Real-time Domain Adaptation in Semantic Segmentation

Alessia Intini[*] Antonio Iorio[*] Giuseppe Scarso[*]

Abstract

This paper explores domain adaptation techniques in semantic segmentation using two datasets, GTA5 and CityScapes. Semantic segmentation is used to assign a label to each pixel in an image. Domain adaptation, on the other hand, is a machine learning paradigm that aims to learn a model from a source domain that can perform well on a different target domain. Using the BiSeNet, the goal was to improve the performance of domain adaptation by comparing different methods and searching for their best possible configuration. The methods used were: Data Augmentation, Adversarial Domain Adaptation and Fourier Domain Adaptation. Performance comparisons of the different methods were performed using mIoU and precision per pixel as metrics.

1. Introduction

In recent years, the field of machine learning has made significant progress, particularly in the area of neural networks, increasing the demand for real-time performance in various applications. This project aims to address the challenge of achieving real-time performance of neural networks by exploiting synthetic datasets and employing domain adaptation techniques.

Traditional approaches to training neural networks often rely on large-scale real datasets, which can take a long time to collect and annotate. To overcome these limitations, it suggests a new paradigm that exploits synthetic dataset, generated through advanced simulation techniques, as a means to speed up the training process.

The project begins by recognizing the difficulty of obtaining good performance on real datasets due to the scarcity of data. Next, it is shown that, in contrast, training on synthetic dataset leads to significantly better results. Finally, various methods are tested for effectively incorporating synthetic dataset during training and adapting the model to maintain accuracy on real dataset. Applying, at first, Adversarial Domain Adaptation and then Fourier Domain Adaptation to improve performance.

[*]Polytechnic Of Turin University

2. Related work

2.1. Semantic segmentation

Semantic segmentation assigns a category label to each pixel of an image, which is a fundamental but challenging task in computer vision research. As semantic segmentation is able to provide the category information at the pixel level, the pixel-level semantic information helps intelligent systems to grasp spatial positions or make important judgments. The goal of semantic segmentation is to divide an image into mutually exclusive subsets, with each subset representing a significant region of the original image. Image classification aims to assign one or more category labels for a whole image, in fact through an image classification algorithm it is possible to know which objects exist in an image. When using this classification, it is not possible to make a distinction that takes into account the different instances of the objects within the same class. For example, all the people in the image will be assigned to the same ”person” class, without distinguishing them as separate instances [7].

2.1.1 Semantic segmentation methods

Semantic segmentation classification by supervision level refers to the categorization of techniques based on the amount and type of supervision required during the training process.

Supervised methods: In the case of the supervised semantic segmentation methods, there is an assumption that sufficient labeled training data are available, including the original images and their corresponding pixelwise semantic annotated images.

Weakly-supervised methods: In fully supervised methods, pixel-level annotations are indispensable for training networks, but this process is very labor-intensive. Therefore, it is useful to conduct research on weakly supervised semantic segmentation methods.The main goal of this method is to achieve satisfactory segmentation accuracy only with weakly supervised annotations; these will involve much less manual labeling effort. The purpose of these methods will be to take full advantage of the limited supervision to generate pixelwise masks acceptable for training.

Semi-supervised methods: Different from the weakly-supervised situation, the semi-supervised semantic segmentation assumes that there is only a small number of fully annotated training images, instead of a large number of training images with weak supervision.

2.1.2 Real-time semantic segmentation

Semantic segmentation is a fundamental task in computer vision, in these fields usually high speed efficient inference is required for fast interaction or response. To speed up the models of real-time semantic segmentation algorithms, there are mainly three approaches: [1]
Cropping: Attempt to limit the size of the input to reduce computational complexity through cropping or scaling. Although the method is simple and effective, the loss of spatial detail corrupts the prediction, especially near boundaries, leading to a decrease in accuracy;
Prune the channels: Another solution is to prune the channels of the network to increase the speed of inference especially in the early stages of the basic model. However, this weakens the spatial capacity.
ENet: For the last case, ENet [6] proposes to leave the last stage of the model to obtain an extremely narrow structure. However, the disadvantage is that you abandon resampling operations in the last stage this leads to poor discriminative ability.
Overall, all the above methods compromise accuracy in favor of speed. In conclusion based on the above observations, it proposes to use the bilateral segmentation network (BiSeNet) with two parts: Spatial Pathway (SP) and Contextual Pathway (CP). These two components are designed to cope with the loss of spatial information and the contraction of the receptive field.

