the outputs are made by [29] and [30] where the main focus is the adaptation for semantic segmentation from simulation into the real world.

Our approach is different from the mentioned methods in at least two aspects. First, we use the self-supervision of the features from the source domain similar to Wasserstein auto-encoders (WAE) [31]; therefore, the model does not

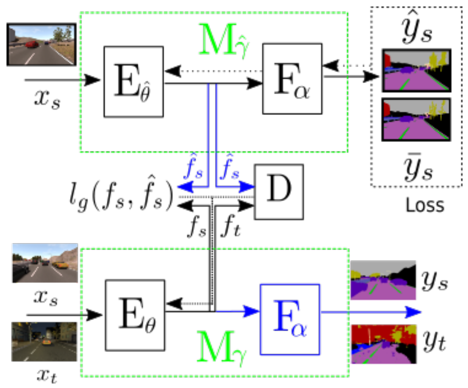


Fig. 2: Adaptation with self-supervision of features. Green dotted boxes represent the segmentation model. Dotted black lines represent the back-propagation. Blue lines represent the process without back-propagation.

only achieve high accuracy for the target domain, but also it does not forget how to classify the source domain. We demonstrate this by reporting the accuracies of the same model with different weather conditions which is in accordance with the proposal of the adaptation to the change of weather conditions during the same ride. Secondly, since we use self-supervision of features, we don't use source domain labels during adaptation training.

III. Domain Adaptation

In UDA, $\mathcal{S} = \{(\mathbf{x}_s^i, \bar{y}_s^i)\}_{i=1}^{n_s}$ is called as "*source domain*" and $\mathcal{X}_t = \{(\mathbf{x}_t^j)\}_{j=1}^{n_t}$ is called as "*target domain*" where n_s and n_t are the number of samples. We consider that only source data $\mathcal{X}_s = \{(\mathbf{x}_s^i)\}_{i=1}^{n_s}$, target data $\mathcal{X}_t$ and previously trained DNN model $M_{\hat{\gamma}}$ are available. $M_{\hat{\gamma}}$ estimates the labels $\hat{y}_s^i$ to minimize the source cost $c_s(M_{\hat{\gamma}}) = \Pr_{(\mathbf{x},\bar{y})\sim p}[M_{\hat{\gamma}}(x) \neq \bar{y}]$. Probability distributions of target and source domain are $\Pr_p$ and $\Pr_q$. We aim to obtain a model M_γ such that the target cost $c_t(M_\gamma) = \Pr_{(\mathbf{x},\bar{y})\sim q}[M_\gamma(x) \neq \bar{y}]$ and source cost c_s simultaneously minimize. We assume that the data sampled from target distribution contain necessary and sufficient information to segment the images in target domain similar to source domain in accordance with other UDA methods. An illustration of the adaptation method is shown in Fig. 2.

The initial layers of the network $M_{\hat{\gamma}}$ are labeled as $E_{\hat{\theta}}$ and the remaining layers as F_α such that $M_{\hat{\gamma}}(x) = F_\alpha(E_{\hat{\theta}}(x))$. γ, θ, $\hat{\gamma}$, $\hat{\theta}$ and α are the parameters of the corresponding networks or network layers where $\hat{\gamma}$, $\hat{\theta}$ are fixed. We optimize the parameters of the network E so that the input features of F_α layer from the target domain become similar in distribution to the source domain. The target cost to be minimized becomes $c_t(E_\theta(\mathbf{x}_t) = \mathbf{f}_t) = \Pr_{(\mathbf{f},y)\sim q}[F_\alpha(\mathbf{f}) \neq y]$ where θ are the parameters of the E to be optimized. This results in a reduction of the parameters to be optimized from $||\gamma||$ to $||\theta||$ which would result in less training time and memory.

We assume that the capacity of E is sufficient enough to produce features that are similar in both domains and to produce the features for an accurate classification by the network F. We do not make any changes in the structure of E, only update the parameters from $\hat{\theta}$ to θ. The following is used to optimize θ:

$$\inf_{\theta\in\Theta} D(\Pr_{(\hat{\mathbf{f}}_s)\sim p}, \Pr_{(E_\theta(\mathbf{x}_t))\sim q}) + \lambda l_g(\hat{\mathbf{f}}_s, E_\theta(\mathbf{x}_s)) \tag{1}$$

where Θ is the set of all parameters for θ, $E_\theta(\mathbf{x}_t^i) = \mathbf{f}_t^i$ are the target domain features obtained from updated network. D measures the distance between the distributions. $\lambda > 0$ is a regularization hyperparameter and $l_g(\hat{\mathbf{f}}_s, E_\theta(\mathbf{x}_s)) = ||\hat{\mathbf{f}}_s - E_\theta(\mathbf{x}_s)||_2^2$ is the self-supervision loss that measures the distance between the source domain features $\hat{\mathbf{f}}_s^i = E_{\hat{\theta}}(\mathbf{x}_s^i)$ and $\mathbf{f}_s^i = E_\theta(\mathbf{x}_s^i)$ obtained with fixed and updated parameters respectively. Next, we formulate the three alternative distribution distance measures for $D(\Pr_{(\hat{\mathbf{f}}_t)\sim p}, \Pr_{(E_\theta(\mathbf{x}_t))\sim q})$.

