A class-based styling approach for Real-time Domain Adaptation in Semantic Segmentation

Rosi Gabriele
Politecnico di Torino
s291082@studenti.polito.it

Tampellini Andrea
Politecnico di Torino
s288266@studenti.polito.it

Merlo Matteo
Politecnico di Torino
s287576@studenti.polito.it

***Abstract*—Nowadays, even with the increasing popularity of semantic segmentation, obtaining large real-world labeled datasets can be expensive. To address this issue, synthetic datasets can be used to train networks, but a domain adaptation step is required due to the domain shift. In this work, we implemented an unsupervised domain adaptation algorithm in a real-time setting. Specifically, we evaluate our model in a synthetic-to-real domain adaptation setting. We propose a class-based style-transfer approach to better generalize on images from different domains. Our results demonstrate that it's possible to improve a domain adaptation network by applying different styles to the source images.**

I. Introduction

With the significant progress made by deep neural networks (DDNs) in many computer vision tasks in the last few years, the analysis of large-scale datasets has become a widespread practice. If we analyze more deeply some of these datasets, we can divide them into two categories: *real-world datasets* and *synthetic datasets*. In many situations it's difficult to obtain a large amount of labels, especially for real-world datasets since this operation is made by hand and the process can last from 60 to 90 minutes per image (example taken by Cityscapes [1]). On the other hand, with synthetic datasets the labels are automatically assigned by the computer. One can argue that it's possible to train a model on a source domain with labels (e.g. a synthetic dataset) and apply it to an unlabeled target domain (e.g. a real-world dataset). However, due to the presence of domain shift [2], such transfer might not perform well.

This is the main reason why different *Domain Adaptation (DA)* techniques have been proposed. Indeed, domain adaptation is a specialized form of transfer learning [3] that aims to learn a model from a labeled source domain that can generalize well to a different (but related) unlabeled or sparsely labeled target domain.

In this work, we propose a model based on [4] by adding few more steps in order to improve it. First, we apply a class-based styling like in [5] to source domain selecting a random subset of classes. Second, a real-time segmentation model predicts the output results. Third, a discriminator tries to understand if the output is from the source or target segmentation output. Then, using an adversarial loss, the segmentation model aims to fool the discriminator in order to generate similar distributions in the output space for both the source and target images. Like in [4] we perform the training for the segmentation model and the discriminator in one single step.

The datasets that we adopt in this work are: IDDA [6], a synthetic dataset and CamVid [7], a real-world dataset. The adaptation that we perform is IDDA $\rightarrow$ CamVid.

The contributions of this work are as follows. First, we use a real-time semantic segmentation model in an Unsupervised Domain Adaptation (UDA) setting. Second, we apply a class-based style transfer algorithm in order to improve the network performances.

II. Related Works

Real-time Semantic Segmentation. Real-time semantic segmentation is a trade-off between accuracy and inference speed since lightweight models are used. Some works use small/lightweight networks to achieve high speed like SegNet [8] or E-Net [9]. Zhao et al. [10] proposed a method that uses image cascade to improve the speed. Wang et al. [11] proposed Driving Importance-weighted Loss (DIL), aiming to alleviate the problem of class imbalance. Li et al. [12] proposed a method combining depth-wise separable convolution and dilated convolution, obtaining a shallow network that can perform many operations in parallel.

Domain adaptation. UDA is a very common field of research in recent years. The main reason behind such hype is the necessity to use synthetic data for real-world tasks. Indeed, in a UDA setting we do not need any annotation of the target data.

One of the most researched techniques for UDA is *Adversarial Training*. In this setting we have a segmentation network $\boldsymbol{G}$, that tries to predict the semantic map for the input image, and another network (a discriminator $\boldsymbol{D}$) that tries to predict which domains (source or target) the prediction maps belongs to. Then, with an adversarial loss on the target dataset, the network propagates gradients from $\boldsymbol{D}$ to the $\boldsymbol{G}$. In this way $\boldsymbol{G}$ is encouraged to generate similar segmentation distributions both for source and target domain.

