Real Time Domain Adaptation in Semantic Segmentation

William Forestiere
S278428
S278428@studenti.polito.it

Giovanni Poggio
S287775
S287775@studenti.polito.it

Juan Aragon
S291466
S291466@studenti.polito.it

Abstract

Semantic segmentation is one of the most challenging tasks in computer vision. Real-world datasets are few and small, and simulated ones introduce hurtful domain shifts. Moreover, a field like autonomous driving would get many benefits from accurate, fast and lightweight models.

With this project, we try to address these issues: in particular, using BiSeNet as the base model, we perform unsupervised domain adaptation training with an adversarial objective in the output features space, using IDDA as the source dataset and CamVid as the target. We then use the trained model to generate pseudo-labels for the target domain, and with these we fine tune the model in a self-supervised manner.

Our extension provides a significant improvement (+5.1 % mIoU) over the adversarial only approach, reaching a maximum mIoU over the test set of CamVid of 42.84 %.

1. Introduction

Semantic segmentation is a challenging task in machine learning with the objective of classifying (i.e. assigning category label to) each pixel of a given image. As it is the case for the whole field of computer vision, this task has also benefited from the advent of deep-learning enabled by the development of capable hardware architectures[4].

Like most solutions based on neural networks, large training datasets are necessary in order to obtain the best performances. However, generating ground-truth labels for such a task is labour intensive: a human has to manually annotate each and every pixel of typically high resolution images. This translates into hundreds of hours of work for labelling few hundreds images, and as a consequence datasets of this kind are few and scarcely populated.

A possible workaround to this problem would be using computer generated images, for which pixel-wise annotations come basically for free. This approach, nevertheless, has its drawbacks: pictures from simulations differ noticeably from real ones, and models trained exclusively on such images tend to perform quite poorly when provided with concrete examples.

This phenomenon is called domain-shift and presents itself as a discrepancy between the source and target feature distributions, to which networks, unlike human learners, often fail to adapt. To lessen its severity, several techniques of Domain Adaptation have been developed, based for example on adversarial training[8].

Semantic segmentation is critical in fields such as computer-aided diagnostics and autonomous driving. The latter, in particular, requires the model not only to produce accurate predictions, but also to do so with great speed. To achieve such performances a network with this goal should be lightweight, minimizing the computational cost of the implementation.

The purpose of this report is to illustrate the methods and outcomes of a project tackling these issue in the context of the course *Machine learning and Deep learning*, part of the master degree in *Data science and engineering* offered by *Politecnico di Torino* in A.Y. 2020-21. In particular, we use BiSeNet[7] with ImageNet pre-trained ResNet backbone as the starting point. We test it on the CamVid dataset[2] to obtain a reference accuracy measure. We then perform Unsupervised Domain Adaptation (UDA) by employing the network as the base model of the AdaptSegNet training template[6], using IDDA[1] as the source dataset and CamVid as the target. Finally, we introduce a Self-Supervised Learning (SSL) step inspired by Bidirectional Learning[5] to further improve the obtained results.

2. Related work

BiSeNet[7] is used as the baseline model for our Semantic Segmentation experiments. It works by having two different paths: a Spatial Path which preserves the spatial size of the input and encodes rich spatial information, and a Context Path to provide a large receptive field. This is coupled with an ARM (Attention Refinement Module) which further refines the features at the end of the Context path, and with a FFM (Feature Fusion Module), that is used to combine the output features of both the Context and Spatial Path.