W-GAN proposal: $m-$th order Wasserstein distance between two distributions can be defined as follows [4]:

$$W_m(\Pr_{(\hat{\mathbf{f}})\sim p}, \Pr_{((\mathbf{f}))\sim q}) = \inf_{\Pr_{\hat{\mathbf{f}}_s\sim p,(\mathbf{f}_t)\sim q}} \mathbb{E}\left[||\hat{\mathbf{f}}_s - \mathbf{f}_t||^m\right] \tag{2}$$

minimization is over all joint distributions. This is an optimal transport problem over joint probabilities of source and target domain and it has a dual formulation (Kantorovich-Rubinstein) [32]. Since the second part of Eqn. 1 has no joint probability component, it doesn't affect the optimization of the W1 distance and it can be rewritten for $m = 1$ as follows:

$$\inf_{\theta\in\Theta}\inf_{\phi\in\Phi} \mathbb{E}\left[D_\phi(\hat{\mathbf{f}}_s)\right] - \mathbb{E}\left[D_\phi(E_\theta(\mathbf{x}_t))\right] + \lambda l_g(\hat{\mathbf{f}}_s, E_\theta(\mathbf{x}_s)) \tag{3}$$

where Φ is the set of all parameters for ϕ, the parameters of the discriminator D. It can be solved with an adversarial neural network as shown by Arjovsky *et al.* [4]. Due to problems in the convergence behavior of the optimization, we use the gradient penalty variant of W-GANs which was introduced by Gulrajani *et al.* [27]. (see Algorithm. 1).

GAN proposal: Next, we propose to use the Jensen-Shannon (JS) distance which is proposed in DNNs by Goodfellow *et al.* [5]. JS distance $D(\Pr_{(\hat{\mathbf{f}}_s)\sim p}, \Pr_{(E_\theta(\mathbf{x}_t))\sim q})$ can be defined as follows:

$$JS = KL(\Pr_{\hat{\mathbf{f}}_s\sim p}||\Pr_{\tilde{\mathbf{f}}\sim r}) + KL(\Pr_{(\mathbf{f}_t)\sim q}||\Pr_{\tilde{\mathbf{f}}\sim r}) \tag{4}$$

KL is the Kullback-Leibler divergence and $\Pr_{\tilde{\mathbf{f}}\sim r} = (\Pr_{\hat{\mathbf{f}}_s\sim p} + \Pr_{(\mathbf{f}_t)\sim q})/2$ is the mixture from two distributions (see Algorithm. 1).

MMD proposal: Maximum mean discrepancy (MMD) is the third distance that we use to measure the similarity between the source and target domain data. It maximizes the two-sample test power and minimizes the Type II error. Given the mean embedding of the distribution p in Reproducing Hilbert Space (RKHS) $\mathcal{H}_k$ with a characteristic positive-definite reproducing kernel k is a unique element $\mu_k(p)$, then the distance between two distributions $D(\Pr_{(\hat{\mathbf{f}}_s)\sim p}, \Pr_{(E_\theta(\mathbf{x}_t))\sim q})$ is the RHKS distance between the mean embeddings of the distributions and it can be computed as:

$$\begin{aligned} \mathrm{MMD}_k(\Pr_p, \Pr_q) = \mathbb{E}_{\hat{\mathbf{f}}_s,\hat{\mathbf{f}}_{s'}} k(\hat{\mathbf{f}}_s, \hat{\mathbf{f}}_{s'}) + \mathbb{E}_{\mathbf{f}_t,\mathbf{f}_{t'}} k(\mathbf{f}_t, \mathbf{f}_{t'}) \\ - 2\mathbb{E}_{\hat{\mathbf{f}}_s,\mathbf{f}_t} k(\hat{\mathbf{f}}_s, \mathbf{f}_t) \end{aligned} \tag{5}$$

where $\hat{\mathbf{f}}_s, \hat{\mathbf{f}}_{s'} \sim p$, $\mathbf{f}_t, \mathbf{f}_{t'} \sim q$ are the features from source and target domains respectively and $\hat{\mathbf{f}}_s \neq \hat{\mathbf{f}}_{s'}$; $\mathbf{f}_t \neq \mathbf{f}_{t'}$. The details can be found in Algorithm. 1.

Interested readers are referred to [31] for a detailed justification of the algorithm and the related proofs.