2.1.3 Evaluation metrics

Another important aspect are the evaluation metrics from the perspective of accuracy and efficiency.
Metrics for accuracy We assume that the category number is k + 1 in total, including k classes and one background, while Y is denoted as the pixel number. For instance, Yij means there are Yij pixels, which are predicted as the jth class, but actually belong to the ith class, i.e. Y ii , Y ij , and Y ji denote the number of True Positive (TP), False Positive (FP), and False Negative (FN), respectively. Now we review three metrics for evaluating semantic segmentation accuracy:

- **Pixel Accuracy (PA)**

  PA denotes the ratio between the number of correctly classified pixels and the total number. The PA can be obtained by Eq.1

$$PA = \frac{\sum_{i=0}^{k} Yii}{\sum_{i=0}^{k}\sum_{j=0}^{k} Yij} \tag{1}$$

- **Intersection over Union (IoU)**

  IoU means the rate between the intersection and union. It can be computed based on Eq.3

$$IoU = \frac{\sum_{i=0}^{k} Yii}{\sum_{i=0}^{k}\sum_{j=0}^{k} Yij + \sum_{i=0}^{k}\sum_{j=0}^{k} Yji - \sum_{i=0}^{k} Yii} \tag{2}$$

- **Mean Intersection over Union (MIoU)**

  MIoU denotes the aver- age of the pre-class IoU and it can computed based on Eq.4

$$MIoU = \frac{1}{k+1}\sum_{i=0}^{k} \frac{Yii}{\sum_{j=0}^{k} Yij + \sum_{j=0}^{k} Yji - Yii} \tag{3}$$

2.1.4 Loss functions

Loss function makes it possible to assess the error between the value predicted by the model and the actual value. The objective of the training phase is to identify those parameters that minimise the loss function, so that the predicted value is closer to the actual value with the consequence that the model becomes more accurate in making predictions. There are different methods to evaluate the loss function depending on the context. Typically, in the case of regression problems, mean squared error is used, whereas for classification problems, cross entropy could be used.

- **Cross Entropy Loss**

  This loss function measures the average distance between the predicted probability distribution ($\hat{y}$) and the true distribution (y). It penalizes predictions further away from the truth.

- **BCE With Logits Loss**

  This function is specifically designed for binary classification tasks. It compares the model's "logits" (raw, unnormalized outputs) with the ground truth. In this case it is used to calculate the loss for the discriminator in the Adversarial Domain Adaptation training.

- **Entropy Minimization Loss**

  This function is used to penalize high entropy prediction more than low entropy ones, used for the Fourier Domain Adaptation.

2.2. Domain adaptation

Domain adaptation (DA) is a machine learning technique used to transfer knowledge from a source domain to a target domain, where the source and target domains may have different data distributions or feature representations. So it is a specific form of transfer learning, where knowledge gained in one domain is transferred to another. In DA, there are typically two domains involved—the source domain and the target domain.

- The source domain is the domain from which labeled data is available for training a model;
- The target domain is the domain for which the model's performance needs to be improved, but labeled data is scarce or unavailable.

Furthermore there are two primary approaches to domain adaptation:

- Unsupervised: No labeled data from the target domain is used during training.
- Semi-supervised: A small amount of labeled data from the target domain may be available and can be used along with the source domain data.

In domain adaptation for semantic segmentation, several approaches, such as *Data Augmentation*, *Adversarial Domain Adaptation* and *Fourier Domain Adaptation*, are employed to address the domain discrepancy problem and improve model performance on target data.

2.2.1 Data augmentation

Data augmentation techniques artificially generate different versions of a real data set to increase its size. It is easy to verify that with the use of data augmentation, the accuracy of machine learning models can increase. Data augmentation techniques are designed to create new artificial data samples from those already in the training dataset. This is done by applying a series of transformations or manipulations to existing data while keeping the class label intact (when supervised data is involved). This process makes it possible to introduce variations and perturbations into the data using the samples already in one's possession [2].
The different techniques used are:

- **Noise:** Adding noise on the image;
- **Cropping:** A section of the image is selected, cropped and then resized to the original;
- **Flipping:** The image is flipped horizontally and vertically;
- **Rotation:** Rotates the image by one degree between 0 and 360;
- **Scaling:** Resizes the image, making it smaller or larger than the original;
- **Translation:** Shifts the image into various area along the x-axis or y-axis,so neural network looks everywhere in the image to capture it;
- **Brightness:** The brightness of the image is changed and new image will be darker or lighter;
- **Contrast:**The contrast of the image is changed and new image will be different from luminance and colour aspects.

