Real-Time Domain Adaptation for Semantic Segmentation

Nicolò Caradonna
s316993
Politecnico di Torino
s316993@studenti.polito.it

Angelo Iannielli
s317887
Politecnico di Torino
s317887@studenti.polito.it

Rashad Gulmaliyev
s317705
Politecnico di Torino
s317705@studenti.polito.it

Abstract

The problem of semantic segmentation is one of the main areas in which the use of deep learning techniques has been most successful. This problem is based on assigning a label to each pixel of an image. However, collecting annotations pixel by pixel is extremely time-consuming. For this reason, it is common to use as training datasets, synthetic datasets generated by software that produce images that are very similar to real things, with accurate pixel-wise ground truth at lower cost. An example is images taken from video games such as GTA5. The use of these datasets, however, leads to a problem called Domain Shift, which causes a significant drop in performance due to the substantial differences between a real dataset and a synthetic one. In this report, we present a series of experiments aimed at evaluating some semantic segmentation and domain adaptation techniques found in literature, showing how domain shift causes a drop in model performance and how the use of a discriminator function allows for improvements by attenuating the domain shift.
Code available at: `https://github.com/Nicocarad/AML_Project.git`

1. Introduction

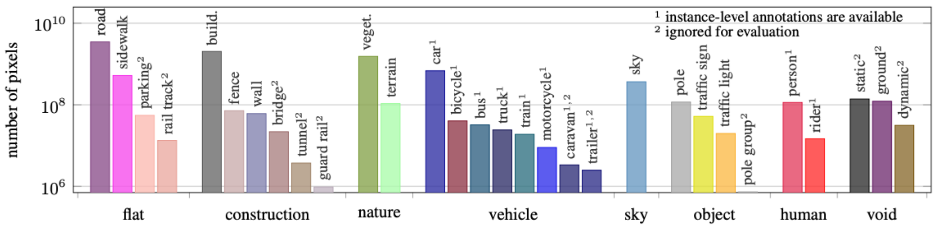
Real-time semantic segmentation is a crucial task in computer vision, aiming to assign semantic labels to each pixel of an image. This task is essential for applications that require immediate interaction with the environment, such as autonomous driving [14, 28], pedestrian detection [2, 8], defect detection [26], etc. Semantic segmentation involves understanding the scene at a pixel level, distinguishing between different objects and background elements. It goes beyond mere object detection by providing a detailed classification of each pixel, which is vital for understanding complex scenes in real-time scenarios. The main motive in real-time semantic segmentation is achieving a balance between accuracy and computational efficiency. Traditional methods [27], such as methods based on support vector machine (SVM) and decision tree, sacrifice spatial resolution to speed up inference, which negatively impacts the segmentation performance around object boundaries and small details.

In recent years, the advancement of deep learning [6, 15, 17, 31] has significantly improved the performance of semantic segmentation algorithms. Propose of designing Convolutional neural networks (CNNs) has been particularly influential, enabling the development of models that can process high-resolution images in real time while maintaining high accuracy. However, these models often struggle with domain shift, where the training (source) domain differs from the application (target) domain, leading to degraded performance. Domain adaptation techniques are crucial in semi-supervised learning scenarios where the source domain contains a large amount of fully annotated data, while the target domain has few or no labeled data. The key aspect of this approach is to use sufficient labeled photo-realistic synthetic data. The primary challenge is to address the domain discrepancy between the synthetic and the real data. In our work, we use the Cityscapes [7] dataset as a target and the GTA5 [20] dataset (synthetic labels) as a source.

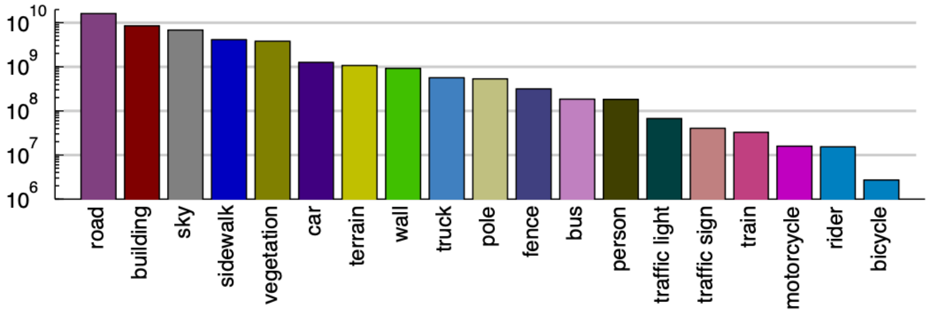
Dataset specifications:

Cityscapes [7] Dataset: (Fig.1a) This dataset is tailored for autonomous driving in an urban environment and involves a much wider range of highly complex inner city street scenes that were recorded in 50 different cities, covering spring, summer, and fall during the span of several months. It provides high-quality pixel-level annotations of urban street scenes. These 5,000 fine pixel-level annotations consist of layered polygons (à la LabelMe [23]). 20.000 images with coarse annotations making it a valuable resource for training and evaluating segmentation models. Here, 30 visual classes are defined for annotation, which are grouped into 8 categories, flat construction, nature, vehicle, sky, object, human, and void.