Algorithm 1: Self-Supervised Domain Adaptation

```
Require: $\mathcal{X}_s$, $\mathcal{X}_t$, $\mathrm{E}_{\hat{\theta}}$, $\mathrm{E}_\theta$, $\mathrm{D}_\phi$, $\lambda > 0$, $n$: Mini-batch size,
  $A \in \{\text{WGAN,GAN,MMD}\}$, $n_{\text{crit}}$, $\lambda_p > 0$: Penalty gradient
Initialize $\theta = \hat{\theta}$
if (A = WGAN) OR (A = GAN) then
    initialize the parameters of $\mathrm{D}_\phi$
else
    provide a characteristic positive-definite kernel $k$
while ($\theta$) not converged do
    Sample $\{(\mathbf{x}_s^i)\}_{i=1}^n \in \mathcal{X}_s$, $\{(\mathbf{x}_t^i)\}_{i=1}^n \in \mathcal{X}_t$
    Compute $\hat{\mathbf{f}}_s^i = \mathrm{E}_{\hat{\theta}}(\mathbf{x}_s^i)$, $\mathbf{f}_t^i = \mathrm{E}_\theta(\mathbf{x}_t^i)$, $\mathbf{f}_s^i = \mathrm{E}_\theta(\mathbf{x}_s^i)$;
    if $A$ = WGAN then
        for $k = 1, \cdots, n_{crit}$ do
            Sample a random number $\epsilon \sim U[0,1]$;
            $\widetilde{\mathbf{f}}^i = \epsilon\hat{\mathbf{f}}_s^i + (1-\epsilon)\mathbf{f}_t^i$;
            Update $\mathrm{D}_\phi$ by ascending
                $$\frac{1}{n}\sum_{i=1}^{n}\mathrm{D}(\hat{\mathbf{f}}_s^i) - \mathrm{D}(\mathbf{f}_t^i) - \lambda_p(||\nabla_{\widetilde{\mathbf{f}}}\mathrm{D}(\widetilde{\mathbf{f}}^i)||_2 - 1)^2$$
        Update $\mathrm{E}_\theta$ by descending
            $$\frac{1}{n}\sum_{i=1}^{n}\mathrm{D}(\mathbf{f}_t^i) + \lambda l_g(\mathbf{f}_s^i, \hat{\mathbf{f}}_s^i)$$
    else if $A$ = GAN then
        for $k = 1, \cdots, n_{crit}$ do
            Update $\mathrm{D}_\phi$ by ascending
                $$\frac{1}{n}\sum_{i=1}^{n}\log\mathrm{D}(\hat{\mathbf{f}}_s^i) + \log(1 - \mathrm{D}(\mathbf{f}_t^i))$$
        Update $\mathrm{E}_\theta$ by descending
            $$\frac{1}{n}\sum_{i=1}^{n}\lambda l_g(\mathrm{G}(\mathbf{f}_s^i, \hat{\mathbf{f}}_s^i) - \log(\mathrm{D}(\mathbf{f}_t^i))$$
    else if $A$ = MMD then
        Update $\mathrm{E}_\theta$ by descending
            $$\frac{1}{n}\sum_{i=1}^{n}\lambda l_g(\mathbf{f}_s^i, \hat{\mathbf{f}}_s^i) + \frac{1}{n(n-1)}\sum_{l\neq j}k(\mathbf{f}_t^l, \mathbf{f}_t^j))$$
            $$+\frac{1}{n(n-1)}\sum_{l\neq j}k(\hat{\mathbf{f}}_s^l, \hat{\mathbf{f}}_s^j)) - \frac{2}{n^2}\sum_{l,j}k(\mathbf{f}_t^l, \hat{\mathbf{f}}_s^{\,j}))$$
```

IV. Experiments

We test our method in four datasets. After pretraining, the network parameters are optimized by using only the old and new domain images during adaptation.

SYNTHIA is a synthetic dataset with several Sequences under different weather and environment conditions [33]. We use the subway Spring Seq-01 (Sp1) with 1188 images for training initial network for all experiments on SYNTHIA. The labels are used only at this stage. The left frontal RGB images are used for training.

Winter and Night weather conditions are used for evaluating our adaptation method. We use Segnet [2] with the convolution layers similar to VGG-16 [34]. We obtain the optimized parameters of VGG-16 from a pre-trained version on ILSVRC/Imagenet [35]. Stochastic Gradient Descent (SDG) is used with a learning rate of 1×10^{-4} and a momentum of 0.9. The batch size is 6. The training stops until the loss converges or the maximum number of iterations is 2000.

After the initial model training, we update the parameters as explained in Algorithm. 1. The initial layers upto Block2 (B2) in VGG16 network is selected as the E network (see Fig. 4a). The features are considered to have sufficient information to transfer the knowledge from the source domain to the target domain. It should be noted that this selection is made manually in this study by considering the available information of the features. Each Block contains convolutional units, max pooling layers, and ReLU with skip connections to deconvolutional Blocks as explained in [2].

A simple discriminator network is selected and is shown in Fig. 4b. B1 and B2 have 2D convolutional layers and instance normalization layer [36]. FC1 and FC2 are fully connected layers. $c_0 = 512$, $c_1 = c_2 = 64$ and $c_3 = 10560$ are number of channels and FC2 outputs a scalar value. For GAN, additional ReLU+softmax layer is used. The discriminator is used for W1 and GAN. The parameters are initialized with random variables from a normal distribution. For domain adaptation training, Adam optimization is used with hyperparameters $\alpha = 1 \times 10^{-4}$, $\beta_1 = 0.7$ and $\beta_2 = 0.9$. The hyperparameters for Algorithm. 1 are $\lambda_p = 0.1$, $\lambda = 10$, $n_{\text{crit}} = 2$ and a batch size of 2. For MMD, an inverse multiquadratics kernel $k(\mathbf{f}_t, \mathbf{f}_s) = C/(C + ||\mathbf{f}_t - \mathbf{f}_s||_2^2)$ is used with $C = 2d_f\sigma_f$. The dimension of the features is d_f. The hyperparameters are $\lambda = 100$, $\sigma_f = 1$ and the batch size is 2. The iterations stop when θ converge or the maximum iterations are 10000 for W-GANs, 20000 for GANs and 5000 for MMDs. The total estimation time is $\sim$ 40 ms where the inference on GPU is $\sim$ 1.4 ms. The computations are performed on Nvidia GTX 1080Ti. The adaptation training time is ~ 0.4h for W1, ~ 4.0h for GAN and ~ 0.8h for MMD.

