Real-Time Domain Adaptation in Semantic Segmentation: an assessment of the relevance of a lightweight domain discriminator

Lorenzo Ottino
s287791@studenti.polito.it

Mustafa O.M.E. Elshaigi
s289815@studenti.polito.it

Alireza Talakoobi
s289641@studenti.polito.it

Abstract

In deep CNN based models for semantic segmentation, high accuracy relies on rich spatial context (large receptive fields) and fine spatial details (high resolution), both of which incur high computational costs. The other limitation is the fact that these models are based on supervision with pixel-level ground truth but may not generalize well to unseen image domains. Our goal is to create a model that generalize well and can adapt source ground truth labels to target domain labels while maintaining both lower computations and inference time. In our paper we combined a real-time semantic segmentation model with an unsupervised adversarial domain adaptation approach, and we used a domain classifier to align the features extracted from the output space. As the labeling process is tedious and labor intensive, we adopted a synthetic-to-real scenario. To meet our objective, we researched the effect of using a light weight discriminator network to enhance the time and avoid over-fitting by reducing the capacity of the model. We compared three different discriminator architectures and we accomplished significant enhancement both in terms of speed and segmentation performance.

1. Introduction

Real-time semantic segmentation and Domain adaptation are two important tasks in computer vision research.
Namely, in Semantic Segmentation given an input image, we want to assign a class label to each pixel in an output image of the same size. Moreover, this task needs to be accomplished in real-time or at least rapidly in many applications, e.g. autonomous vehicles or video surveillance must get rapid results to be useful in a real-world scenario.
One of the main issues in this task is the availability of big annotated datasets. Since manually annotating each image in the dataset is very time-consuming and impractical, a common solution is to train the algorithm with synthetic data and try later to adapt it to a target distribution of interest, which is the goal of Domain Adaptation. In this paper, we use the BiSeNet architecture, introduced by [13] to perform real-time semantic segmentation, and then perform unsupervised adversarial domain adaptation.
Contributions: we compare three different discriminator functions for the adversarial domain adaptation. Our main goal is to try different strategies to improve the training time of a discriminator without compromising its accuracy. In this context, we find that a discriminator function based on depthwise separable convolution can significantly reduce training time and improve the domain adaptation accuracy if compared to discriminators that use fully convolutional layers.

2. Related Works

Semantic Segmentation: Working on semantic segmentation in computer vision is a challenging task. Semantic segmentation has substantially improved in recent years due to new deep learning models such as ResNet [7]. In ”A brief survey on semantic segmentation with deep learning” [6], a great number of novel methods are reviewed, e.g. PSPNet[14] that introduced Pyramid Pooling and DeepLabV3 [3] that uses Atrous Spatial Pyramid Pooling. But most of these methods are also computationally expensive and memory consuming. Semantic segmentation necessitates two key pieces of data: rich spatial information and a sizeable receptive field. There are different ways to accelerate the model and make it closer to real-time such as: restricting the input size to reduce the computation size, pruning the channels of the network to boost the inference speed and dropping the last stage of model to reach an extremely tight frame work. Modern techniques sometimes sacrifice spatial resolution in order to attain real-time inference speed, resulting in poor model performance. We can witness a novel Bilateral Segmentation Network in BiseNet [13]. It presents a model with a spatial path to preserve Spatial information and a Context path to collect sufficient receptive information. On the Cityscapes, CamVid [2], and COCO-Stuff datasets, this model strikes the ideal balance between speed and segmentation performance.
Domain Adaptation: As previously stated, semantic segmentation is a difficult task, and even improved models