One of the first implementations of this technique was proposed by Ganin et al. [13, 14] with the Domain-Adversarial Neural Networks (DANN) algorithm. Hoffman et al. [15] applied for the first time the adversarial domain adaptation to a semantic segmentation task. In the following year, Hoffman et al. [16] proposed Cycada, a network that applies an image-to-image translation by using CycleGAN [17].

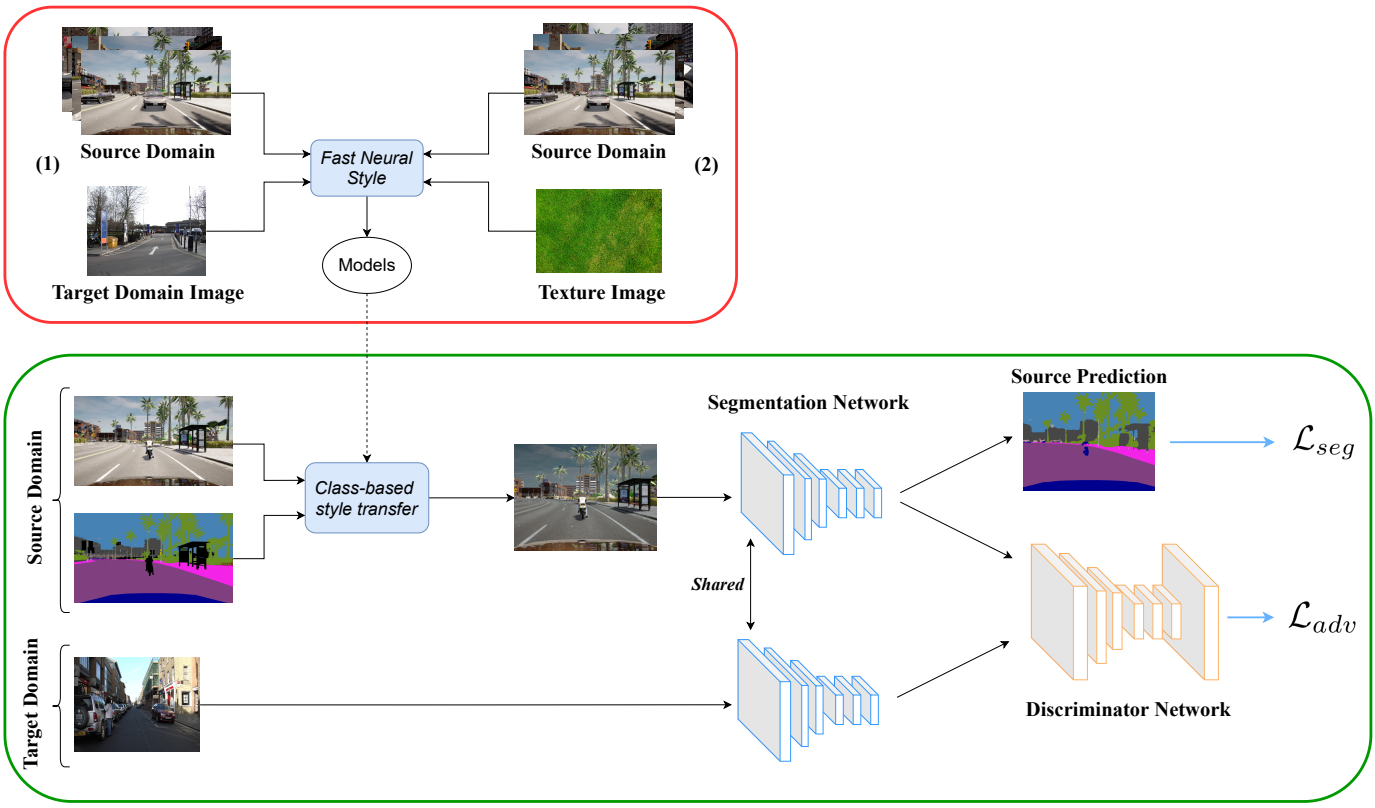

Fig. 1. Architecture overview. Inside the red rectangle we have the **Fast Neural Style** network. The training can be done in two ways: (1) foresees the training on the whole IDDA in order to create a model with the *target domain image* style that is a CamVid image, (2) foresees the training on the whole IDDA in order to create a model with the *texture image* style. On the other hand, in the green rectangle, we have the domain adversarial network. Given the image and the corresponding ground-truth label from the source domain, we forward them to the **class-based style transfer** algorithm that uses the model obtained from the Fast Neural Style network in order to stylize one or more random classes. Then we pass the stylized image, along with the target domain image, to the **segmentation network** to obtain output predictions. On the source prediction, a segmentation loss is computed based on the ground truth. To make target predictions closer to the source ones, we utilize a **discriminator** to distinguish whether the input is from source or target domain. Then an adversarial loss is calculated on the target prediction and is back-propagated to the segmentation network.

III. Proposed Method

A. Overview of proposed model

Fig. 1 represents our network architecture that consists of three main components:

- A style transfer network **S**
- A segmentation network **G**
- A discriminator network **D**

Given an image I_s, extracted from the source set $\mathcal{X}_s \subset \mathbb{R}^{H\times W\times 4}$, and its ground-truth segmentation maps Y_s, we forward them to the style transfer network **S**. This network utilizes pre-trained models in order to transfer the selected style to the selected source classes. Then we forward it to **G** in order to predict the segmentation softmax output P_s.