Ablation Study-1: First, we perform the ablation study with three alternative distance measures: Wasserstein (W1), Jensen-Shannon (GAN) and Maximum Mean Discrepancy (MMD) on this synthetic dataset. We use Winter Seq-01 (Wi1) to adapt to Winter and test this adapted model on both Wi1 and Wi6 (Wi1 $\rightarrow$ Wi1 and Wi1 $\rightarrow$ Wi6). We believe that this is a realistic scenario since the vehicle will observe the new images from different weather conditions during the ride that it didn't use for adaptation training. Mean Intersection over Union (mIoU) is used to measure the accuracy of the method (Table. I). The ratio of change in the accuracy with respect to the initial model are given in parenthesis as percentage. *"Init"* refers to the initial non-adapted pretrained model. We repeat the same procedure for Night conditions. For Sp6, that is the similar weather condition to the source domain, mIoU decreases slightly. mIoU for target domain Wi1 $\rightarrow$ Wi1 increases significantly for all distance measures. For Wi1 $\rightarrow$ Wi6, which has the unobserved target domain images, mIoU increases except for GAN. Overall, for the cases where the the evaluation is made on the same images with which the adaptation training is performed, mIoU is higher. The increase in adaptation is significantly higher than the slight reduction in the source domain (Wi1 $\rightarrow$ Sp6). To find the possible capacity of SegNet for Winter, we train with Wi1 labels and test it with Wi6 images, since the network has access to the labels in this case, this is called as the oracle. The mIoU

TABLE I: Tests on Non-Adapted Images

%(%)	Wi1 $\rightarrow$ Sp6	Wi1 $\rightarrow$ Wi1	Wi1$\rightarrow$ Wi6	Ni1$\rightarrow$Sp6	Ni1$\rightarrow$Ni1	Ni1$\rightarrow$Ni6
Init	**59.8(0)**	55.9(0)	47.8(0)	**59.8(0)**	53.7(0)	35.6(0)
W1	58.9(-1.5)	59.7(6.9)	49.4(3.2)	57.7(-3.5)	55.2(2.9)	**36.9(3.8)**
GAN	58.8(-1.6)	57.5(2.8)	46.9(-1.8)	58.9(-1.5)	**56.6(5.4)**	35.2(-1.0)
MMD	59.6(-0.2)	**62.4(11.8)**	**50.2(5.0)**	59.0(-1.3)	**56.6(5.4)**	**36.9(3.8)**

TABLE II: Self Adaptation to (Wi)nter

%(%)	Wi1	Wi2	Wi4	Wi5	Wi6
Init	55.9(0)	31.2(0)	31.1(0)	**37.5(0)**	47.8(0)
Adapt	**59.7(6.9)**	31.3(0.3)	**32.6(4.9)**	35.6(-5.2)	**50.0(4.5)**
Mixed	59.2(6.1)	**31.6(1.4)**	31.3(0.8)	36.4(-3.0)	47.9(0.2)

is 53.9%, which is slightly higher than adaptation result of 50.2% for unobserved target domain data. The adapted model achieves a higher mIoU for target domain than the oracle when the environment is similar to the source domain and only weather conditions change. Therefore, the structure of the environment is important during adaptation. To make a comparison with other studies, we use Cycada [28], which adapts from fall to winter in Synthia. It has an mIoU of 63.3% which is lower than the mIoU of its oracle which is 70.5%. Our system achieves a higher accuracy with respect to our oracle. For Night, mIoU of Ni1$\rightarrow$Sp6 does not decrease significantly after adaptation (Table. I). For Ni1$\rightarrow$Ni1, all distance measures increase the mIoU; however, GAN reduces mIoU for Ni6. The improvement in Night is less in overall with respect to the Winter condition probably due to the insufficient information of the Night weather condition since the lack of light reduces details.

Discussion: From this ablation study, it can be concluded that for a network with a small number of parameters, MMD and W1 measures achieve a higher accuracy for adaptation. However, their generalization to unobserved target domains may be insufficient. Their superiority may be related to their better convergence properties with respect to GANs; while the problem of generalization is probably related to the capacity of the DNN. We use the W1 measure for the next ablation study since it is faster to train and has similar accuracy with MMD measure.

Ablation Study-2: Next, we investigate adaptation and test in the same target domain in detail for both Winter (Table. II) and Night (Table. III). The pretrained model in Spring-01 is used in all scenarios. In addition, we adapt by using a mixture of the Winter and Night conditions of the related target domain to investigate the very basic strategy of adaptation to both domains. The adaptation improves mIoU except for Wi5 and Ni5. The reason for decrease in 5^{th} scenario is probably related to the dissimilar structure. When the differences in the distribution of visual features are high, it becomes difficult for a limited network to adapt to different distributions. For the mixed case, winter improves while night reduces, from which we can conclude that the simple mixing strategy is not sufficient to adapt to both conditions at the same time with a simple network.

Qualitatively, source domain Sp6 is not affected after adaptation to winter as shown in Fig. 3. For Wi1, adaptation improves the segmentation by reducing the region on the right which is erroneously classified as sky. Here, Wi6 is given as

TABLE III: Self Adaptation to (Ni)ght

%(%)	Ni1	Ni2	Ni4	Ni5	Ni6
Init	53.7(0)	14.1(0)	27.3(0)	**22.0(0)**	35.6(0)
Adapt	**55.2(2.9)**	**16.2(14.6)**	**28.1(2.7)**	20.4(-7.4)	**37.7(5.8)**
Mixed	52.1(-2.9)	13.9(-1.2)	26.7(-2.2)	21.5(-2.3)	34.5(-3.1)

an example of a test on non-adapted image (Wi1 $\rightarrow$ Wi6). Even though the adaptation is not made on these images, the road which is confused with lane markings due to snow on it is partially corrected after adaptation. In these illustrations, MMD achieves a better adaptation.