require a large amount of training data and still perform poorly in some areas. For example, it is challenging to provide enough data in most real-world applications, and the trained model may not generalize well to unknown image domains. Constructing synthetic datasets based on rendering, such as GTA5, SYNTHIA, and IDDA [1], is one way to deal with the need for huge data. This strategy is less expensive, but we cannot achieve satisfactory results merely by training on synthetic datasets because the model does not generalize well to real-world data in this manner. The huge domain shift between synthetic and real-world images is mostly responsible for this. Many domain adaptation techniques have been developed to close this performance gap . Many methods are introduced in "A review of single-source deep unsupervised visual domain adaptation" [15] but we can see that domain adaptation for pixel-level prediction tasks have not been explored widely. There are some methods that represent impressive results such as DANN [5] that applies adversarial learning in a fully-convolutional way on feature representations. AdaptSegNet [12] proposes an efficient domain adaptation algorithm through adversarial learning in the output space and mentions pixel-level predictions are structured outputs that contain information spatially and locally. The discriminator models that are mainly used in domain adaptation techniques are usually fully convolutional models but we found a new depth wise structure in Xception [4].

3. Method

3.1. Overview of the Proposed Model

Our algorithm is inspired by [12] and consists of two models: a segmentation network which represents the generator G and a Discriminator network D which acts as a domain classifier to encourage the generator to reduce the gap between the source and the target distribution in the output space. We are referring to the features extracted from the output space as segmentations. Two sets of images $\in \mathbb{R}^{H\times W\times 3}$ were used as source and target domains, and we denote them as $\{I_S\}$ and $\{I_T\}$. We split the training into two phases. We first train the generator by passing the images from the source domain I_s (with annotations) to the segmentation network for optimizing the generator G which produces predictions P_s. Then Target images I_t (without annotations) are passed to the generator to predict segmentations in form of a tensor of Softmax outputs P_t. We set the ground truth for the discriminator as $\{0 : source\_label\}$ and $\{1 : target\_label\}$. Since we desire to bring the source and target predictions $\{P_s, P_t\}$ as close as possible, we adopted an unsupervised learning approach, in which we pass the target predictions P_t along with the $source\_label$ to the discriminator. The generator will try to fool the discriminator and maximize its errors. The second phase is training the discriminator D with both source and target segmentations extracted from the output of the generator with the correct ground truth labels. The network propagates gradients from the discriminator, which will be optimized to encourage the generator G to produce similar segmentation distributions in the target domain to the source domain. Figure 1 shows an overview of the network architecture. We used an adversarial loss on the target predictions. Only gradients from the generator are propagated in this phase.

3.2. Domain Adaptation With Adversarial Training

Features extracted from the output space are more suitable for the task of domain adaptation in semantic segmentation as proposed by Adapt-SegNet[12]. They based their assumption on the fact that low-dimension segmentations contain rich information like scene layout and context. These segmentations should share strong similarities both locally and spatially regardless of the origin of the image $(i.e. I_S, I_T)$. So in our experiments we performed the domain adaptation using an adversarial learning scheme where the low-dimension softmax outputs extracted from the output of the generator G are passed to three different discriminator Networks. Our objective is to examine the effect of changing the discriminator architecture in terms of segmentation performance and speed.

3.3. Objective Function for Domain Adaptation

With the proposed network and inspired by Adapt-SegNet[12], we formulate a domain adaptation task containing two loss functions from both models:

$$L(I_s, I_t) = L_{seg}(I_s) + \lambda_{adv} L_{adv}(I_t) \tag{1}$$

where L_{seg} is the dice loss using ground truth annotations in the source domain (See section 3.3.1). L_{adv} is the adversarial loss that adapts the predicted segmentations of target images with the distribution of source predictions. We used a Binary Cross Entropy loss with logits as in [12] (see Section 3.3.2). λ_{adv} in 1 is the weight used to balance the two losses as suggested by [12].