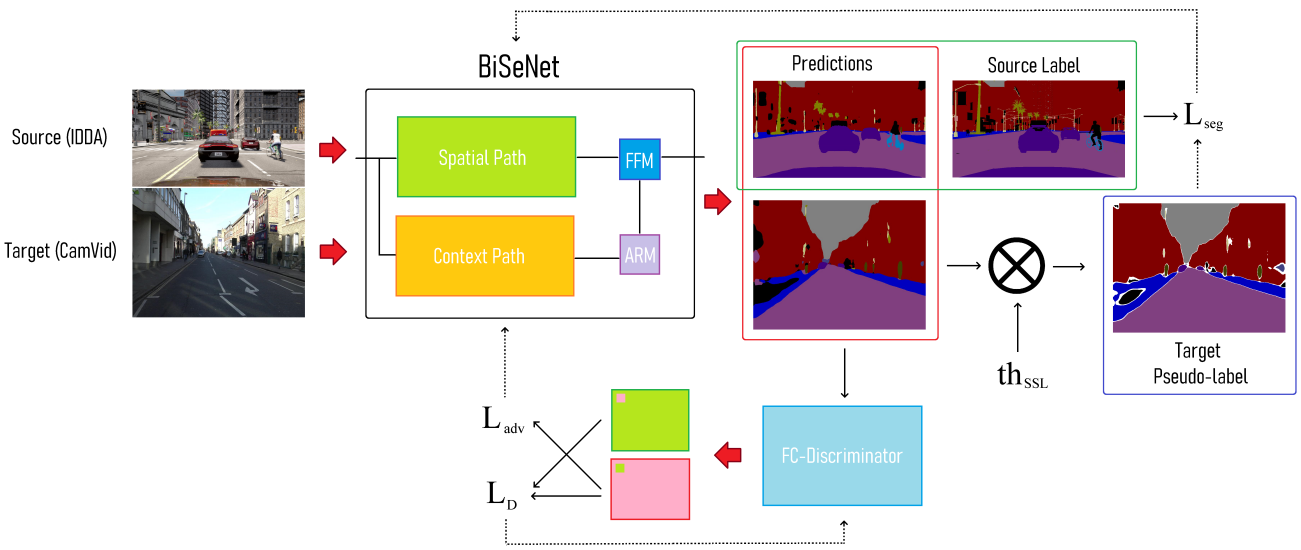

Figure 1. Visualization of the network architecture, workflow and loss propagation

Data Augmentation The best way to increase the generalization capabilities of a model is training with more data, but more often than not this is not feasible because of the complexity of the labeling process. Data Augmentation is an approach to solve this problem: by altering the training images that are already present while keeping their labels unchanged, the network learns how to generalize better because it's getting trained on data that is slightly different every time. Data Augmentation has been proven to be essential in training any kind of Deep Neural Networks, and **RandAugment** [3] in particular, an automated augmentation strategy, has led to state-of-the-art results over other augmentation strategies.

Domain Adaptation (DA) has been the focus of intense research in recent years. The goal of DA is to address the domain shift between source and target domains, which makes a network unable to perform well on the target data because of such discrepancy. A common way to perform DA is with Adversarial discriminative methods, where a discriminator is trained to distinguish between source and target images, therefore closing the domain gap.

In our experiments we work in a specific DA learning paradigm, the **unsupervised** one. With Unsupervised domain adaptation there's a new limitation: images from the target domain are unlabeled. This is quite representative of real world conditions, where we might train our model in a source domain, but then expect it to perform well in a different domain that we have no information about (i.e. no labels). Therefore there's the need to overcome the domain-shift with even less information than before, by exploiting the source domain to the maximum.

We decided to perform **Unsupervised Adversarial DA in the output space** using the method described in **AdaptSegNet**[6]. By having a segmentation network M and a discriminator D, we first optimize M using source images with annotations, then predict target labels and train our discriminator D to distinguish between source and target domain. With an adversarial loss on the target prediction, the training encourages M to generate similar distributions on both source and target images, so as to fool the discriminator by making their segmentation distribution as similar as possible.

Self-Supervised Learning is an approach that has been exploited to further increase many models' generalization ability. SSL is usually conducted by defining a pretext task, such as rotating or coloring, which can be solved by self-supervision. Learning how to perform this apparently unrelated task helps the network to better understand the underlying characteristics of an image. Such a SSL method is defined in Bi-Directional Learning[5], which produces pseudo-labels from an Adversarial trained model that are later used to further train the network, resulting in an increase of performance.