Discussion: In overall, a network with rather small number of parameters has limitations even if the images are synthetic and easier to classify. For simple cases such as adaptation to a single weather condition in a similar structured environment, the network is capable of learning by using the distance measure losses which are easier to compute. However, the accuracy reduces for complex cases such as adaptation to scenes with dissimilar structures or mixed weather conditions. Therefore, we use networks with more parameters which should result in a higher capacity for learning different conditions in more complex conditions for real world experiments.

Foggy Cityscapes is a variant of the CS ([3], [37]). The realistic foggy images are obtained from the real images by overlaying a fog with three different magnitudes. Fog-02 is thick while Fog-005 is thin and Fog-01 is in between the two. We use the 2975 training images of CS with fine labeling to obtain the pretrained DNN model. For adaptation, we use the mixture of the three fog conditions by considering that in a natural scenario, the thickness of the fog may vary as the vehicle moves. The validation images of the Foggy CS are the target domain data. We use networks with larger number of parameters, namely MobileNetV2 (MNV2) [1] and DeepLabV3 (DLV3) [38] and use their PyTorch implementations based on repositories [39], [40] with modifications as necessary.

For pretraining on source domain, we use a learning rate of 7×10^{-3} and SGD for optimization. For decoding layers of MNV2, we use a small single Atrous Spatial Pyramid Pool (ASPP) layer with 320 channels as input to reduce the computation time which is named as mobile ASPP (mASPP) [1]. We use Xception65 [41] as the backbone of DLV3 and ASPP for its decoding. A batch size of two is used and the training continues either until the parameters of the network converge or the number of epochs reach to 60, where one epoch equals to one pass of training images for both network architectures. For domain adaptation training, we assign network E as the initial layers of MNV2 upto 6^{th} bottleneck layer with output of 160 channels as shown in Fig. 4c. C1 corresponds to convolutional block and B corresponds to Bottleneck layer. For DLV3, we assign the entry flow of Xception65 upto the entry of the last block which has 256 channels as the network E (Fig. 4d). X2 corresponds to the blocks consisting of convolutions, ReLU and Maxpooling layers of the Xception network. We consider the computation times and memory requirements and select only the initial few layers of these more complex networks.

Ablation Study: The discriminator for W1 and GAN is similar to the previous discriminator (Fig. 4b). The parameters

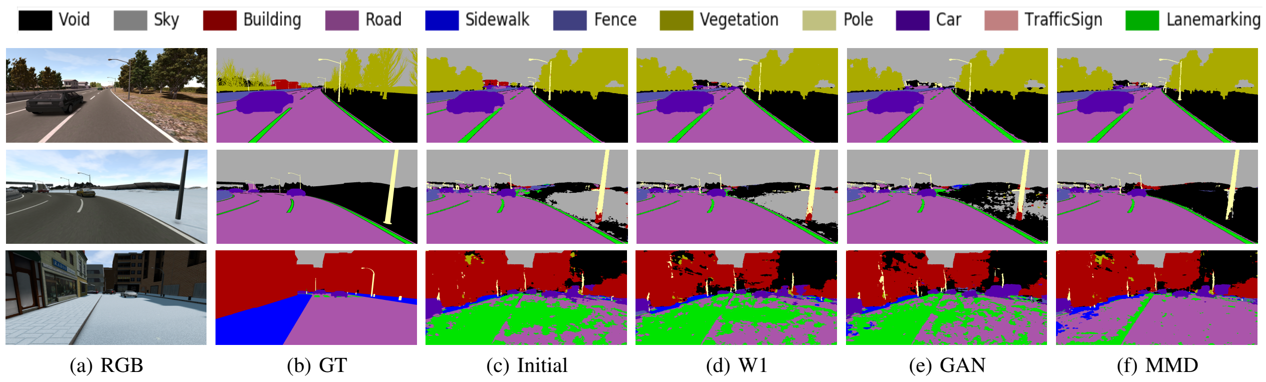


Fig. 3: Sample outputs from SYNTHIA. From top to bottom: Labels, Sp6, Wi1 and Wi6. From left to right: RGB, Ground Truth, images estimated with the initial model, after adaptation with the distance measures W1, GAN and MMD.

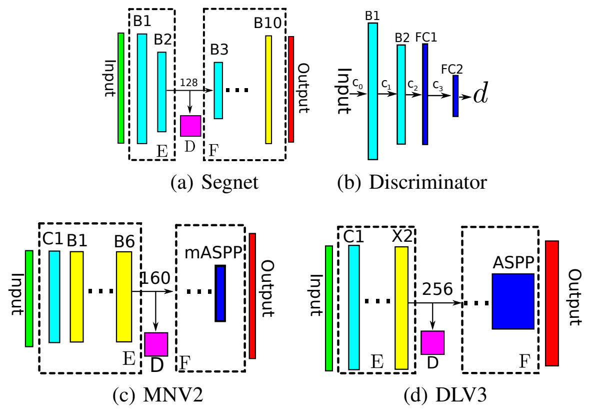


Fig. 4: Block diagrams of the networks.