3.3.1 Segmentation Network objective Function

We defined the segmentation loss in 1 as the Dice loss for images from the source domain, we choose this loss to override the limitation of Cross Entropy loss which averages the per-pixel loss for pixels individually, so large objects contribute more to it than small ones, which is why it requires additional weighting to avoid ignoring minority classes. Dice loss instead considers the loss information both locally and globally, which is critical for high accuracy. Given an input image I_s, ground-truth G_s and the prediction $P_s = G(I_s)$, we used the dice coefficient to measure the overlap between the I_s and G_s. Our objective is to maximize this overlap (i.e. to maximize the Dice Coefficient). Hence, we minimize the loss {1-(dice coefficient)}

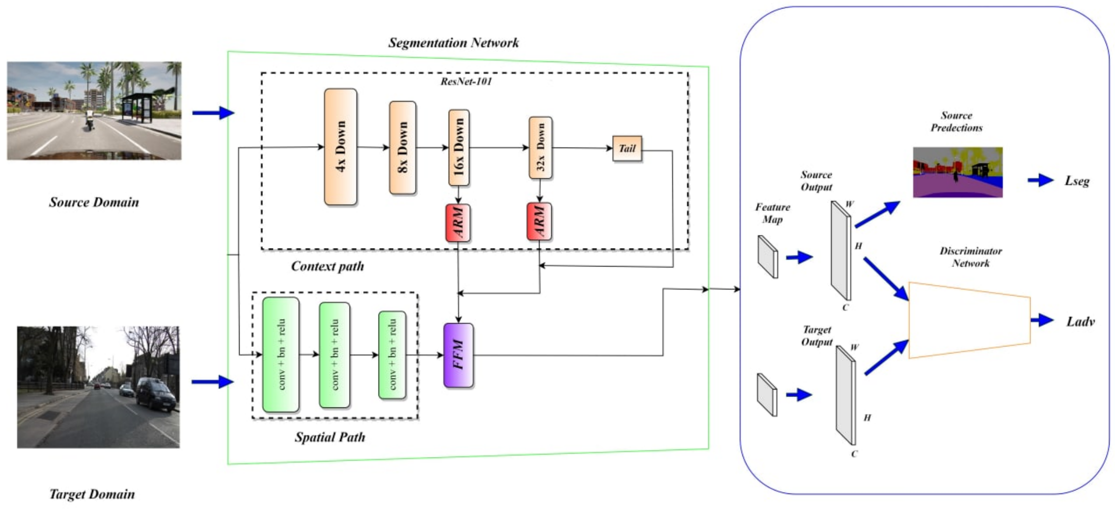


Figure 1. Overview of the network architecture.

as shown 2:

$$L(P_s, G_s) = 1 - \left(\frac{2\sum_i^N p_i g_i}{\sum_i^N (p_i + g_i)} \right) \quad (2)$$

where p_i, g_i denote to the value of i_{th} pixel on the prediction map P and the ground-truth G respectively. As suggested by BiSeNet [13] we added two specific auxiliary loss functions to supervise the output of the Context Path and α in 3 is set to 1 and K is set to 3 :

$$L_{seg} = L(P_s, G_s) + \alpha \sum_{i=2}^{k} L(P_{s_i}, G_s) \quad (3)$$

For the mini batch we take the mean of the losses over all images in the $mini\_batch$.
To make the distribution of P_t closer to P_s, we used $L_{adv}({P_t}^{(H \times W \times 1)}, Source\_label^{(H \times W \times 1)})$ as in 1. This loss is designed to train the segmentation network and fool the discriminator by maximizing the probability of the target prediction being considered as the source prediction.

3.3.2 Discriminator Network objective Function

We define the discriminator loss L_D as a BCE_with_logits as in [12], in the second phase of training we use the same L_{adv} in 1 but with different objective, we desire to minimize the discriminator errors to encourage the generator G to produce similar segmentation distributions in the target domain to the source domain in next iteration. We use the loss in:

$$L_D = \frac{1}{2} L_{adv}(P_s, S\_label) + \frac{1}{2} L_{adv}(P_t, T\_label) \quad (4)$$

Where S_label and T_label refer to the source and target labels respectively as defined in (section 3.1).

4. Network Architecture

In this section we explain the purpose of each block in our model for both segmentation and discriminator networks.