Meanwhile, given an image I_t (without annotation) extracted from the target set $\mathcal{X}_t \subset \mathbb{R}^{H\times W\times 3}$, we forward it to **G** in order to predict the segmentation softmax output P_t.

Now our goal is to have the two predictions (i.e., P_s and P_t) with a similar distribution. To achieve this we forward them to the discriminator network **D** whose job is to distinguish whether the input is from source or target. Then, with an adversarial loss on the target dataset, the network propagates gradients from **D** to **G** which encourages **G** to generate similar segmentation distributions both for source and target domain.

B. Losses

The objective function of the proposed method is the sum of the segmentation loss $\mathcal{L}_{seg}$ and the adversarial loss $\mathcal{L}_{adv}$:

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

where λ_{adv} is a weight used to balance the two losses.

Segmentation loss. For the segmentation network we test the performance with two different losses:

- *Dice loss* defined as follows:

$$\mathcal{L}_{seg}(I_s) = 1 - \frac{2\sum_{h,w} Y_s^{(h,w,c)} P_s^{(h,w,c)}}{\sum_{h,w} Y_s^{(h,w,c)} + \sum_{h,w} P_s^{(h,w,c)}} \tag{2}$$

- *Cross-entropy loss* defined as follows:

$$\mathcal{L}_{seg}(I_s) = -\sum_{h,w}\sum_{c\in C} Y_s^{(h,w,c)} \log\left(P_s^{(h,w,c)}\right) \tag{3}$$

where Y_s is the ground-truth label for source images and P_s is the segmentation softmax output (i.e. $P_s = \mathbf{G}(I_s)$).

Adversarial loss. To make the distribution of the target semantic prediction closer to the source one, we define an adversarial loss as follows:

$$\mathcal{L}_{adv}(I_t) = -\sum_{h,w} \log\left(\mathbf{D}(P_t)^{(h,w,c)}\right) \tag{4}$$

Discriminator loss. In order to train the discriminator network to distinguish whether the output is from source or target domain we define a loss as follows:

$$\begin{aligned}\mathcal{L}_d(P) = -\sum_{h,w}(1-z)\log\left(\mathbf{D}(P)^{(h,w,c)}\right) \\ +z\log\left(\mathbf{D}(P)^{(h,w,c)}\right)\end{aligned} \tag{5}$$

where $z = 0$ if the sample is from target domain and $z = 1$ if the sample is from source domain.