2.2.2 Adversarial Domain Adaptation

Adversarial Domain Adaptation is based on the idea of learning a representation of the data that is invariant with respect to the distribution of the domain, in this context a **generator** and a **discriminator** are used; the former is in charge of generating synthetic data that must appear realistic and indistinguishable from real data, while the latter is in charge of evaluating the samples and assigning them a probability that is true or false. Based on the result of the discriminator, the two components are updated so that the generator produces samples that can be identified as true by the discriminator, while the discriminator is updated to distinguish images more accurately. Consequently, a balance is needed between the loss of the generator and the loss of the discriminator.

2.2.3 Fourier Domain Adaptation

Instead, in this other technique it describes a simple method for unsupervised domain adaptation, in which the discrepancy between the source and target distributions is reduced by exchanging the low-frequency spectrum of one for the

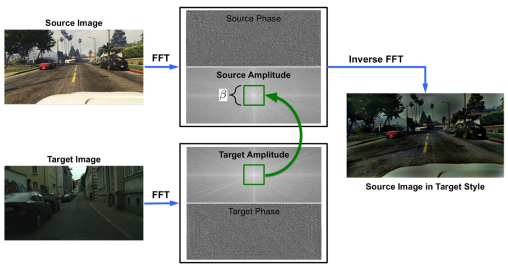


Figure 1. Mapping a source image to a target "style" without altering the semantic content. A randomly sampled target image provides the style by exchanging the low-frequency component of the source image's spectrum for its own. The result shows a smaller domain gap at the perceptual level and improves learning.

other. This method is applied to semantic segmentation, where densely annotated images are abundant in one domain (e.g., synthetic data) but difficult to obtain in another (e.g., real images).This method may also be simpler than others usually applied in fact it requires only the application of a simple Fourier transform and its inversion. Despite its simplicity it can achieve good performance when integrated into a relatively standard semantic segmentation model [9]. The method consist in simply compute the (Fast) Fourier Transform (FFT) of each input image, and replace the low-level frequencies of the target images into the source images before reconstituting the image for training, via the inverse FFT (iFFT), using the original annotations in the source domain Figure 1.

3. Methodology

3.1. Dataset

Two datasets Cityscapes and GTA5 were used to perform real-time domain adaptation experiments in semantic segmentation.

3.1.1 Cityscapes

Cityscapes is a benchmark suite and large-scale dataset to train and test approaches for pixel-level and instance semantic labeling. It contains several hundreds of thousands of frames, with a resolution (1024,2048), were acquired from a moving vehicle during the span of several months, covering spring, summer, and fall in 50 cities, primarily in Germany but also in neighboring countries. The sensors from where the frames were taken were mounted behind the windshield and yeld high dynamic-range(HDR) images with 16 buts linear color depth, but they provide a low dynamic-range (LDR) 8 bit RGB images that are obtained by applying a logarithmic compression curve [4].
The dataset contains 8 categories and 19 class for evaluation:
Human: contains two class: person and rider;
Vehicle: contains the class:car, truck, bus, train, motorcycle, bicycle, caravan, trailer;
Nature: contains the class: vegetation, terrain;
Construction: contains the class: building, wall, fence, guard rail, bridge, tunnel;
Object: contains the class:traffic sign, traffic light, pole, pole group;
Sky: contains the class sky;
Flat: contains the class: road, sidewalk, parking, rail track;
Void: contains the class: ground, dynamic, static, ego vehicle, unlabeled, out of roi, rectification border.