3. Method

3.1. Loss functions

The segmentation loss used is a variant of the **Dice loss**, in turn derived from the *Dice coefficient*. This is a very common measure of the accuracy of segmentation models, equivalent to the F-score. It is defined as

$$D = \frac{2|A \cap B|}{|A| + |B|} = \frac{2TP}{2TP + FP + FN} \quad (1)$$

where A and B are the prediction and the target label, and the expression on the right is the form it takes for a binary classification task. The loss inspired by such coefficient can be obtained by employing the confidence probability output of the network as the prediction and the one-hot encoding of the label as the target. The implementation in the base repository of the project differs only by lacking the 2 at the numerator. The formulation then becomes:

$$\begin{aligned} & L_{dice}(\hat{Y}, Y) = \\ & = 1 - \frac{\sum_{i,j} \sum_c \hat{Y}_{i,j,c} Y_{i,j,c}}{\sum_{i,j,c} \hat{Y}_{i,j,c} + \sum_{i,j,c} Y_{i,j,c}} \\ & = 1 - \frac{\sum_{i,j} \hat{Y}_{i,j,c:Y_{i,j,c}=1}}{2HW} \end{aligned} \tag{2}$$

where $\hat{Y} = softmax(M(X))$, M being the model and X the image with label Y. The final form of numerator is due to the one-hot encoding of the label, and that of the denominator by also taking into account that the softmax confidence of each pixel sums to 1.

The **adversarial loss** is used to train M to segment the target images without being provided with the respective labels. It in fact incentivizes the model to generate inferences similar to those it produces on the source, maximizing the probability of fooling the discriminator. The function is the same employed by [6], i.e.

$$L_{adv}(\hat{Y}) = -\sum_{h,w} log(D(\hat{Y})_{h,w}) \tag{3}$$

where D is the discriminator. This is itself a fully convolutional network with the goal of distinguishing whether the prediction was made starting from an image belonging to the source or target datasets. The **discriminator loss** is a *binary cross entropy* (BCE) loss formulated as follows:

$$\begin{aligned} L_D(\hat{Y}) = -\sum_{h,w} & [(1-z)\log(D(\hat{Y})_{h,w}) \\ & + z\log(1 - D(\hat{Y})_{h,w})] \end{aligned} \tag{4}$$

where $D(\hat{Y})$ is the output of the discriminator, i.e. the down-sampled pixel-wise probability of the prediction being from the source domain; z is then (deterministically) equal to 1 if the sample is from the target dataset and equal to 0 if from the source.

3.2. Implementation

For the first sets of experiments the network is trained using only the segmentation loss on CamVid[2] images and labels: for each epoch, and at each iteration over a mini-batch, the loss computed as in equation (2) is back-propagated in order to learn the task at hand. These experiments are carried out mainly to discover the optimal set-up in terms of backbone used and data augmentation policy.

Next, the model is coupled with a fully convolutional discriminator network as in [6] in order to perform Domain Adaptation in the output space in an unsupervised manner. It is therefore presented with the images and annotations from the synthetic source domain (the IDDA dataset[1]), and the images alone for the now target dataset CamVid. On the former the supervised segmentation loss is computed, again using equation (2), whereas the latter are not compared to their respective labels.

Instead the output prediction is fed to the discriminator. The adversarial loss is then applied as in equation (3) and back propagated through the segmentation network alone, thus exploiting the already mentioned adversarial learning paradigm, which over the last years was proven successful in many similar situations.

Finally, the segmentation inferences from both the source and the target sets are used to train only the discriminator, by propagating the corresponding losses computed as in equation (4). An algorithmic overview of this process is shown in **Algorithm 1**.

A difference with respect to what was implemented in [6] is that the prediction alignment via UDA is performed only on the final output prediction instead of also on the intermediate layers. This was done in accordance to the project guidelines as, although not the one chosen by this group, it could be considered as an extension to further improve the performances.