TABLE IV: Foggy CS with MobilenetV2

%(%)	Initial	Finetune	W1	GAN	MMD
Fog02	46.9(0)	67.7(44.3)	55.2(17.7)	53.0(13.0)	**57.9(23.5)**
Fog01	59.4(0)	69.0(16.2)	64.5(8.6)	63.2(6.4)	**66.3(11.7)**
Fog005	66.6(0)	69.4(4.1)	68.5(2.9)	67.9(1.9)	**69.5(4.3)**
Normal	72.0(0)	68.4(-5.1)	69.7(-3.2)	69.3(-3.7)	**70.0(-2.8)**

are $c_0 = 160$, $c_1 = c_2 = 8$ channels and $c_3 = 262144$ for MNV2 and $c_0 = 256$, $c_1 = c_2 = 8$ channels and $c_3 = 262144$ for DLV3. Adam optimization is used with same hyperparameters. Other parameters are as follows; for MNV2 with W1, $\lambda_p = 10$, $\lambda = 10$, $n_{\text{crit}} = 2$; for MNV2 with GAN, $\lambda_p = 0.1$, $\lambda = 20$, $n_{\text{crit}} = 2$; for MNV2 with MMD, $\lambda_p = 10$, $\lambda = 10$, $n_{\text{crit}} = 1$, $\sigma_f = 1$; for DeepLabV3 with W1, $\lambda_p = 1$, $\lambda = 20$, $n_{\text{crit}} = 2$; for DLV3 with MMD, $\lambda_p = 10$, $\lambda = 1$, $n_{\text{crit}} = 1$. For MMD, again the same inverse multiquadratics kernel is used. The iterations stop when θ converge or the maximum iterations for W1 or GAN are 10000 or 2000 for MMD. For MNV2, the inference time is $\sim$ 178 ms including all operations to prepare the image for segmentation. The adaptation training time is $\sim$ 1.2h for W1, $\sim$ 6.0h for GAN and $\sim$ 0.8h for MMD. The inference time is $\sim$ 1456 ms for DLV3 and training times are $\sim$ 1.3h for W1, $\sim$ 6.5h for GAN and $\sim$ 0.9h for MMD accordingly. For the evaluation, we use test time augmentation (tta) with horizontal flip.

The results for MNV2 are given for each foggy condition separately in Table. IV. *Finetune* uses the labels of the foggy conditions for training (oracle). As expected these have the highest accuracies since they use labels for learning. The accuracy of the normal condition decreases after finetuning since it isn't used for finetuning. MMD has the highest accuracy among all the distance measure types.

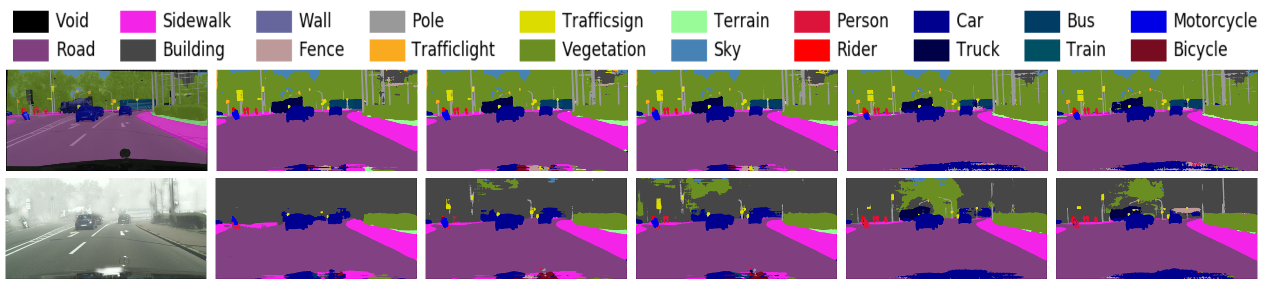
TABLE V: Foggy CS with DeepLabV3

%(%)	Initial	W1	W1, $\lambda = 0$	MMD
Fog02	60.8(0)	65.1(7.0)	**66.0(8.5)**	62.2(2.2)
Fog01	65.7(0)	**70.6(7.5)**	70.0(6.5)	70.1(6.7)
Fog005	68.5(0)	**73.5(7.3)**	73.1(6.7)	73.3(7.0)
Normal	74.2(0)	**74.2(0)**	68.4(-7.9)	73.5(-1.0)

For DLV3, we only provide results for W1 and MMD since GAN has similar or less accurate results as shown in the previous evaluations. In addition, we show the effect of self-supervision by providing results for no self-supervision (W1, $\lambda = 0$, Table. V). The mIoU of source domain drops significantly without self-supervision. The initial DLV3 model with Foggy conditions have a higher accuracy than the MNV2 due to higher number of parameters. However, our approach still increases the mIoU results for both distance measures. In DLV3, we measure the similarity between features at an earlier layer than in the MNV2 and W1 gives better results.

The visualizations of the adaptation results are shown in Fig. 5. The RGB and Ground Truth masks are overlaid on the upper left. The foggy variant images are obtained by using the scenes of the RGB images, therefore they have the same GT. The foggy RGB can be seen in the first column. For MNV2, the initial network cannot segment the vegetation or the lights which are at a distance from our vehicle. Although the fog is heavy in Fog02, MNV2 is able to segment these regions after adaptation. Furthermore, the bus, which is far away, is also better segmented, especially with MMD. For DLV3, after adaptation, the visibility of the poles and the traffic lights increase slightly for Foggy condition.