3.1.2 GTA5

GTA5 is used for creating large-scale pixel-accurate ground truth data for training semantic segmentation systems. Generally, detailed semantic annotation of images is not available because the internal operation and content of the game are largely inaccessible. But this problem can be overcome by a technique known as detouring, a wrapper is inserted between the game and the operating system, allowing us to record, modify, and reproduce rendering commands. By hashing distinct rendering resources – such as geometry, textures, and shaders – communicated by the game to the graphics hardware, this generate object signatures that persist across scenes and across gameplay sessions. This allows us to create pixel-accurate object labels, speeding up the labelling process. In this way, 25 thousand images to be extracted from the game, with a resolution (1052, 1914) and the entire labelling process took only 49 hours. RenderDoc was used to collect the data into a format that is suitable for annotation. GTA5 applies post rendering effects, such effects in dataset creation are not applied [8].
In order to generate a labelled image, the following steps are applied:

1. you start with an image where groups of pixels share mesh, texture and shader properties. This group is called patch;

2. You have that each object is a set of patches. Consequently, to label an object, you simply propagate the patches that make it up and assign it the label. The previous operations are applied to all the elements represented in the image.

3.2. Metrics

During the training phase, it is important to be able to monitor learning progress, so metrics were used that can express the goodness of fit of the model. When evaluating a standard machine learning model, predictions are usually classified into four categories: true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) [3]. These metrics are associated with semantic segmentation, as mentioned earlier, in the case of this experiment it was necessary to use only two: accuracy (Equation 4) and mIoU. Using the four categories mentioned above, we can express these two metrics as:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$IoU = \frac{TP}{TP + FP + FN} \tag{5}$$

The Equation 5 calculates the Intersection over Union (IoU)

for a single object or class. To calculate the Mean Intersection over Union (mIoU) over a set of predictions, it takes the average of the IoUs calculated for each class or object.

3.3. Architecture

3.3.1 BiSeNet

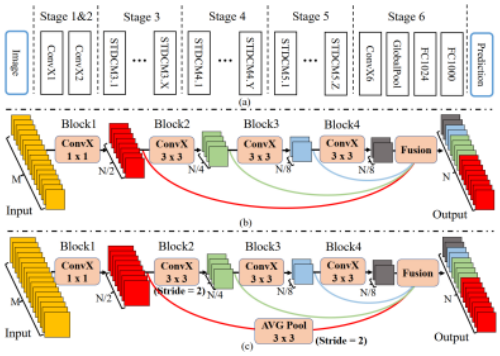


Figure 2. General STDC network architecture. ConvX operation refers to the Conv-BN-ReLU. (b) Short-Term Dense Concatenate module (STDC module) used in our network. M denotes the dimension of input channels, N denotes the dimension of output channels. Each block is a ConvX operation with different kernel size. (c) STDC module with stride=2.

BiSeNet, short for Bilateral Segmentation Network, is an architecture designed for semantic segmentation tasks in computer vision. It was proposed to achieve a balance between accuracy and efficiency in real-time segmentation applications. The architecture consists of a spatial path and a context path.

The spatial path captures fine-grained details through a lightweight convolutional network, while the context path captures global contextual information using a larger receptive field. These paths are then fused using a bilateral fusion module to combine local and global features effectively.

BiSeNet is known for its ability to achieve high segmentation accuracy with reduced computational complexity, making it suitable for real-time applications.

3.3.2 STDC

Short-Term Dense Convolution (STDC) is the heart of the module. As it can see in the Figure 2, it applies multiple 3x3 convolutions, then concatenates their outputs. This captures diverse contextual information at different scales while maintaining lightweight computation [5].

3.4. Used Model

Model uses the pretrained on imageNet STDC networks as the backbone of the encoder and adopt the context path of BiSeNet to encode the context information Figure 3.

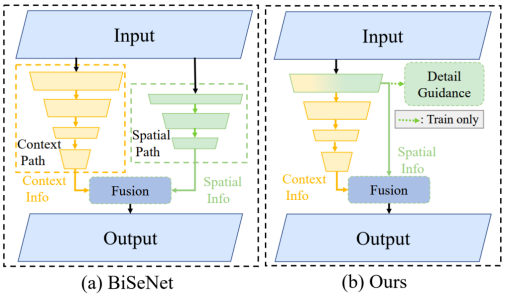


Figure 3. (a) presents Bilateral Segmentation Network (BiSeNet), which use an extra Spatial Path to encode spatial information. (b) demonstrates our proposed method, which use a Detail Guidance module to encode spatial information in the low level features without an extra time-cosuming path.