Algorithm 1: Adversarial UDA

Data: $(\mathcal{S}, Y_{\mathcal{S}}), (\mathcal{T}, Y_{\mathcal{T}} = \varnothing)$
Models $M,\ D$
for $epoch \leftarrow 0$ **to** N **do**
  train M on $(\mathcal{S}, Y_{\mathcal{S}})$ with (2)
  train M on $(\mathcal{T})$ with (3)
  train D with (4)
end

As an extension aimed at bettering the capabilities of model, we chose to implement the self-supervised protocol outlined in [5]. Unlike this work, we do not pre-process the source images with CycleGAN, which would align the images also in the input feature space by transferring the target style to the source images fed to the model. Instead, we directly apply the model obtained from the previous step on the target images, in order to generate pseudo-labels.

These are comprised of only those pixels for which the inference is given with high confidence, whereas the others are given a newly included ignore annotation. To do so, the one-hot encoding of the target label is given the possibility of being made of only zeros, resulting in the numer-

ator of equation (2) excluding low-confidence pixels from the summation. Finally, the implementation of the function was also modified not to count those same pixels at the denominator, resulting in the following loss

$$L_{dice}(\hat{Y}, Y) = \\ = 1 - \frac{\sum_{i,j} \hat{Y}_{i,j,c:Y_{i,j,c}=1}}{2(HW - \sum_{h,w} \mathbb{1}_{Y_{h,w,c}=0 \forall c})} \tag{5}$$

Two noticeable properties of this equation are the fact that it is proportional to one exploited previously — they only differ by a multiplicative factor of the dice coefficient, around 4% as the ratio of negligible pixels — and that it coincides with the dice loss when no ignore label is present. It was thus deemed safe and appropriate to use for both computations of the segmentation loss.

This is indeed what happens, as outlined in **Algorithm 2**, during the self-supervised leaning phase: an extra step is added to the standard adversarial procedure in which the generated pseudo-labels are used in place of the missing ground-truth of the target domain, and the segmentation loss computed is back-propagated to the model, with a multiplier λ_{SSL} which provides the possibility of tuning its effect on the network.

Algorithm 2: Adversarial UDA with SSL

Data: $(\mathcal{S}, Y_{\mathcal{S}}), (\mathcal{T}, Y_{\mathcal{T},0} = \varnothing)$
Models $M_0,\ D$
for $k \leftarrow 1$ **to** K **do**
  generate pseudo-labels:
  $Y_{\mathcal{T},k} = M_{k-1}(\mathcal{T})$
  for $epoch \leftarrow 0$ **to** N **do**
    train M_k on $(\mathcal{S}, Y_{\mathcal{S}})$ with (5)≡(2)
    train M_k on $(\mathcal{T}, Y_{\mathcal{T},k})$ with λ_{SSL}(5)
    train M_k with (3)
    train D with (4)
  end
end

4. Datasets

CamVid[2] is a dataset made of real images recorded from high-definition camera mounted on the dashboard, which provides a similar field of view to that of the driver. This dataset is composed of 701 humanly labeled images extracted from the acquired footage, and it's divided as follows: 367 training samples, 101 validation samples and 233 test samples. In our experiments we used the training and validation samples for training our models, while using the test samples as validation. Each pixel may belong to one of 32 semantic classes, however we are interested in labeling only 11 of them. The eleven classes of interest are: Bicycle, Building, Car, Pole, Fence, Pedestrian, Road, Sidewalk, SignSymbol, Sky and Tree. All other classes will be grouped into a single "void" class that will be predicted normally.

IDDA[1] is an artificial dataset obtained with the CARLA simulator, which is able to produce pixel perfect labels for any given scene. It was generated with a number of different settings, varying the scenario and time of the day, the weather conditions, the vehicle and thus the viewpoint of the simulations. Our version of this dataset is a very small subset of it composed of 3379 training images. Each pixel can belong to one of 24 semantic classes, but again, we have been using only 11 of these, corresponding to the same classes of interested defined before.