Discussion: For the same dataset, we observe that two different distance measures achieve a better accuracy in two different DNNs. MMD has better results with MNV2 probably due to the fact that the features from MNV2 in fog condition can be easily discriminated by MMD while discriminator based methods try to learn the difference. However, for DLV3, the features are processed more and have less information



(a) RGB & GT (b) MNV2 Initial (c) MNV2 W1 (d) MNV2 MMD (e) DLV3 Initial (f) DLV3 W1

Fig. 5: CS samples for a scene with different estimators. GT: Ground Truth, MNV2: MobileNetV2, DLV3: Deeplabv3, From Top to bottom: Labels, Original image (source domain), Fog02.

TABLE VI: BDD at Different Weather Conditions

		DLV3		MNV2	
Dist	Domain	Rainy	Snowy	Rainy	Snowy
Init		36.96	37.25	32.68	33.46
W1	Self	37.25(0.78)	37.52(0.73)	33.26(1.78)	**33.99(1.58)**
	Mixed	36.4(-1.52)	37.4(0.42)	33.17(1.51)	33.76(0.91)
MMD	Self	**37.35(1.07)**	37.53(0.77)	33.11(1.3)	33.52(0.18)
	Mixed	37.18(0.59)	**37.58(0.9)**	**33.46(2.37)**	33.84(1.14)

related to their distribution. W1 based discriminator (critic) learns how to find the distance in distribution better than the measurement of MMD. It should be noted that the results are close and only fog is added to these images; therefore, the structures are similar in both domains. Both of the models don't forget the source domain segmentation and they can be used in foggy and normal conditions with similar accuracy.

Berkeley Driving Dataset (BDD) [42] consists of images taken from cameras by the users inside the vehicles. They are labeled for semantic segmentation and have attributes for the weather conditions. We use the images with attributes of clear, rainy and snowy weather conditions at daytime. We add an additional mixed category where again we simply mix the rainy and snowy images during adaptation training (Table. VI). The same parameters with the Foggy CS experiments are used. We use the clear daytime images and labels for pretraining only and adapt this to other conditions. The results are improved slightly in all of the cases except for rainy condition tested with DLV3 mixed adaptation. The increase for DLV3 is slightly less than MNV2, probably due to already higher performance of DLV3. Although the increase is slight, the effects can be important for autonomous vehicles such as the cars in the front may not be detected without adaptation (e.g. Fig. 6).

Discussion: The improvement in mIoU is less w.r.t. Foggy CS. One of the reasons is that the images in BDD are not from a single view point of the vehicle but from different viewpoints. This adds an additional dimension to the problem of feature distribution computation. Another reason is that the weather attributes have errors in the original dataset, such as some night time images have attributes of daytime, or some snowy images have rainy attribute. However, the method is still capable of improving the accuracy since the majority of the images share a common distribution.

Semantic KITTI [43] is a dataset collected from cameras mounted on a vehicle. Although they don't contain images

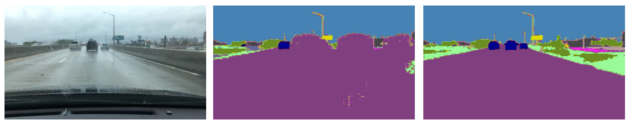


Fig. 6: BDD sample. Left: RGB in rainy; Center: No adaptation; Right: DLV3 MMD rainy adaptation.

TABLE VII: UDA applied to KITTI

CS only	**W1**	**MMD**	**Bolte *et al.* [25]**	**Finetune**
44.1	**60.4**	59.0	59.5	63.2

at varying weather conditions, we perform these evaluations to find the performance of our method in adaptation to a new dataset with real images in an autonomous vehicle setting. We use the DNN pretrained in CS and adapt to KITTI without using any labels from KITTI (Table. VII). Semantic KITTI contains 200 train and 200 test images. For adaptation, we use only test images; for finetune (oracle), we use the labels of train to finetune parameters of pretrained DNN and evaluate on test images. DLV3 with W1 has an mIoU of 60.4%, which is higher than other UDA methods on semantic KITTI ([25]). Currently, our method achieves the best UDA performance on the benchmark.

Discussion: The images are from the same vehicle; therefore the difference in the distribution of the features due to view points doesn't exist. Furthermore, the images are generally from clear daytime. However, the initial CS pretrained DNN performs only 44.1 % (Fig. 7). The adaptation is successful and achieves a mIoU close to the *oracle* since the discriminator learns the difference between the two distributions which have different environment properties and backpropogates this successfully through the inital layers of DNN.

V. Conclusion

We have proposed a method which is capable of adapting a DNN to new conditions without forgetting the source domain segmentation. We achieve this without changing the architecture of the original DNN. Furthermore, we don't

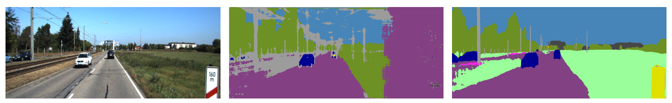


Fig. 7: KITTI sample. Left: RGB; Center: DLV3 trained in CS only. Right: Adaptation to KITTI with W1.

require the labels of the old weather condition or the new weather condition for adaptation training. We conclude that the selection of the distance measure depends on the data and network type. For the target domain with a similar structure of the scenes with similar viewpoints in a different weather condition on a DNN with less number of parameters, MMD achieves higher results; while when the features get diverse either due to both different structure and weather conditions with a complicated DNN, W1 can learn to differentiate these features better. The results show that the approach is capable of improving the accuracy after adaptation to new conditions; whereas, the amount of improvement depends on the network structure and the dissimilarity of new data from source domain data.

Acknowledgement

Parts of the experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations. We thank Gabriel Othmezouri for his support and helpful discussion. This work was supported by Toyota Motor Europe.