C. Network Architecture

*1) **Segmentation network**:* for the segmentation network, we adopt BiSeNet (Bilateral Segmentation Network) [18] that is a state-of-the-art approach to Real-time Semantic Segmentation. With this network we have the possibility to choose between ResNet-18 or ResNet-101 [19] as our segmentation baseline model.

BiSeNet is composed by two main components: *Spatial Path* (SP) and *Context Path* (CP).

The aim of the spacial path is to encode significant spatial information by preserving the spatial information contained in the original input image. This component is composed by three convolution layers with $stride = 2$, followed by a batch normalization [20] and a ReLU [21].

On the other hand, context path is used to obtain a large receptive field using a light-weight model and global average pooling. To better refine the features at each stage an *Attention refinement module* (ARM) is used.

The output of these two components could be simply summed up together. But to combine features efficiently a specific module called *Feature Fusion Module* (FFM) is introduced.

*2) **Class-Based Styling**:* Our proposed expansion to the previously described architecture was inspired by the styling technique used by Kim & Byun in [22], and also to the real-time Class-Based Styling (CBS) method displayed by Kurzman et al. in [5]. When training a segmentation network on a synthetic dataset, it's always possible for the network to overfit on specific textures, especially when they are repeatedly used in different images. Using a neural style transfer algorithm is useful to diversify the textures and improve the final performances. In [22], the stylized datasets were prepared beforehand using the StyleSwap [23] algorithm, which cannot be used in a real-time setting. Instead, we exploited the style-transfer algorithm used in CBS to stylize specific classes in a live video. This algorithm was introduced by Johnson et al. in 2016 [24], and it's able to train a model to translate images to different styles in real-time. We introduced the style-transfer step in our architecture as a data augmentation step. Every time an image is taken from the source dataset, a style model is selected and is applied to stylize the image.

Several style models were trained (see Fig. 2), and they can be divided into two main categories:

- **CamVid styles**: these models were trained on images taken from the CamVid dataset. These models were used to change the synthetic textures and make the segmentation network more familiar with the target images colors.
- **Texture styles**: these models were trained on different images and textures that are not easily seen on a street. These models were used to prevent the segmentation network to overfit on the synthetic textures.

The proposed class-based styling algorithm is very similar to the CBS algorithm, and works as follows. For each run, a set of style models I_s is used.

Consider an image I_s, extracted from the source set $\mathcal{X}_s \subset \mathbb{R}^{H\times W\times 4}$, and its ground-truth segmentation maps Y_s. Then, given a style set $\mathcal{T}$, we randomly select a style t. We also randomly select a subset of source semantic classes where each class has a probability p of being selected. p can change or remain constant throughout the whole training, the probability used in our runs is better explained in section IV.

After this initial step, the whole image is stylized using the style t. Then a mask is created based on Y_s, and is used to identify the selected classes that will be stylized from the others. The resulting image consists of the original image in background, and all the stylized classes in foreground.

*3) **Discriminator**:* : the discriminator network was taken from [4], and consists of 5 convolutional layers with kernel 4×4, stride 2 and channel numbers $\{64, 128, 256, 512, 1\}$. Each convolutional layer (with the exception of the last layer) is followed by a Leaky ReLU [25] parametrized by 0.2.

D. Network Training

To obtain the models required to perform the class-based styling, we need to train in a separate way the Fast Neural Style network. This algorithm accepts an input dataset, that can be composed by an arbitrary number of images, and a single target image that provides the style to the images. In this work we train different models with input dataset equal to IDDA and target images as follows (see Fig. 2):

- 5 different CamVid images (generates five models)
- 5 different Texture images (generates five models)

On the other hand, we jointly train the segmentation network and the discriminator network in one stage.