5. Experiments

In this section, we briefly describe all the phases carried out along the project, showing the main goals and the obtained results. Unless specified otherwise, all experiments are performed using a polynomially decaying learning rate for the main network starting at 2.5×10^{-2}, with *power* coefficient 0.9 and 100 maximum iterations.

5.1. Semantic Segmentation on CamVid

This step consisted of training and testing the BiSeNet semantic segmentation model using only the CamVid dataset. The starting starting point is based on the repository available at `https://github.com/taveraantonio/BiseNetv1.git`.

5.1.1 Backbone selection

We tested two versions of the same model changing only the backbone architecture between ResNet-18 and ResNet-101, both pre-trained on ImageNet. The goal of this test was to define the baseline/upper-bound for the domain adaptation phase since the dataset will then be employed as the target domain.

At this stage we do not employ any kind of data augmentation technique, leading to noticeable overfitting over the training set, with the validation accuracy in terms of mIoU lagging behind, stalling and even decreasing while that on the training set is still rising.

From the results shown in Table 1, however, it is clear that ResNet-101 outperforms ResNet-18 in both training and validation, achieving top-1 test accuracy of 62.00% in terms of mIoU. It was thus selected as the backbone for all subsequent experiments.

	Test mIoU	Train mIoU
Resnet-18	57.50%	84.87%
Resnet-101	**62.00%**	88.04%

Table 1. Comparison between backbone architectures

5.2. Data Augmentation

5.2.1 Scale Jittering analysis

A common procedure in data augmentation consists in applying scale jittering and random crops at training time, as opposed to using a single scale as in the previous experiments. As all augmentation techniques, this is done in order to prevent overfitting and increase the generalization capabilities of the model by exposing it to a wider range of inputs.

The results from Table 2 show that training the model with images at different scales leads to significantly better results (+2.0% test mIoU). The two scale ranges tested improved the previous result in a similar way, by increasing the mIoU accuracy to 64%, yet the larger range (from 0.5 to 2.0) shows a lower degree of overfitting as it can be seen from the lower train mIoU and thus lower difference between training and testing. This suggests that by training for longer, letting the accuracy on the training set reach comparable levels to those achieved with the narrower scale, also the validation mIoU should increase and overtake the weaker scale jittering policy. The wider range was thus chosen for further experiments in virtue of its better generalization properties.

Scale Jitter	Test mIoU	Train mIoU
$[1]$	62.00%	88.04
$[0.8, 1.2]$	64.07%	86.29
[**0.5,2.0**]	64.00%	78.07

Table 2. Comparison between scale jittering ranges

5.2.2 RandAugment tuning

Considering the critical role of data augmentation in the improvement of deep learning models and based on the results obtained by [3], we introduced a stochastic procedure for augmenting the training images by adapting the implementation of RandAugment to deal with the peculiar labels distinguishing semantic segmentation. This was queued to the scale jittering and performed with a probability of $1/2$. The goal was to further improve the accuracy, improving the model's capacity by exposing it to altered samples which with some probability can be wildly different from the original image.

At this phase, we took into consideration 17 transformations, 10 of which are applied at the pixel level, and 7 are affine image transformations. The complete list of transformations together with their maximum magnitude ranges is reported in Table 3. These are randomly sampled with replacement and applied with a scaled intensity. The pragmatic interest of RandAugment is the possibility of testing very different augmentation policies by changing only two parameters, making the tuning of the strategy fast and simple. These are:

- N, the number of transformations applied;
- M, the magnitude of the transformations.

