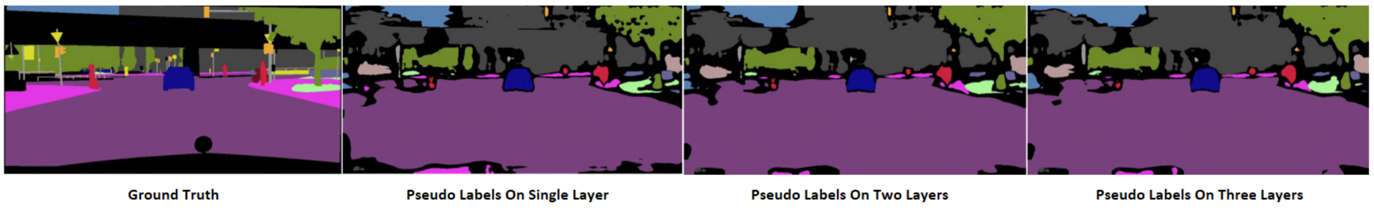
Multiple Layer Self Supervised Learning for Real-time Semantic Segmentation

Davide Bartoletti
Politecnico di Torino
Student ID: s296091
s296091@studenti.polito.it

Domenico Bulfamante
Politecnico di Torino
Student ID: s301780
s301780@studenti.polito.it

Giovanni Sciortino
Politecnico di Torino
Student ID: s302959
s302959@studenti.polito.it


Ground Truth
Pseudo Labels On Single Layer
Pseudo Labels On Two Layers
Pseudo Labels On Three Layers

Preview figure. Comparison between pseudo labels generated and the ground truth

Abstract - Convolutional neural network-based approaches for semantic segmentation rely on supervision with pixel-level ground truth, but may not generalize well to unseen image domains. As the labeling process is tedious and labor-intensive, developing algorithms that can adapt source ground truth labels to the target domain is a field of great interest. In this paper, we propose an adversarial learning method for domain adaptation in the context of semantic segmentation, improved through a self-supervised learning based on the creation of pseudo labels on the output of our architecture. To further enhance the adapted model, we deploy an adversarial network based on pseudo label creation at different feature levels, considering two and three layers.

I. Introduction

Semantic segmentation has the aim of providing a label for every single pixel of an image according to specific categories. There are many applications of this algorithm such as robotics, scene understanding and autonomous driving. The main problems of semantic segmentation are two. The first is the big amount of parameters used by the architecture that causes long training and inference time. The second is the struggle to obtain large quantities of labeled data, making infeasible and costly the application of segmentation in real-world. Moreover, the implementation of a real-time problem is crucial and fundamental. To cope with limited labeled training data, many have attempted to directly apply models trained on a large-scale labeled source domain to a sparsely labeled or unlabeled target domain. Unfortunately, direct transfer across domains often performs poorly due to the presence of domain shift or dataset bias. For this reason recent works have focused their attention on the usage of a synthetic dataset with computer-generated annotations, so that the labeling process is automatized, and fine annotated labels on real images are not needed anymore. The main problem is that the trained model might not generalize well to unseen images, especially when there is a domain gap between the training (source) and test (target) images.

To address this issue, knowledge transfer or domain adaptation techniques have been proposed to close the gap between source and target domains, where labels are not available. Thanks adversarial learning techniques in the feature space this problem can be solved. Some of them are used in order to reduce the number of parameters involved in the algorithm, while many methods try to boost performances introducing, for example, Self-Supervised Learning and pseudo-labelling techniques. In this paper we start implementing an unsupervised adversarial domain adaptation algorithm, showing also a light version in order to decrease the number of parameters and the total number of Floating Point Operations (FLOPS).

The self-training scheme involves iterative processing of target data; it generates target pseudo labels and retrains the network. However, since only the confident predictions are taken as pseudo labels this approaches inevitably produce sparse pseudo labels in practice. So to tackle this problem we generated pseudo labels also from different layers, combining them to reduce the sparsification. As it is shown in the Preview Figure, on the top of the paper, the pseudo labels generated with three layers, compared with the standard single layer ones, are less sparsificated.