4.1. Segmentation Network

To perform real-time semantic segmentation, we used BiseNet model[13]. The model is based on splitting the function of spatial information preservation and receptive field sufficiency into two paths: a Spatial Path (SP) and a Context Path (CP); these two paths compute concurrently, considerably increasing the efficiency. The architecture of the bilateral network consists of the following blocks:

4.1.1 Spatial path

The spatial information of the image is crucial to predict the detailed output, to preserve the resolution of the input image without losing speed, BiSeNet proposed a spatial path that consists of three layers. Each layer includes a convolution with stride = 2, followed by batch normalization and ReLU Therefore, this path extracts an output feature map that is 1/8 of the original image. It encodes rich spatial information due to the large spatial size of feature maps while maintaining a low number of trainable parameters. The spatial path has only three convolution layers. Therefore, it is not computation intensive.

4.1.2 Context path

While the Spatial Path encodes affluent spatial information, the Context Path is designed to provide sufficient receptive field. To avoid loosing the speed by applying larger kernels or a pyramid pooling module which are computation demanding and memory consuming, the proposed Context Path utilizes lightweight model and global average pooling.

In our experiments we applied different versions of ResNet $[ResNet-18, ResNet-101]$ [7] as backbone, we found ResNet-101 pretrained on ImageNet gives the best results. The backbone model is used to rapidly down-sample the feature map and obtain large receptive field, which encodes high level semantic context information. Then we add a global average pooling on the tail of the backbone model, which can provide the maximum receptive field with global context information. Finally, the up-sampled output feature of global pooling and the features of the backbone model are then passed to the **Attention Refinement Module** ARM, designed to refine the output feature maps before combining them, ARM employs global average pooling to capture global context and computes an attention vector to guide the feature learning. It integrates the global context information easily without any up-sampling operation. Therefore, it demands negligible computation cost.

4.1.3 Feature Fusion Module (FFM)

The output of Spatial Path is low level feature representation, while the output of Context Path is high level. Therefore, BiSeNet proposed a specific Feature Fusion Module (FFM) to fuse these features. We first concatenate the output features of Spatial Path and Context Path, and then we apply batch normalization[8] to balance the scales of the features. Next, concatenated feature is pooled to a feature vector and a weight vector is computed, This weight vector can re-weight the features by performing feature selection and combination.

4.2. Discriminators architectures

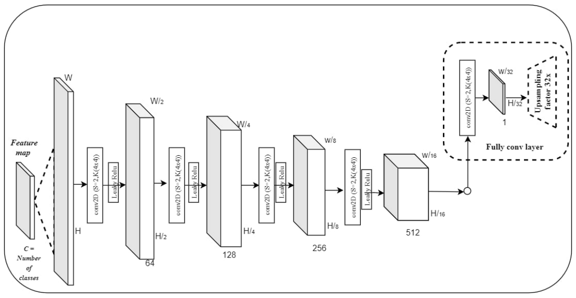
The main focus of this work regards the discriminator function used to perform domain adaptation. An overview of their architectures in shown in Figure 2.

4.2.1 Fully Convolutional

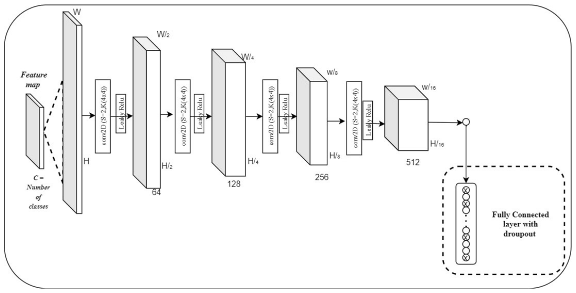
The first discriminator (2a) is inspired by AdaptSegNet [12], and it is composed of 5 fully convolutional layers. The first four layers are convolutional layers with $\{Stride : 2, Padding : 1, Kernel\_Size : 4\}$ followed by a LeakyRelu activation function, and the last one is followed by an up-sampling layer to reshape the input image to its original dimension.