The Stochastic Gradient Descent (SGD) optimizer was used to train the segmentation network, whereas the Adam optimizer [26] was used to train the discriminator. The same hyperparameters proposed in [4] were used, and can be summed up as follows. The SGD optimizer exploits the Nesterov acceleration with momentum set to 0.9 and weight decay set to 10^{-4}, and initial learning rate of 2.5×10^{-4}. The discriminator optimizer, instead, has a momentum set to 0.9 and 0.99 and an initial learning rate of 10^{-4}. Both the networks learning rates decay polinomially with power of 0.9. Furthermore, the

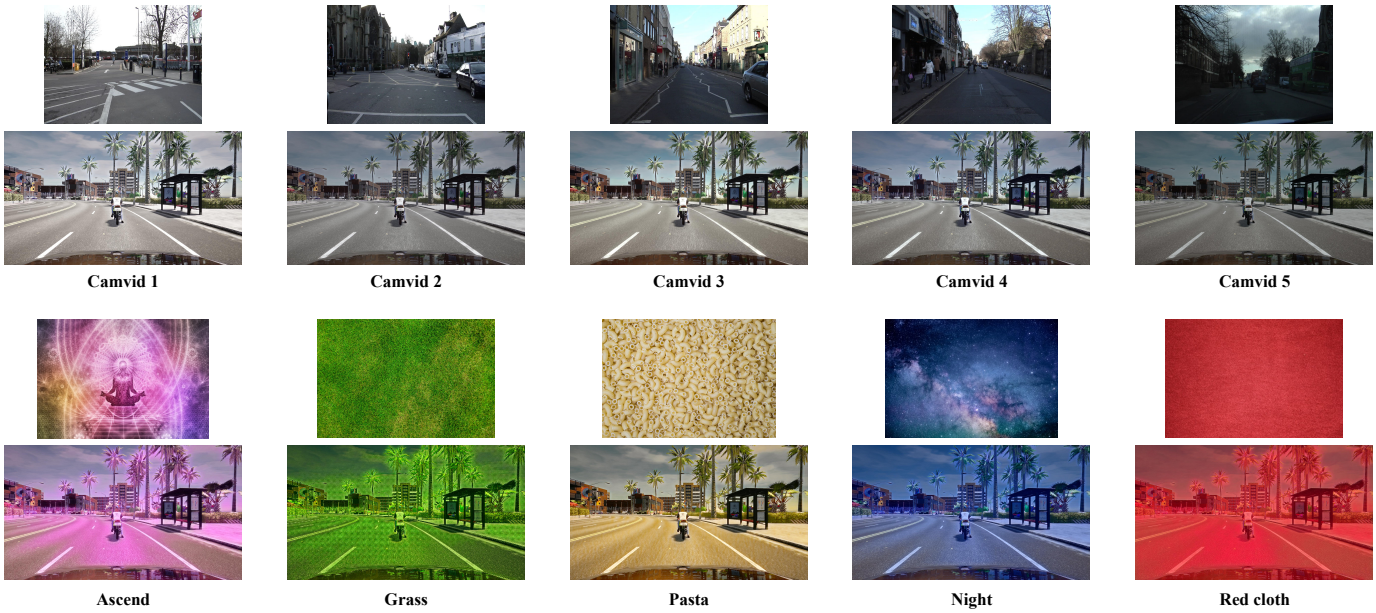

Fig. 2. Style models used for the class-based styling.

adversarial loss is multiplied by the λ_{adv} constant equal to 0.001.

The Fast Neural Style network is trained for 4 epochs on the entire length of IDDA. We set style-weight as 5×10^9 and content-weight as 10^5. The learning rate is set as 10^{-3} and the backbone as VGG16 [27].

IV. Experiment

In the following sections, we are describing the experiments performed and the results obtained while building and testing the proposed network architecture. The code is available at https://github.com/Gabrysse/CBS-realtimeDA-semSeg. Our experiments were divided in the following three steps:

1) At first, we tuned the segmentation network hyperparameters in order to perform well on the CamVid dataset;
2) Then, we added a discriminator to the architecture in order to implement unsupervised domain adaptation between the source dataset IDDA and the target dataset CamVid;
3) Finally, we tested different approaches to improve the domain adaptation performances.