In our implementation M is scaled by a factor of $1/30$, meaning that $M = 30$ provides the maximum intensity. We performed a pseudo-random search over four configurations, the results of which are reported in Table 4. The best performance was obtained when only a few but strong transformations were considered ($N = 5$, $M = 22$), reaching a test mIoU of 66.87% (+2.9% over the model without RandAugment). From now on we will reference this result as the baseline/upper bound for the domain adaptation phase.

	min_value	max_value
AutoContrast	0	1
Equalize	0	1
Invert	0	1
Rotate	0	30
Posterize	0	4
Solarize	0	256
SolarizeAdd	0	110
Color	0.1	1.9
Contrast	0.1	1.9
Brightness	0.1	1.9
Sharpness	0.1	1.9
ShearX	0	0.3
ShearY	0	0.3
CutoutAbs	0	40
TranslateXabs	0	100
TranslateYabs	0	100
Flip	0	1

Table 3. List of transformations for augmentation

N, M	nickname	**Test mIoU**	Train mIoU
$0, 0$	none	64.00%	78.07
$4, 10$	few weak	62.99%	77.22
$\mathbf{5, 22}$	**few strong**	**66.87%**	81.43
$11, 5$	many weak	64.73%	80.35
$12, 25$	many strong	60.48%	70.38

Table 4. Comparison between RandAugment's parameters N & M

5.3. Unsupervised adversarial domain adaptation

As it was mentioned previously, datasets for semantic segmentation are few and small, but Deep Learning tech-

niques work best when plenty and diverse data is provided. To overcome this scarcity many sizeable synthetic datasets have been created. However, the difference in appearance between real and simulated images introduces a domain shift which significantly hurts performance. The field dealing with minimizing this barrier is Domain Adaptation, the focus of this next experiment.

As described earlier, we implement adversarial domain adaptation in the output space on the blueprint of [6]. We performed three experiments with different learning rate schedulings:

1. $lr_0 = 2.5e-2$, $max\ epochs = 100$
2. $lr_0 = 2.5e-3$, $max\ epochs = 100$
3. $lr_0 = 5e-3$, $max\ epochs = 200$

The last experiment was set to perform 200 epochs, which changes the LR scheduling, but stopped early at 100 both for comparison and for the purpose of the extension. Moreover, as the portion of the IDDA dataset used contains more than $7\times$ the images with respect to the CamVid training subset, we opted for limiting an epoch to the smaller cardinality, shuffling the source images to ensure variability. The results are reported in Table 5.

lr_0	max_epochs	**Test mIoU**	Src mIoU
2.5e-2	100	36.40%	69.59
5e-3	200*	**37.77%**	62.27
2.5e-3	100	36.40%	57.97

Table 5. Performance after adversarial training with different learning rates. *All models are trained for 100 epochs, either naturally or with early stopping.

It is now clear that the accuracy suffered a significant decrease due to the domain shift that can not be completely overcome with adversarial training. What follows is our attempt at retrieving part of this difference with our proposed extension.

As we already mentioned before, our implementation is inspired by the self-supervised semantic segmentation approach described in [5], where where the model self-trains on pseudo-labels generated at the previous outer iteration. At first, no label is provided for the target domain and training proceeds as (and in fact coincides with) the unsupervised domain adaptation carried out in the step before.

Then, the best predicting model is used to generate pseudo-labels to be used as surrogate ground-truth for computing the segmentation loss also on the target images. The confidence threshold for the pixels to be included was 90%, as this was the best performing in [5].

An overview of the overall network architecture and workflow is shown in Figure 1.

We trained for 50 iterations starting from last (adversarial only) version of the model. This meant starting with the same scheduling ($lr_0 = 5e-3$, $max\ epochs = 200$) but from epoch 101, effectively reducing the learning rate to $2.67e-3$. We proceeded like this, since the model was already semi-trained, with the goal being to fine tune it with the newly introduced task.