4.2.2 Fully Connected Layer + Droupout

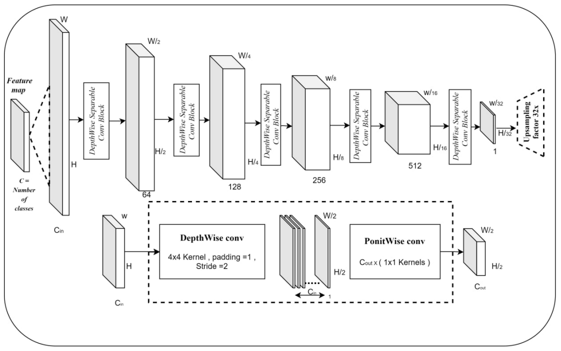
We tried to enhance the discriminator accuracy by replacing the last convolutional layer with a Fully Connected layer (2b). Fully Connected layer maps the high-dimensional features extracted by the convolutional layer to a lower-dimensional space before classification, this should prevent the discriminator loss from becoming small too quickly, especially at the initial stages of training. Since fully connected layers require more computations, to avoid loosing the speed we apply dropout [11] on it. Our intuition is applying dropout may lead to better generalization and reduce over-fitting.



(a) Fully convolutional discriminator



(b) Discriminator with dropout and fully connected layer



(c) Depthwise separable convolution layer

Figure 2. Considered discriminator architectures

4.2.3 Depthwise Separable Convolutional

For our last discriminator (2c), since our desire is to build a real time semantic segmentation model, our concern is to have a lighter discriminator while maintaining the accuracy, we took inspiration by Depthwise separable convolution[4]. To build our discriminator architecture we used a linear stack of 5 depthwise separable convolution blocks, where each block is followed by a Leaky_ReLU non-linearity parameterized by 0.2 (except for the last block), as implemented in [10]. Each block consists of:
DepthWise convolutions layer: is used control the spatial dimension of the input image. We used

$\{Stride : 2, Padding : 1, Kernel\_Size : 4\}$ to reduce the spatial dimension by the following fractions [1/2, 1/4 ,1/8, 1/16 ,1/32].

PointWise convolutions layer: is used to control the channels (contextual dimension) as shown by the following [N_c, 64, 128, 256 ,512, 1], where N_c is the number of channels for the input of the discriminator network. An up-sampling layer is added after the last block for re-scaling the output to the size of the input.

A detailed comparison in terms of number of parameters is in ($Table 1$)

5. Experiments

In this section we provide more details about the implementation, the datasets used, and the experiments conducted.

5.1. Datasets

Two datasets have been used in this project:

IDDA: introduced in 2020 by *Alberti et al.* [1], it consists in a large-scale synthetic dataset. It was specifically created to support works in the field of domain adaptation in semantic segmentation and contains 1006800 annotated pictures taken from a digital simulator with 1920×1080 resolution. IDDA is also a multi-domain dataset since its images come from 7 different digital towns, with 3 different simulated weather conditions and 5 viewpoints, meaning that all images are taken as if there were a camera placed at the height of the rear-view mirror of five different cars.

CamVid: introduced in 2009 by *Brostow et al.* [2], it is a commonly used real-world dataset. It contains pixel-level annotated 920×720 images taken by a high-resolution camera during a two hours long drive. Only 11 classes were used in this project, plus a ”void” class that represents everything else.

5.2. Implementation protocol

Seven experiments were conducted in this work. The results are presented in Table 2.

5.2.1 Semantic segmentation

First, we trained BiSeNet on the CamVid dataset using two different backbones, ResNet18 and ResNet101, and training for 50 and 100 epochs to obtain the first assessment on training time and accuracy of this model (1).

Second, we applied data augmentation, specifically horizontal flipping and Gaussian blur, intending to increase the model stability to lower resolution images and different sides of roadway. We also stretched the images with random scales and then randomly cropped them into a fix size. The best configuration found at the first step was retrained with the aforementioned data augmentation (2). The results found in these experiments will be the target benchmark for the domain adaptation part.

5.2.2 Unsupervised domain adaptation

After the pure real-time semantic segmentation experiments, we implemented an unsupervised domain adaptation pipeline.