Each of these steps is better explained in the following subsections.

A. Datasets

As previously mentioned, two datasets were used: CamVid as the source domain and IDDA as the target domain.

CamVid is a dataset containing 468[1] labeled images. These images are 960x720 pixels and were taken from the real world. It's labeled with 11 different classes.

[1]To train the network, we used both the training images and the evaluation images of the CamVid dataset

Model	Backbone	Loss	Epoch	Accuracy	mIoU
BiSeNet	ResNet-18	Dice	50	85.6	59.7
			100	86.9	**63.0**
	ResNet-101		50	84.3	55.6
			100	**87.0**	62.1
BiSeNet + Augumentation	ResNet-101	Dice	100	**88.0**	**67.1**
AdaptSegNet + Augumentation	ResNet-101	CrossEntropy	50	**67.6**	**31.9**
		Dice		64.5	29.1
		CrossEntropy	100	66.1	30.8
		Dice		64.4	28.2
		AdvEnt	50	65.7	30.5

TABLE I
Results of the first two steps of our experiment.

IDDA is a dataset containing 3379[2] labeled images. These images are 1920x1080 and were generated by a computer simulator. It's labeled with 27 classes, but it's possible to map them to match the CamVid labels. To match the CamVid image size, we performed a downscaling of both the images and the labels to 1280x720, and then a random crop to 960x720.

B. Segmentation Network Tuning

In the first step, we exploited the BiSeNet network to perform semantic segmentation on the CamVid dataset. The network was trained with two different backbones, ResNet-18 and ResNet-101, and their performances at 50 and 100 epochs were compared in order to find the best configuration and get an upper bound for the next steps results. The dice loss was used in all these trainings. The results of this experiment are presented in the first part of Table I. It's possible to

[2]Note that we used a reduced version of IDDA where the viewpoint is "Audi", the town is "Town 1" and the weather is "Clear Noon".

Method	bicycle	building	car	pole	fence	pedestrian	road	sidewalk	sign-symbol	sky	tree	Accuracy	mIoU
AdaptSegNet (50 epoch, CE)	*1.10*	*58.19*	*28.90*	*16.93*	*0.33*	*14.80*	*63.99*	*15.88*	*22.66*	*87.41*	*40.21*	*67.6*	*31.9*
Camvid1	4.62	57.71	20.92	19.21	0.13	17.40	57.25	12.38	15.44	71.04	39.68	63.4	28.7
Camvid2	3.48	56.61	20.70	**20.59**	0.17	16.69	56.69	16.20	19.85	65.16	42.31	62.8	29.0
Camvid3	2.92	56.96	25.90	16.86	1.02	14.46	62.98	16.30	15.99	87.72	38.94	66.7	30.9
Camvid4	4.99	58.27	24.25	18.70	0.10	19.24	55.04	17.58	26.48	72.19	41.78	63.2	30.8
Camvid5	3.34	**61.22**	**37.12**	19.25	0.31	**25.50**	63.37	17.67	24.75	87.84	41.19	**68.7**	**34.7**
Ascend	**6.46**	53.80	25.45	18.35	0.54	14.26	57.23	15.76	28.19	63.27	32.74	61.3	28.7
Grass	3.18	57.00	22.00	18.28	0.17	15.44	61.22	16.19	9.78	87.41	33.84	65.9	29.5
Night	4.63	59.52	31.77	16.88	1.11	16.69	58.35	15.46	25.18	73.34	37.05	64.7	30.9
Pasta	6.13	53.83	18.74	17.64	0.57	14.15	**65.41**	**17.79**	26.71	88.54	33.53	66.0	31.2
Red Cloth	6.07	49.96	33.11	16.32	0.99	14.18	63.03	13.56	26.87	61.79	29.32	61.5	28.7
All Camvid	0.00	60.98	31.26	19.63	0.07	19.19	61.46	16.74	25.01	**89.14**	**44.00**	68.0	33.4
All Texture	0.00	59.83	36.37	16.46	**1.16**	0.00	61.65	13.72	21.63	87.72	39.06	67.7	30.7
All styles	0.96	60.97	35.55	19.17	0.55	16.38	62.83	17.53	**29.00**	88.10	39.41	68.0	33.7