4. Experiments

4.1. Starting point

In the experiment, it started from the BiseNet neural network and pre-trained backbone with STDC. Network is always trained for 50 epochs with a learning rate of 0.01. Different combinations of batch sizes and optimizers were made during the experiments performed, in particular batch size was made to change between 4, 6 and 8, while Adam and SGD were considered as optimizer. The results of all experiments performed were evaluated with the metrics mentioned above.
During the training phases, the various datasets were not used at the original resolution, but they were reduced to speed up the above-mentioned phase. The training cases are on:

- Cityscapes: resolution (512, 1024);
- GTA5: resolution (720, 1280);
- Both datasets used: in this case the images have been adapted to GTA5's resolution.

4.1.1 Train on Cityscapes

As a first step, network was trained on Cityscapes, experimenting with different batch sizes and optimizers. The evaluation was based on Pixel Accuracy and Mean Intersection over Union(mIoU) Table 1. To perform the training a lower resolution of the images was used, instead the original resolution was used for validation. It is possible to observe that the best results with Adam and SGD optimizer are obtained, for the first with a batch size equal to 8 instead for the second with value 4 and a possible outcome is in Figure 4. In general, optimizer SGD performs better than Adam.

Training on **Cityscapes** *with Resize (512,1024)*			
Optimizer	**Batch-size**	**Accuracy** Metric [%]	**mIoU** Metric [%]
Adam	8	73.3	38.5
	6	69.2	34.6
	4	69.9	35.9
SGD	8	77.5	48.0
	6	77.0	47.9
	4	**77.7**	**49.1**

Table 1. Table showing the results of training on Cityscapes using different batch sizes and optimizers.

4.1.2 Train on GTA5

Subsequently, the same network was trained with the same initial configurations on the GTA5 dataset. Not having two subsets available; one for training and one for validation, then it was decided to split the dataset in the following ways: 75% of the images for training and 25% for validation. Accordingly, the results obtained are shown in Table 2. Here the results are better, so the idea is to use this synthetic dataset in order to train the model to be usable also with real world images. This also brings with it another advantage, that the creation of synthetic datasets is easier since pixel-level labels can be done with an automated process as previously explained, instead of being done totally manually. A good outcome with the best configuration can be seen in Figure 4.

Training on **GTA5** *with Resize (720,1280)*			
Optimizer	**Batch-size**	**Accuracy** Metric [%]	**mIoU** Metric [%]
Adam	8	80.6	62.6
	6	79.4	55.5
	4	79.8	61.3
SGD	8	**81.1**	**67.0**
	6	80.7	62.7
	4	80.4	61.3

Table 2. Table showing the results of training on GTA5 using different batch sizes and optimizers.

4.1.3 Domain shift

- **Training on GTA and validation on Cityscapes**
  At this stage, it was necessary to implement domain adaptation, i.e. to test the network trained with GTA5 on cityscapes. Having used the network already trained on GTA5 again, the dataset was divided by considering 75% of it for training and 25% for validation. As was to be expected, the results of this phase were not the best, in fact performance degraded particularly as shown in Table 3. It can see a result in Figure 4. For this reason, it is necessary to implement techniques that can improve this performance.

GTA5 → Cityscapes			
Optimizer	**Batch-size**	**Accuracy** Metric [%]	**mIoU** Metric [%]
Adam	8	39.6	13.2
	6	63.1	18.5
	4	24.5	10.4
SGD	8	50.9	20.1
	6	**60.6**	**21.6**
	4	57.7	20.8

Table 3. Table showing the results of training on GTA5 and validadion on Cityscapes using different batch sizes and optimizers.

After running the previous experiments, it was concluded that the best optimizer to be used for the next steps was SGD, so from this point on it was assumed that this would always be used. In addition, it was not possible to find a batch size that was globally the best, since both the use of large and small batch sizes can bring advantages and disadvantages depending on the context.

- **Training on GTA and validation on Cityscapes with data augmentation**
  As can be seen from the Table 3, when cross domain is applied, performance drops, so in order to achieve improvements, it was decided to use data augmentation techniques. This makes it possible to generate new artificial data from the training data one has, so that the model can learn from a wider variety of data and thus improve its performance. During the various experiments, several transformations had to be tried, but not all of them brought improvements in performance. The probability of data augmentation is set to 0.5, this parameter represents how much probability there is that the transformation will be applied to the image during training, its value is [0,1].