II. Related Works

A. Semantic Segmentation

Before the arrival of deep networks, the best performing methods tried to classify the pixels of an image independently and through classifier models. The success of deep convolutional neural networks led the researchers to exploit feature learning capabilities for structured problems such as semantic segmentation. An example is covered by the application of networks designed for object categorization to the segmentation task, particularly by replicating the deepest layer features in blocks to match image dimensions. Newer architectures have advanced the state-of-art by learning to decode or map low-resolution image representations to pixel-wise predictions. Recently, lots of approaches based on FCN [1][7] have improved the performances on different benchmarks. Most of these approaches are intended to encode additional spatial information or to enlarge the receptive field; the main problem is that to reach the state-of-art results, we need a substantial amount of pixel-wise annotations. Because of this, synthetic databases such as GTA5, where the labeling process is partially automated, have been introduced to reduce the labeling cost of the architectures. While some methods use an encoder-decoder architecture, which cost less at inference time than dilated convolution methods, others address the loss of semantic information during the encoder-decoder downsampling and upsampling operations. Our baseline model is relayed on the BiseNet architecture[1],[7].

B. *Domain adaptation*

In order to solve the domain shift problem, domain adaptation methods have been developed. To reduce the gap between the semantic segmentation network and the target domain, several strategies have been proposed. Most of them can be grouped into adversarial training and self-training methods.
In adversarial training, the domain discriminator is trained to provide supervision to align the domain distributions based on style-transferred inputs or network features/outputs; while the generator has the role of fooling the discriminator. To further explore UDA (Unsupervised Domain Adaptation) and enhance the model, Tsai et al. [5] constructed a multi-level adversarial network to perform domain adaptation at different levels.
In self-training, the network is adapted to the target domain by employing high-confidence pseudo-labels. To regularize the training and avoid pseudo-label drift, approaches such as confidence threshold or pseudo-label prototypes have been used.

III. Methods

The main goal of our research is to develop an architecture to reduce the domain shift between our synthetic (source) and real-world (target) datasets. To do so, we start by defining our baseline performance for the domain adaptation phase by creating a model based on BiseNet's architecture that was directly trained on a portion of Cityscape's images and tested on the other. The pixel accuracy and MioU that resulted were used to set the upper bound for the DA phase. Following the establishment of our baseline, we apply an unsupervised adversarial DA technique to our synthetic-to-real context (where GTA was used as the source dataset and Cityscape was used as the target). In the following sections, we present a theoretical perspective on our work.

A. *Unsupervised Adversarial Domain Adaptation*

The algorithm proposed to cope with the Unsupervised Domain Adaptation task is composed by two modules, a segmentation network G and a discriminator D. The first one is used to predict the output result, while the second one to distinguish whether the input is from the source or target domain.

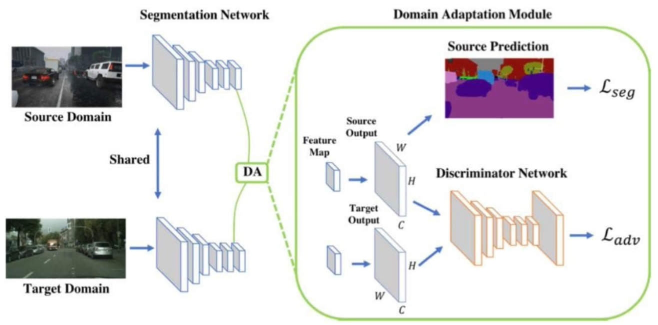


Figure 1. *Unsupervised adversarial domain adaptation algorithmic overview. The image was taken from [5]. The segmentation network process source and target images of size H x W to generate output predictions. Based on the source ground truth, a segmentation loss is generated for source predictions. The discriminator receives both source and target predictions and attempts to identify whether the input is from the source or target domain. On the target prediction, an adversarial loss is computed and back-propagated to the segmentation network.*

A source image with its labels is passed to the segmentation network in order to optimize G. The softmax P_t segmentation output for the target picture I_t (without label) is then anticipated. The fundamental goal of this DA architecture is to get source predictions and target images as close to each other as possible. To accomplish this, the discriminator D gets two predictions and attempts to determine if the input is from the source or target domain. The network propagates gradients from D to G with an adversarial loss on the target prediction, driving G to generate similar segmentation distributions in the target domain to the source prediction. Figure 2 illustrates a high-level overview of the proposed algorithm.