TABLE II
RESULTS OF ADAPTING IDDA TO CAMVID

see that training the network for 100 epochs seems to yield better results than training it for 50 epochs, while there is no significant difference between using ResNet-18 and ResNet-101 as the backbone.

After the first results were obtained, the network was trained again while using two different data augmentation techniques on the dataset. The implemented techniques are horizontal flips and Gaussian blur on the whole image. Each time an image was taken from the dataset, there was a 1/4 probability for each of the following scenarios to take place: no change was made, the image was flipped horizontally, the image was blurred, the image was both flipped horizontally and blurred. This addition improved the results especially when using the ResNet-101 backbone as shown in the second part of Table I. After assessing these results, we decided to keep using the ResNet-101 backbone to conduct further experiments, and we moved to the second step of our experiment.

C. Domain Adaptation and Adversarial Training

In the second step, the network architecture was expanded in order to perform unsupervised domain adaptation on two different datasets. In the following experiments, we used the IDDA dataset as the labeled source dataset, while the CamVid dataset was used as the unlabeled target dataset. We also introduced a discriminator network to perform adversarial training between target and source domain.

We ran multiple trainings to test different configurations: during each epoch, the network was trained with all the images from the CamVid dataset and a random subset of images taken from the IDDA dataset to match the size of CamVid. The most significant results of our runs are presented in the last part of Table I.

At first, the network was trained for 100 epochs to compare dice and crossentropy losses. The best results were obtained using the crossentropy loss that was able to reach a pixel precision of 66.1 and a mIoU of 30.8. Considering that the best results of both runs were achieved in the first 50 epochs, and didn't significantly improve over the last 50 epochs, the training of the next runs was reduced to 50 epochs in total. By reducing the number of epochs to 50, the performance improved to a pixel precision score of 67.6 and a mIoU score of 31.9 with crossentropy. We also tried to adapt the AdvEnt loss proposed by Vu et al. in [28] to our architecture, but there was no performance improvement.

D. Class Based Styling

After assessing that the best performances can be reached training over 50 epochs using the crossentropy loss, this configuration was used to test the expansion described in Section III.

For each image, the probability p to stylize a class was set to 0.5 for the whole duration of the training. The network was trained several times with different combinations of style models t, that can be summed up as follows:

- 1 single CamVid style,
- 1 single Texture style,
- 5 different CamVid styles,
- 5 different Texture styles,
- 5 different CamVid styles and 5 different Texture styles.

Method	Accuracy	mIoU
AdaptSegNet (50 epoch, CE)	*67.6*	*31.9*
CBS Camvid5	*68.7*	*34.7*
(1) Global stylization Camvid5	64.9	31.2
(2) Global stylization Camvid5 (even batch)	65.3	31.1
(3) CBS Camvid5 (probability 1)	64.9	30.9
(4) CBS Camvid5 (probability 2)	63.6	30.6
(5) CBS Camvid5 per-batch	62.0	29.6
(6) CBS Camvid 1, 2, 3, 4	64.9	31.0

TABLE III
RESULTS OF ABLATION AND VARIATION STUDY

All the results are reported in Table II. It's possible to see that while some runs didn't improve the performances obtained without styling, other runs were able to significantly improve both the mIoU and pixel precision scores. In particular, the runs with multiple styles achieved better results overall. Unexpectedly, the best results were obtained by the network trained with the *Camvid5* style only, which was able to reach a mIoU score of 34.7 and a pixel precision score of 68.7 at the end of the training.