This domain adaptation pipeline works as follows: at each epoch, some images with their labels from the synthetic domain are used to train the semantic segmentation modules; then the discriminator is trained with the output features coming from images from both datasets without annotations to minimize an adversarial loss (BCE Loss in our case) and encourage confusing the two domains.

More specifically, the best configuration found in terms of number of epochs, backbone, and augmentation was trained on a random subset of images from IDDA with the same size as the CamVid dataset at each epoch for 100 epochs.

In this framework, we tried the three discriminators presented in section 4.2 (experiments 3, 4, 5), and the best one in terms of mIoU and training time was considered for deeper investigations. In particular, the best discriminator was trained once again with the length of IDDA (6). In this way, since the number of images present in the CamVid dataset is approximately 1/6 of the IDDA dataset, at each epoch the whole CamVid dataset was used 6 times. Finally, we tried the best discriminator with the CamVid size, but removing the Gaussian blur to images coming from IDDA(7).

5.2.3 Implementation details

All the experiments were run on Google Colab Tesla T4 15GB GPUs. The implementation of the models was done with Pytorch. During training, we used as optimizers stochastic gradient descent for the segmentation module (SS) and adam for the domain adaptation part (DA), with batch size equal to 4. Similarly to BiSeNet [13], we applied the ”poly” learning rate strategy, with initial learning rates 2.5e-2 for SS and 1e-4 for DA. The value for λ_{adv} in (1) was set to 0.001 as in AdaptSegNet [12]. We used the auto mix precision proposed by Mixed Precision Training [9] to reduce the training time by using less memory while maintaining the accuracy of the model. The random values for the scales in the data augmentation are: {0.5, 1, 1.25, 1.5, 1.75, 2}. Code is publicly available on GitHub.

6. Results

In this section, we look at the findings of our experiments and compare them to one another.

Target: We tested the four model configurations and determined that training BiseNet with ResNet101 and 100 epochs delivers the best results, so we utilized that as the Target for our experiments. Also, we can see that, with the exception of ”SignSymbol,” the Target offers us good accu-

Model	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Total parameters
Fully-Conv	196’608	2’097’152	8’388’608	33’554’432	131’072		44’367’872
Dropout	196’608	2’097’152	8’388’608	33’554’432	131’072	8’256	44’376’128
Depth-wise	1’036	9’408	35’200	135’936	9’217		190’797

Table 1. Number of trained parameters for each discriminator.

Experiment	Bicyclist (%)	Building(%)	Car(%)	Column Pole(%)	Fence(%)	Pedestrian(%)	Road(%)	Sidewalk(%)	SignSymbol(%)	Sky(%)	Tree(%)	Precision(%)	MIOU(%)	Avg Training Time (Minutes/Per Epoch)
1. Target	**60.7**	**80.0**	83.2	**35.7**	**45.9**	**61.6**	**89.3**	**83.9**	12.0	**90.8**	**72.7**	**87.5**	65.1	**5:10**
2. Target + augmentation	52.3	78.9	**84.0**	31.6	45.1	57.3	88.1	83.8	**31.7**	90.3	**72.7**	87.2	**65.3**	**5:10**
3. Fully Conv. discriminator (Camvid length)	0.0	**58.6**	50.9	9.4	2.4	9.4	63.0	12.3	9.8	77.6	35.7	66.7	30.0	14:00
4. Dropout discriminator (Camvid length)	**14.2**	49.6	47.4	10.8	0.7	16.3	66.5	19.6	0.0	70.4	18.2	62.9	28.6	14:40
5. Depth-wise Conv. (Camvid length)	4.5	57.5	50.4	**18.0**	0.70	**20.1**	**66.9**	**26.4**	**16.4**	**87.1**	**44.7**	**69.5**	**35.8**	**10:10**
6. Depth-wise Conv. (IDDA length)	4.9	51.0	**54.0**	0.0	**3.0**	8.7	63.0	18.0	0.0	64.2	26.1	62.6	26.7	79:00
7. Depth-wise Conv. + no Gaussian blur (Camvid length)	4.5	50.9	49.7	16.4	2.1	11.9	64.2	14.1	8.3	83.6	20.5	63.7	29.7	11:10