The joint loss for the adaptation task is defined as:

$$\mathcal{L}\left(I_s, I_t, Y_s\right) = \mathcal{L}_{seg}\left(I_s, Y_s\right) + \lambda_{adv}\mathcal{L}_{adv}\left(I_t\right) \qquad (1)$$

where $\mathcal{L}_{seg}$ is the cross-entropy loss using source ground truth labels, $\mathcal{L}_{adv}$ is the adversarial loss and λ_{adv} is the weight used to balance the two losses. In particular, $\mathcal{L}_{seg}$ is defined as:

$$\mathcal{L}_{seg}\left(I_s\right) = -\sum_{h,w}\sum_{c\in C} Y_s^{(h,w,c)} \log\left(P_s^{(h,w,c)}\right) \qquad (2)$$

where Y_s is the ground truth annotations for source images, $P_s = G\left(I_s\right)$ is the segmentation output, C is the number of classes and h, w refer to a specific pixel of the image with dimensions HxW.

Then, the images in the target domain are forwarded to G to obtain the prediction $P_t = G(I_t)$. The adversarial loss $\mathcal{L}_{adv}$ in (2) is defined as:

$$\mathcal{L}_{adv}(I_t) = -\sum_{h,w} \log\left(D(P_t)^{(h,w,1)}\right) \tag{3}$$

This loss is designed to train the segmentation network and mislead the discriminator by increasing the probability of the target prediction being treated as the source prediction, bringing the P_t distribution closer to P_s.

The discriminator is trained by giving it as input the segmentation softmax output $P = G(I)$ and using a cross-entropy loss $\mathcal{L}_d$ for the two classes (i.e., source and target). This loss can be written as:

$$\mathcal{L}_d(P) = -\sum_{h,w} (1-z)\log\left(D(P)^{(h,w,0)}\right) + z\log\left(D(P)^{(h,w,1)}\right). \tag{4}$$

where $z = 0$ if the sample is obtained from the target domain, and $z = 1$ for samples taken from the source domain.

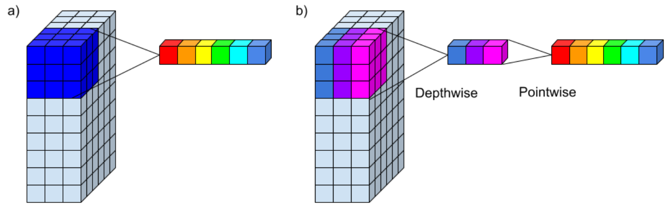


Figure 2. *A normal convolution and a depthwise separable convolution are compared. a) Standard convolution with a 3x3 kernel and three input channels. The projection of one value from the 3x3x3 (dark blue) input values to six colorful output channels is illustrated. b) Depthwise separable convolution with 3 input channels and a 3x3 kernel. First, a depthwise convolution converts each input channel's 3x3 pixels to one corresponding output pixel (matching colors). Then, using these three output pixels, a pointwise convolution determines the six final output pixels.[8]*

B. Lightweight depthwise-separable convolutions

The discriminator implemented initially in our framework is a common Fully Convolutional Discriminator (FCD) with 5 convolution layers performing 2D convolutions. A lightweight variant is obtained replacing each convolutional operational with a depthwise-separable convolution.

This kind of convolutions are designed to separate spatial (image height and width) interactions from channel interactions (e.g. colors). Indeed, a standard convolution brings it all together by multiplying values over several spatial pixels as well as throughout all channels. In depthwise separable convolutions, to separate them, we first use a convolution that is merely spatial and independent of channel; this implies that each channel has its own convolution. Following that, from the spatial interaction previously mapped, we use an other convolution to model the channels ones. This second operation is sometimes referred to as a pointwise convolution since it employs a 1x1 kernel, which represents no spatial interactions but only works on single 'points'. Since the DSC approach leads to a reduction of parameters and Floating point operation, as it's shown in section *IV*, its usage is suggested for models that need to be run on memory constrained devices.

C. SSL

We adopt a self-supervised learning technique generating pseudo labels of the target domain to help the UDA algorithm achieving better results. This Self-Supervised Learning (SSL) approach [6] can be divided in two steps:

- First, pseudo labels are not generated and the training is performed until the segmentation model performs good enough to produce confident annotations for the target domain;
- Then, when the segmentation model is sufficiently accurate, starting from its output predictions, we obtain the pseudo labels $\widehat{Y}_t$ for the target domain that can be used to modify the joint loss function of the adaptation task as:

$$\mathcal{L}\left(I_s, I_t, Y_s, \widehat{Y}_t\right) = \mathcal{L}_{seg}(I_s, Y_s) + \lambda_{adv}\mathcal{L}_{adv}(I_t) + \mathcal{L}_{seg}\left(I_t, \widehat{Y}_t\right) \tag{5}$$

In order to produce pseudo labels we choose the "max probability threshold" (MPT) [6] to filter the pixels with high prediction confidence in I_T. Thus we can define the mask map $\hat{y}_T$ as $m_T = \mathbb{1}_{[\operatorname{argmax} G(I_T) > \text{ threshold }]}$. Through this threshold function, classes that do not have a probability higher than the right level are classified as background (label : 255). A graphical representation of this method is reported in Fig.3.