  The transformations that have been considered are:

  – **ExtRandomCrop():** This operation is used to randomly crop a portion of the image given a desidered size;

Batch-size	Accuracy Metric [%]	mIoU Metric [%]
ExtColorJitter(0.5, 0.2, 0.1, 0.1, 0.2)		
4	**55.5**	**22.2**
ExtColorJitter(0.5,0.2, 0.3, 0.3, 0.4), ExtGaussianBlur()		
4	51.3	18.6

Table 4. Table showing the results of training on GTA5 and validation on Cityscapes using batch size 4 and optimizer SGD, but using different configuration of data augmentation.

- **ExtRandomHorizontalFlip():** This operation reflects the image horizontally at random, it swaps pixels from left to right with a specified probability;
- **ExtColorJitter() :** This operation has some configurable parameters each one related to a different property of the image,
- **ExtGaussianBlur():** This operation applies a Gaussian blurring effect to the image, that is, it uses a Gaussian function.

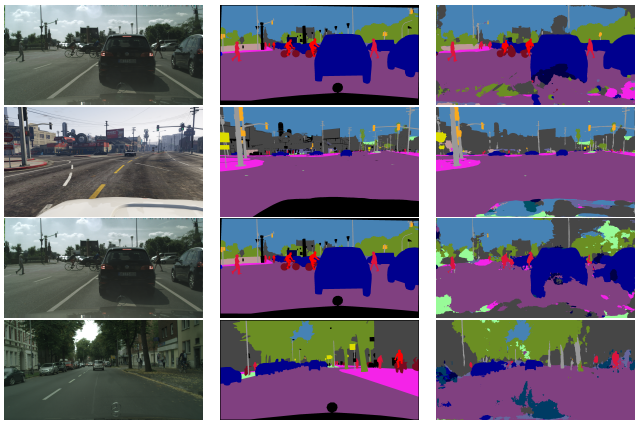
All of these transformations have been implemented, but not all of them have brought about real improvements. In fact, in a first step, all transformations were used by setting the parameters as follows:

- *ExtRandomCrop((720,1280))*,
- *ExtRandomHorizontalFlip()*,
- *ExtColorJitter(p=0.5, brightness=0.2, contrast=0.3, saturation=0.3, hue=0.4)*,
- *ExtGaussianBlur()*.

This first configuration did not lead to improvements, so it was decided to try another one, which turned out to be the best. In this final configuration, ExtGaussianBlur() was removed and the ExtColorJitter parameters were changed, setting them as follows (p=0.5, brightness=0.2, contrast=0.1, saturation=0.1, hue=0.2). In fact, in this context where colors are very important, one must be cautious in using these transformations that act precisely on them. In fact, it was necessary to use a 'lighter' transformation. The result of all these experiments can be seen in the Table 4. In addition, again, comparisons were made between the results that can be obtained by changing optimizer and batch size. The result are reported in the Table 5 and it can find an example in Figure 4

GTA5 → Cityscapes *with Data Augmentation*			
Optimizer	**Batch-size**	**Accuracy** Metric [%]	**mIoU** Metric [%]
SGD	8	58.7	21.3
	6	**62.3**	**22.5**
	4	55.5	22.2

Table 5. Table showing the results of training on GTA5 and validation on Cityscapes using different batch size and different optimizer, but using the best data augmentation configuration.