Table 2. Experimental results: the above metrics are mIoU per each class, average precision per pixel, mean mIoU and average training time. Best values have been highlighted.

racy in almost every class, and the MIOU is 65.1% in the end. To improve the results, we employed data augmentation on our dataset, which resulted in a 20% rise in ”Sign-Symbol” which is the class with the worst results although the MIOU and training time averages are nearly identical.

Domain Adaptation: Table 2 demonstrates that when domain adoption is used, the accuracy of the model falls significantly compared to the Target model. However, keep in mind that the model was trained on a synthetic dataset, so these results are understandable due to the domain gap. The MIOU for a fully convolutional discriminator is 30%. Some classes, such as ”Building”, ”Car”, ”Road”, and ”Sky”, produce satisfactory results, while others, such as ”Bicyclist” and ”Fence,” are almost undetected. This could be because these classes were already challenging to detect in the Target model, and the domain shift makes it even more difficult for the model to detect them. Because we are training the discriminators as well as the generator model, the training period has been increased to 14 minutes. We are also training on both the CamVid and IDDA datasets. The results for the dropout discriminator are nearly identical, but we can see that some classes that were previously more difficult to detect have improved slightly, such as ”Bicyclist”, ”Column pole,” ”Pedestrian,” and ”Sidewalk,” however, the overall precision has slightly decreased.

Best Model: As shown in table 2, the depth-wise discriminator produces the best outcome in the instance of MIOU with 35.8% and 10 minutes of training time.We explained that for this discriminator we tried a lighter architecture, and we can see that it improves accuracy as well as training time by an impressive margin of 5.8%, and it can detect ”Tree” class with a satisfactory result in addition to our best performing classes in previous discriminators. We discovered that when we trained this discriminator on the IDDA dataset size, the model overfitted due to the large difference in size between IDDA and CamVid, and the training time was also 8 times longer. We can also deduce from the last row of table 2 that Gaussian blur greatly aids our model and that without it, almost all classes become less recognizable, hence it is preferable to use it during preprocessing.

7. Conclusions

We investigated real-time domain adaption in semantic segmentation in this research. Our model structure is fairly similar to AdaptSegNet with some important adjustments. To further enhance our work, we tested three different discriminators and, the Depth wise convolution model produced the best results. This approach resulted in 5.8% increase in MIOU and a 30% reduction in training time, leading us to believe that adopting lighter models as a starting point for further study is promising.

References

[1] Emanuele Alberti, Antonio Tavera, Carlo Masone, and Barbara Caputo. IDDA: A large-scale multi-domain dataset for autonomous driving. *IEEE Robotics Autom. Lett.*, 5(4):5526–5533, 2020. 2, 5

[2] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009. 1, 5

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2018. 1

[4] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. 2, 4

[5] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17:59:1–59:35, 2016. 2

[6] Shijie Hao, Yuan Zhou, and Yanrong Guo. A brief survey on semantic segmentation with deep learning. *Neurocomputing*, 406:302–321, 2020. 1

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1, 4

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. 4

[9] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. *CoRR*, abs/1710.03740, 2017. 5

[10] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. 4

[11] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014. 4

[12] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. *CoRR*, abs/1802.10349, 2018. 2, 3, 4, 5

[13] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. *CoRR*, abs/1808.00897, 2018. 1, 3, 5

[14] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6230–6239. IEEE Computer Society, 2017. 1

[15] Sicheng Zhao, Xiangyu Yue, Shanghang Zhang, Bo Li, Han Zhao, Bichen Wu, Ravi Krishna, Joseph E. Gonzalez, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia, and Kurt Keutzer. A review of single-source deep unsupervised visual domain adaptation. *CoRR*, abs/2009.00155, 2020. 2