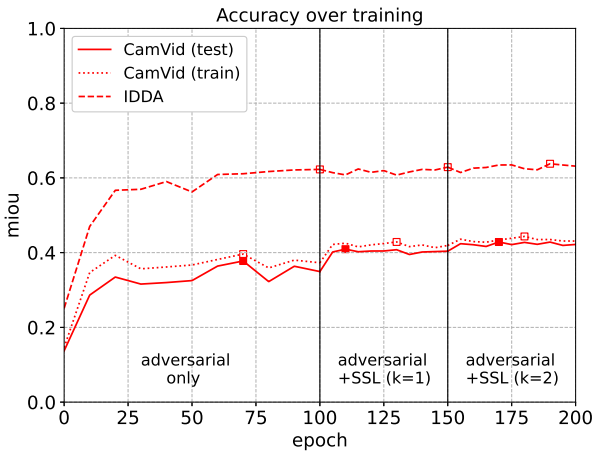

Figure 2. Evolution of mIoU across training phases. Notice the step increase at the beginning of each SSL iteration.

We obtained a top-1 mIoU of almost 41%, and decided to repeat the process, producing fresh pseudo-labels with the newly trained model and again training for 50 epochs (151 to 200). This corresponds to $k = 2$ using the notation adopted in the description of Algorithm (2).

The final accuracy in terms of mIoU obtained by the model on the CamVid test set was 42.84%, a more than 5% increase with respect to the one obtained training only with the adversarial paradigm. Our results are summarized in the table below:

	Test mIoU	Gain	Src mIoU
M_0 (Adv)	37.77%	–	62.27
M_1 (SSL)	40.93%	+3.16%	62.88
M_2 (SSL)	**42.84%**	+5.07%	63.78

Table 6. Final obtained results

The performance for all semantic classes at the different steps of the extension can be seen in Table 7. The evolution of the network's performance on the target dataset, across the different phases of our extension, is shown in Figure 2.

6. Discussion

Due of time constraints, our experiments were limited to only 2 iterations of the SSL procedure, therefore additional increases in performance may be possible with further training.

Moreover, being only recruits in this field, we likely have proceeded in more than one unorthodox way. We reckon

IDDA → CamVid												
	Bicyclist	Building	Car	Pole	Fence	Pedestrian	Road	Sidewalk	SignSymbol	Sky	Tree	mIoU
M_0 (Adv)	0.1571	0.5782	0.6439	0.2153	0.0073	0.2971	0.7385	0.2704	0.0	0.7945	0.4517	0.3776
M_1 (SSL)	0.1615	0.6269	0.6748	0.2233	**0.0259**	0.3938	**0.7390**	0.3283	0.0	0.8395	0.4892	0.4093
M_2 (SSL)	**0.1787**	**0.6602**	**0.7288**	**0.2373**	0.0144	**0.4165**	0.7385	**0.3428**	0.0	**0.8763**	**0.5194**	**0.4284**

Table 7. Performance of the trained models at the extension phase

a more thorough and systematic exploration and hyper-parameter tuning might give even better results: for example and again due to lack of time, we did not perform any tuning of the SSL confidence threshold nor of the pseudo-labels segmentation loss multiplier.

Having said so, we hope for there to be value in what we have discovered and hope it might give some inspirations for future research.

7. Conclusions

With our work, we explored the problems and solutions of domain adaptation for semantic segmentation. We did so by setting a high bar by training in a supervised manner over the later target domain. We then shifted our focus on adversarial unsupervised domain adaptation in the output space, and later to studying the effect that Self-Supervised Learning can have on the such learning paradigm. In particular, we proved that even a very simple method such as producing pseudo-labels from an adversarial-trained model, and training such a model with its pseudo-labels leads to a significant improvement in the model's generalization capabilities, successfully narrowing the gap between source and target.

References

[1] Emanuele Alberti, Antonio Tavera, Carlo Masone, and Barbara Caputo. IDDA: a large-scale multi-domain dataset for autonomous driving. *CoRR*, abs/2004.08298, 2020.

[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. Video-based Object and Event Analysis.

[3] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019.

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

[5] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional learning for domain adaptation of semantic segmentation. *CoRR*, abs/1904.10620, 2019.

[6] 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.

[7] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. 2018.

[8] 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.