After finding out the best styles to train the networks, we used the *Camvid5* style to train again slightly modified versions of the network. In particular, we wanted to test the network performance when changing the following aspects:

Ablation of Class-Based styling: the aim of this test is to verify the utility of the class-based styling approach. The network was trained two times: the first run all the source images were completely stylized (1) and the second run only half of the batches were stylized (2).

Variation of class styling probability: as explained in section III, each class of each image has a probability p of being stylized. In these experiment, we wanted to test whether it can be useful to stylize classes that are harder to predict more often. The networks were trained two times, with p calculated as follows: after each epoch, p was changed to $1 - IoU\_class$ in the first run (3), and $(1 - IoU\_class)^2$ in the second run (4).

Variation of batch styling: in the previous runs, every image taken from the dataset had a randomly selected style that was applied to a randomly selected subset of classes. The network was trained one time with the following variation: the style and the subset of classes were the same for every image of each batch (5).

Ablation of the best style: to verify the robustness of the followed approach with multiple styles, the network was trained again with all the CamVid styles previously used except for *Camvid5* (6).

The results of these runs can be seen in Table III. In general, none of these runs improved the previously obtained results.

E. Discussion

With the proposed expansion, we were able to improve the baseline results achieved by the network used in the second step of our experiment. We believe that there is still room for improvements. For example, it could be interesting to perform a finetuning of the styles based on the single class IoU scores and use the best styles for each specific class. In particular, the runs that used mixed styles achieved improved results, but still underperformed while predicting harder classes such as the *bicycle* class where single styles achieved better results. Another aspect that needs to be taken into account is the consistency and reproducibility of the results. In the described runs, only a random subset of the source dataset was used to train the network in each epoch. Furthermore, the proposed expansion includes stochastic features such as the style model being and the stylized class subset being selected randomly for each image. It could be interesting to reproduce the experiments and train the network with the entire source dataset.

V. CONCLUSION

In this work, a style-transfer algorithm was used on the source dataset to achieve better results in an unsupervised domain adaptation task. Experimental results show that a class-based styling approach was able to outperform the initial domain adaptation network. Even if the best results were achieved using a single style model, we believe that using multiple styles together in conjunction with finetuned style models and hyperparameters could lead to even better results.

REFERENCES

[1] 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," 2016.

[2] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," *CVPR 2011*, pp. 1521–1528, 2011.

[3] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[4] 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," 2020.

[5] L. Kurzman, D. Vazquez, and I. Laradji, "Class-based styling: Real-time localized style transfer with semantic segmentation," 2019.

[6] E. Alberti, A. Tavera, C. Masone, and B. Caputo, "Idda: a large-scale multi-domain dataset for autonomous driving," 2020.

[7] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, 2008.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[9] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," 2016.

[10] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," 2018.

[11] W. Wang and Z. Pan, "Dsnet for real-time driving scene semantic segmentation," 2019.

[12] G. Li, I. Yun, J. Kim, and J. Kim, "Dabnet: Depth-wise asymmetric bottleneck for real-time semantic segmentation," 2019.

[13] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," 2015.

[14] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," 2016.

[15] J. Hoffman, D. Wang, F. Yu, and T. Darrell, "Fcns in the wild: Pixel-level adversarial and constraint-based adaptation," 2016.

[16] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," 2017.
[17] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2020.
[18] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," 2018.
[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.
[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.
[21] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," *Proceedings of the 14th International Conference on Artificial Intelligence and Statisitics (AISTATS) 2011*, vol. 15, pp. 315–323, 01 2011.
[22] M. Kim and H. Byun, "Learning texture invariant representation for domain adaptation of semantic segmentation," 2020.
[23] T. Q. Chen and M. Schmidt, "Fast patch-based style transfer of arbitrary style," 2016.
[24] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," 2016.
[25] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.
[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.
[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.
[28] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation," 2019.