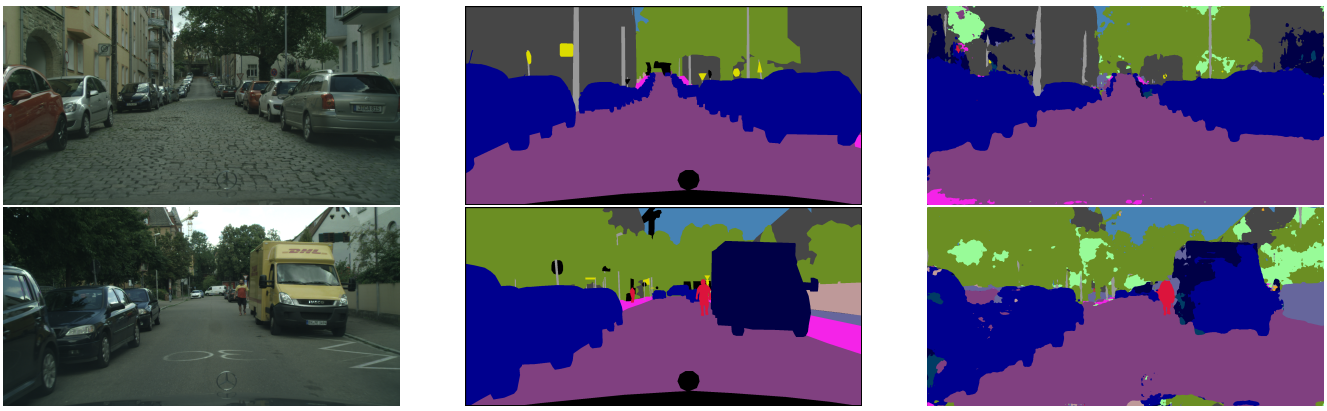


(a) Original (b) Correct label (c) Predict label

Figure 4. These images show the results of the training experiments, the first line shows the training on Cityscapes, the second line shows the result of training on GTA5, the third line shows the result of training on GTA5 and validation on Cityscapes, and the fourth line shows the result under the same conditions as the previous case but applying the data augmentation.

4.2. Unsupervised Adversarial Domain Adaptation

As reported earlier in the context of Unsupervised Adversarial Domain Adaptation, it is necessary to have a generator and a discriminator. In this context, no real generator was used, but the GTA5 dataset was used as the synthetic images. Accordingly, GTA5 was used as the source domain and Cityscapes as the target domain. After that to make target predictions closer to the source ones, the network utilizes a discriminator to distinguish whether the input is from the source or target domain (the discriminator was implemented as in the [10]). Then an adversarial loss is calculated on the target prediction and is back-propagated to the segmentation network.
In addition, two default values were set, the learning rate of the discriminator and λ_{adv}. The former is the learning rate concerning the discriminator and it is important that it is always less than the learning rate of the generator to ensure



(a) Original (b) Correct label (c) Predict label

Figure 5. These images show the results of the training experiments, the first row shows the training with Adversarial Domain Adaptation, the second row shows the result of training with Fourier Domain Adaptation where the parameters of the transformation took the best configuration explored during the experiment, which are $\eta = 2.0$, $\beta = 0.006$, and $\lambda_{\text{ent}} = 1.6e^{-6}$.

stable convergence of the adversarial training process, so it was decided to use the value $3e^{-4}$ [10]. Whereas for the second one, by doing several experiments it was concluded that the value that offered the best performance is $2e^{-3}$. The results obtained for this method are shown in Table 6 and it can find an example Figure 5.

Batch-size	**Accuracy** Metric [%]	**mIoU** Metric [%]
6	**69.8**	**30.2**
4	62.4	28.4

Table 6. Table showing the results of training on adversarial domain adaptation, the learning rate for discriminator is set as $3e^{-4}$, while $\lambda_{\text{adv}} = 2e^{-3}$.

4.3. Improvements

4.3.1 Fourier Domain Adaptation for Semantic Segmentation

To improve domain adaptation performance, it was decided to apply the fourier transform to the synthetic images, with the aim of reducing the domain gap between the two datasets. To make this possible, a function that applied the fourier transform to the source dataset had to be applied during the training phase. To this function the source and target images were provided as input, so that after calculating the two fourier transforms it was possible to substitute each input source image with the low level frequencies of the target image. At the end of this process, it was necessary to reconstruct the training image by means of an inverse FFT, using the original annotations in the source domain.
As can be seen in the Table 7, it was only possible to run the experiment for the FDA using batch size 4. Since there were limited resources available to perform the training, this caused it to be impossible to perform other experiments on larger batch sizes. An example of the result is shown in Figure 5.
The results shown were obtained by placing the parameters necessary for FDA execution at the following values: $\eta = 2.0$, $\beta = 0.006$, and $\lambda_{\text{ent}} = 1.6e^{-6}$. These values were determined by starting from the configurations in [9] and adapting them to our context. For example, several experiments were performed by varying β , but the results with values higher than the one chosen showed no improvement. On the contrary, instead, in the source image to which the Fourier transform was applied, noise was generated that reduced performance.