References

[1] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *CVPR*, 2018, pp. 4510–4520.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE PAMI*, vol. 39, no. 12, pp. 2481–2495, 2017.

[3] C. Sakaridis, D. Dai, and L. V. Gool, "Semantic Foggy Scene Understanding with Synthetic Data," *IJCV*, pp. 1–20, 2018.

[4] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[6] B. Gong, K. Grauman, and F. Sha, "Connecting the Dots with Landmarks: Discriminatively Learning Domain-Invariant Features for Unsupervised Domain Adaptation," in *ICML*, vol. 28, 2013, pp. 1–9.

[7] D. Dai and L. Van Gool, "Dark Model Adaptation: Semantic Image Segmentation from Daytime to Nighttime," in *ITSC*, 2018.

[8] M. Wulfmeier, A. Bewley, and I. Posner, "Incremental Adversarial Domain Adaptation for Continually Changing Environments," in *ICRA*, 2018.

[9] A. Valada, R. Mohan, and W. Burgard, "Self-Supervised Model Adaptation for Multimodal Semantic Segmentation," in *CoRR*, 2018.

[10] D.-k. Kim, D. Maturana, M. Uenoyama, and S. Scherer, "Season-Invariant Semantic Segmentation with A Deep Multimodal Network," in *Field and Service Robotics*, 2018, pp. 255–270.

[11] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud," in *ICRA*, 2019.

[12] Y. Chen, W. Li, X. Chen, and L. Van Gool, "Learning Semantic Segmentation from Synthetic Data: A Geometrically Guided Input-Output Adaptation Approach," in *CVPR*, 2019.

[13] O. Erkent, C. Wolf, C. Laugier, D. Gonzalez, and V. Cano, "Semantic grid estimation with a hybrid bayesian and deep neural network approach," in *IEEE IROS*, 2018, pp. 888–895.

[14] H. Porav, T. Bruls, and P. Newman, "I Can See Clearly Now : Image Restoration via De-Raining," in *ICRA*, 2019.

[15] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation," in *CVPR*, 2019.

[16] Y. Luo, L. Zheng, T. Guan, J. Yu, and Y. Yang, "Taking A Closer Look at Domain Shift: Category-level Adversaries for Semantics Consistent Domain Adaptation," in *CVPR*, 2019, pp. 2507–2516.

[17] X. Zhu and D. Lin, "Adapting Object Detectors via Selective Cross-Domain Alignment," in *CVPR*, 2019, pp. 687–696.

[18] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *CoRR*, vol. abs/1412.3474, 2014.

[19] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schoelkopf, and A. Smola, "A Kernel Method for the Two-Sample Problem," *JMLR*, vol. 13, pp. 723–773, 2012.

[20] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning Transferable Features with Deep Adaptation Networks," in *ICML*, vol. 37, 2015.

[21] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *JMLR*, vol. 17, no. 1, pp. 2096–2030, 2016.

[22] 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 *CVPR*, 2018, pp. 7472–7481.

[23] Y. Zhang, Z. Qiu, T. Yao, D. Liu, and T. Mei, "Fully Convolutional Adaptation Networks for Semantic Segmentation," in *CVPR*, 2018, pp. 6810–6818.

[24] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, and R. Chellappa, "Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation," in *CVPR*, 2018, pp. 3752–3761.

[25] J.-A. Bolte, M. Kamp, A. Breuer, S. Homoceanu, P. Schlicht, F. Huger, D. Lipinski, and T. Fingscheidt, "Unsupervised domain adaptation to improve image segmentation quality both in the source and target domain," in *CVPRw*, 2019.

[26] X. Peng, Y. Li, Y. L. Murphey, X. Wei, and J. Luo, "Domain Adaptation with One-step Transformation," in *IEEE, SSCI*. IEEE, 2018, pp. 539–546.

[27] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *NIPS*, 2017, pp. 5767–5777.

[28] J. Hoffman, E. Tzeng, T. Park, J.-y. Zhu, U. C. Berkeley, P. Isola, K. Saenko, A. A. Efros, T. Darrell, and U. C. Berkeley, "CYCADA: Cycle-Consistent Adversarial Domain Adaptation," in *ICML*, 2018, pp. 1–15.

[29] Y. Zhang, P. David, and B. Gong, "Curriculum Domain Adaptation for Semantic Segmentation of Urban Scenes," in *ICCV*, 2018.

[30] Y. Li, L. Yuan, and N. Vasconcelos, "Bidirectional Learning for Domain Adaptation of Semantic Segmentation," in *CVPR*, 2019, pp. 6936–6945.

[31] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf, "Wassertein Auto-Encoders," in *ICLR*, 2018, pp. 1–17.

[32] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008, vol. 338.

[33] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez, "The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *CVPR*, 2016.

[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[36] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *CoRR*, vol. abs/1607.08022, 2016.

[37] 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 *CVPR*, 2016.

[38] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," 2017. [Online]. Available: http://arxiv.org/abs/1706.05587

[39] M. P. Shah, "Semantic segmentation architectures implemented in pytorch." *https://github.com/meetshah1995/pytorch-semseg*, 2017.

[40] H. Taniai, "Pytorch implementation for semantic segmentation (deeplabv3+, unet, etc.)," *https://github.com/nyoki-mtl/pytorch-segmentation*, 2019.

[41] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017, pp. 1251–1258.

[42] 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," *CoRR*, vol. abs/1805.04687, 2018.

[43] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *IJCV*, 2018.