D. SSL multilevel

One of the main problem of the previous approach is that, setting an high value of the threshold function, many pixels are categorized as background, causing a sparsification of the preudo labels. So inspired from the context path of Bisenet architecture, we propose a multi-layers method to get more information from the model outputs. From the prediction of the multiple-layers (we considered two and three layers), after an upsampling step to match the dimension of the image, we perform a majority voting operation for each pixel at each output according to their confidence. Successively these outputs are concatenated and the previous procedure is repeated with the purpose of creating the final pseudo labels. The loss in this case is calculated in the same way as (5), but considering the final output defined before. This process is shown in Figure 4.

IV. Experiments

A. Datasets

We test our model on two subset of Cityscape and GTA5 datasets. The subset of GTA5 [4] is made up of 500 annotated

Table I
BISENET EVALUATION RESULTS ON CITYSCAPES

Backbone	road	sidewalk	building	wall	fence	pole	light	sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mIoU(%)
ResNet-101	97.01	76.12	88.77	47.46	42.85	42.97	49.01	61.59	88.97	55.45	91.28	66.16	43.82	90.27	40.21	36.57	33.95	38.16	61.86	60.7
ResNet-18	96.05	70.99	86.40	29.81	38.04	38.16	37.44	53.49	87.35	52.45	89.72	59.91	32.08	87.47	23.15	46.08	42.5	28.49	54.75	55.5

Table II
MODEL VARIANTS RESULTS

Experiments	road	sidewalk	building	wall	fence	pole	light	sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mIoU(%)
Target Only	97.01	76.12	88.77	47.46	42.85	42.97	49.01	61.59	88.97	55.45	91.28	66.16	43.82	90.27	40.21	36.57	33.95	38.16	61.86	60.7
FCD	76.28	23.13	76.77	28.22	10.85	21.09	19.04	5.78	80.75	27.37	70.53	40.95	10.38	66.58	13.11	7.70	0.	6.47	0.	30.8
FCD-Light	81.94	33.81	75.76	22.63	10.98	21.06	11.54	6.20	78.55	21.75	70.53	36.95	7.11	66.54	14.95	8.72	0.	8.52	0.	30.4
FCD-SSL	78.92	23.06	78.59	23.21	14.21	21.90	16.20	6.15	81.73	23.84	72.3	45.41	13.3	72.96	12.48	6.52	0.	7.01	0.	31.5
FCD-SSL-2	79.94	21.85	78.74	22.37	11.40	20.60	16.49	6.09	81.79	25.69	74.82	43.33	10.87	72.12	12.63	4.38	0.	6.42	0.	31.0
FCD-SSL-3	80.62	24.26	78.55	21.33	12.74	21.07	15.80	5.68	81.96	25.66	74.27	43.85	11.11	73.98	12.73	3.17	0.14	6.91	0.01	31.3

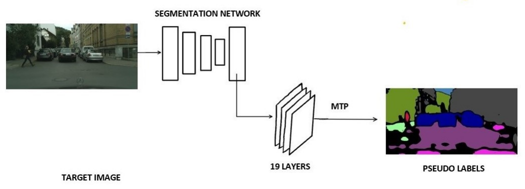


Figure 3. *Pseudo labelling creation on a single layer.*

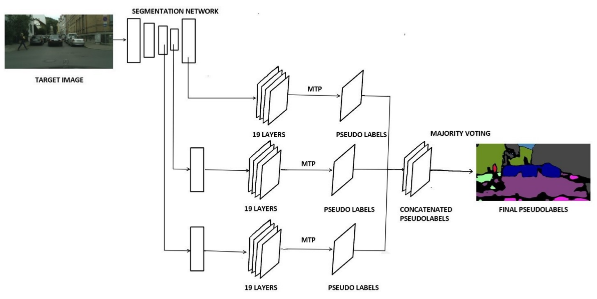


Figure 4. *Process of pseudo labelling creation on multiple layers.*

photos from the aforementioned video-game. The standard 19-classes, which Cityscapes shares, is used for training and evaluation. The subset of Cityscapes [3] is made up of real-world pictures gathered from various German cities. It's composed by 500 images for training and 250 for both validation and test set.