β	**Batch-size**	**Accuracy** Metric [%]	**mIoU** Metric [%]
0.006	4	**68.5**	**30.2**
0.01	4	67.4	28.9

Table 7. Table showing the results of training with the Fourier transform, the parameters that were used are $\eta = 2.0$ and $\lambda_{\text{ent}} = 1.6e^{-6}$, with a variation of β.

5. Conclusions

In this project, the effectiveness of real-time networks such as Bisenet with STDC as the backbone, placing significant emphasis on domain adaptation. Despite being lightweight networks, our results show that it is possible to train a model with synthetic data and adapt it to the real world.
However, it may be necessary to apply more sophisticated

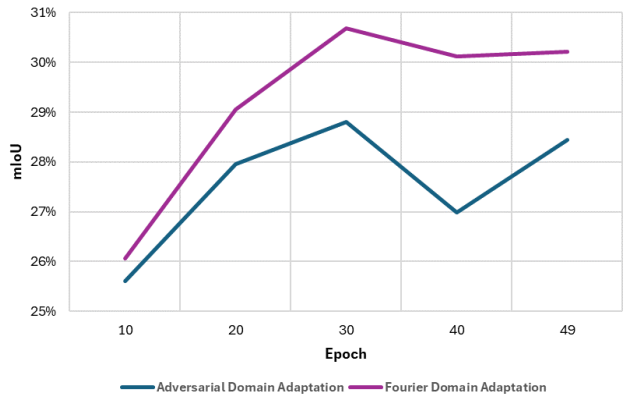


Figure 6. This graph shows the performance of mIoU by comparing the use of Avversarial Domain Adaptation and Fourier Domain Adaptation in the best configuration. The data refer to experiments performed with batch size 4 and SGD optimizer.

techniques that improve performance in the case covered by the paper domain adaptation techniques, such as Adversarial or Fourier Domain Adaptation, were used. Improvements over Adversarial Domain Adaptation were observed in the use of FDA, in fact this preprocessing technique could be an alternative to sophisticated architectures or laborious data augmentation. The experiments discussed were conducted with little parameter variation, so there may be better configurations than those used.

Therefore, further research will be needed to fine-tune and optimize the effectiveness of these techniques in more complex and heterogeneous scenarios. On going investigations in this area could lead to even more significant developments, paving the way for new applications and advances in the automatic understanding and interpretation of real-world images.

The source code of the experiment is available at the following link `https://github.com/peppe-sc/AML_project_poliTO`.

In this repository can find source code and the document containing the main results of the experiments with the name 'Document_project3_s309895_s317748_s308807'.

References

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

[2] Cem Dilmegani. Top data augmentation techniques: Ultimate guide for 2024, 2023. Available at:.

[3] Jeremy Jordan. Evaluating image segmentation models, 2018. Available at:.

[4] Sebastian Ramos Timo Rehfeld Markus Enzweiler Rodrigo Benenson Uwe Franke Stefan Roth Bernt Schiele Daimler AG RD TU Darmstadt MPI Informatics TU Dresden Marius Cordts, Mohamed Omran. The cityscapes dataset for semantic urban scene understanding. *arXiv*, 2016.

[5] Junshi Huang Xiaoming Wei Zhenhua Chai Junfeng Luo Xiaolin Wei Meituan Mingyuan Fan, Shenqi Lai. Rethinking bisenet for real-time semantic segmentation. *arXiv*, 2021.

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

[7] Yanrong Guo Shijie Hao, Yuan Zhou. *A Brief Survey on Semantic Segmentation with Deep Learning*. Elsevier B.V., 2020.

[8] Stefan Roth Vladlen Koltun TU Darmstadt Intel Labs Stephan R. Richter, Vibhav Vineet. Plaing for data: Ground truth from computer games. *arXiv*, 2016.

[9] Stefano Soatto Yanchao Yang. Fda: Fourier domain adaptation for semantic segmentation. *cvpr*, 2020.

[10] Samuel Schulter Kihyuk Sohn1 Ming-Hsuan Yang Manmohan Chandraker Yi-Hsuan Tsai, Wei-Chih Hung. Learning to adapt structured output space for semantic segmentation. *cvpr*, 2018.