B. Implementation details

Our baseline model is BiSeNet, initialized with a ResNet-101 pretrained on ImageNet. The first step of the task is performed using a Fully Convolutional Discriminator (FCD) with 5 convolutional layers with kernel size 4x4, channel numbers 64, 128, 256, 512, 1, stride 2 and padding 1. For all the following experiments, a lightweight discriminator is implemented by replacing each 2D convolution with a depthwise convolution of kernel size 4x4, each followed by a pointwise convolution, with kernel size 1x1. The channel numbers, stride and padding remain unaltered. Each convolutional layer is followed by a Leaky ReLU with a negative slope of 0.2.

The general setting for our experiment will be based on a number of epochs equal to 100 and a Stochastic Gradient Descent (SGD) with batch size 4, momentum 0.9, weight decay $1 * 10^{-}4$ and initial learning rate $2.5 * 10^{-}2$ that progressively decreases according to the "poly" learning rate strategy We use to crop images according to the dimension (1024,512), normalizing them with mean: (73.15, 82.90, 72.39).

Table III
COMPLEXITY MEASURES OF FCD AND FCD-LIGHT

	FCD	FCD-Light
Parameters	2.781M	0.189M
FLOPS	30.953G	2.178G
Training Time	6.23h	5.21h

C. Results

First of all we start training the BiSeNet model using ResNet-101 and ResNet-18 as backbone. Training on the more complex model, ResNet-101, produces the best results and for this reason our work will focus on it for all the following points. On the Table I we report the evaluation results of the two different ResNet versions on Cityscapes database.

Table II shows the results on the GTA $\rightarrow$ Cityscapes domain adaptation task. Analysing the values, as expected, we can see that the UDA get performances that are approximately half of what was achieved with the model based only on the target. When each convolution in this discriminator is replaced with its lightweight counterpart (FCD-Light), we get lower values of mIOU, but the model is less computational expensive in terms of parameters and number of FLOPS (table III). The labels are generated from a segmentation model trained on 100 epochs using one or more layers. Since the FCD-light has lower performances in term of mIOU than the traditional discriminator, so we will take into account only the first version of the FCD. As in Table II, evaluating the results of the several variants of SSL methods, we can notice an appreciable increase of mIOU in the single layer case reaching 31.5%. Moreover taking in account the single IoU of each class there is a slight increase of the evaluation measurement in some classes when more layers are taken in consideration. Indeed, for the three layers implementation we have approximately the same results in terms of mIoU, but we can notice, from the result table (Table II), that for the first time a significant IoU value is reached for small classes as bicycle and train. This fact is due to the less sparsification of the labels resulting in a better detection of smaller details.

V. Conclusion

In this work we implemented an architecture able to perform Domain Adaptation in Semantic Segmentation; the model, in general, achieves adequate and comparable results with respect to similar implementation. Moreover, a lighter framework is presented having the discriminator with Depthwise-Separable convolutions. The accuracy of the DSC model is typically lower due to the simplifications. However in our case, the trade-off in accuracy is so small that the computational benefits make its employment effective. Finally we evaluate an extension of the task generating pseudo labels from single and multiple layers. The result shows that there can be still work to be done in this direction evaluating and improving the multi layer approach. One example could be the extraction of objects' borders and their usage on SSL techniques.

References

[1] "BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation" Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, Nong Sang.

[2] "A Review of Single-Source Deep Unsupervised Visual Domain Adaptation", Sicheng Zhao, Xiangyu Yue, Shanghang Zhang, Bo Li, Han Zhao, Bichen Wu, Ravi Krishna, Joseph E. Gonzalez, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia, Kurt Keutzer.

[3] "The Cityscapes Dataset for Semantic Urban Scene Understanding", M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele.

[4] "Playing for Data: Ground Truth from Computer Games", Stephan Richter, Vibhav Vineet, Stefan Roth, Vladlen Koltun.

[5] "Learning to Adapt Structured Output Space for Semantic Segmentation", Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, Manmohan Chandraker.

[6] "Bidirectional Learning for Domain Adaptation of Semantic Segmentation, Yunsheng Li, Lu Yuan, Lu Yuan.

[7] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation, 2020.

[8] https://www.paepper.com/blog/posts/depthwise-separable-convolutions-in-pytorch/