GTA5 [20] Dataset: (Fig.1b) This dataset is generated from the Grand Theft Auto V game and includes 24,966 frames with high-quality pixel-level annotations. Classes



(a) Cityscapes



(b) GTA5

Figure 1. Number of annotated pixels per class in datasets. Note the logarithmic scale.

were defined to be compatible with other semantic segmentation datasets, such as CamVid [3], KITTI [21], and Cityscapes [7], for outdoor scenes. The use of synthetic data from GTA5 [20] helps in training models that can generalize well to real-world scenarios without requiring extensive manual labeling.

This project focuses on implementing and enhancing some techniques for real-time semantic segmentation, aiming to improve their applicability and performance in diverse environments.

2. Related works

2.1. Semantic Segmentation

Semantic segmentation involves assigning each pixel a semantic label, e.g., person, car, road, or tree, in an image. (Fig. 2) The Evolution of semantic segmentation has seen a shift from traditional machine learning methods to deep learning-based approaches, which leverage large datasets and powerful neural network architectures to achieve state-of-the-art results.

Traditional segmentation algorithms like threshold selection and super-pixels were limited by their reliance on handcrafted features and the inability to capture complex patterns. The introduction of methods [1, 5, 12, 34] based on fully convolutional networks (FCNs) [18] revolutionized the field by enabling end-to-end learning of pixel-wise predictions from raw image data. Subsequent models, such as SegNet [1], DeepLabV3 [5], and PSPNet [33], have further refined this approach by incorporating mechanisms like encoder-decoder structures, dilated convolutions, and pyramid pooling to improve spatial resolution and context aggregation.

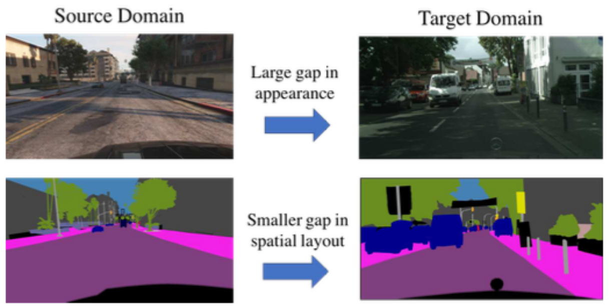


Figure 2. Our motivation for learning adaptation in the output space. While images may be very different in appearance, their outputs are structured and share many similarities, such as spatial layout and local context.

2.2. Real-time Semantic Segmentation

The need for real-time performance in applications like autonomous driving has driven the development of two mainstreams to devise efficient segmentation methods. *(i) lightweight backbone*. DFANet [15] adopts a lightweight backbone to reduce computation costs and introduces a cross-level feature aggregation module to enhance performance. DFNet [16] utilizes a "Partial Order Pruning" algorithm to obtain a lightweight backbone and an efficient decoder. *(ii) multi-branch architecture*. ICNet [33] devises a multi-scale image cascade to achieve a good speed-accuracy trade-off. Techniques such as Bilateral Segmentation Network (BiSeNet) [32] and Short-Term Dense Concatenate (STDC) network focus on maintaining high inference speeds while balancing accuracy. BiSeNet [32] achieves this by using a two-path architecture to separately process spatial and contextual information, while STDC reduces redundancy and integrates feature aggregation directly into the network design.

2.3. Domain Adaptation

Domain adaptation aims to mitigate the performance drop when a model trained on one domain (source) is applied to a different domain (target). Traditional methods include *discrepancy-based approaches*, which minimize the difference between source and target distributions, and *adversarial learning*, which uses a discriminator network to align features across domains. Recent advancements have explored using self-supervision and auxiliary tasks to improve domain adaptation performance further.

Discrepancy-Based Methods. Discrepancy-based methods explicitly measure and minimize the discrepancy between the source and target domains on corresponding activation layers of the two network streams. Long *et al*. [18] designed a Deep Adaptation Network, where the discrepancy is defined as the sum of the multiple kernel variants of maximum mean discrepancies (MK-MMD) between the fully connected (FC) layers. Sun *et al*. [25] proposed correlation alignment (CORAL) to minimize domain shift by aligning the second-order statistics of the source and target features of the last FC layer. Recent methods extend these ideas by incorporating higher-order moment matching (HoMM) [6] or aligning distributions [13, 22] at multiple network levels, such as convolutional and fully connected layers.

Adversarial Discriminative Methods. Adversarial discriminative models usually employ a domain discriminator with an adversarial loss to encourage domain confusion. Ganin *et al*. [9] proposed a Domain-Adversarial Neural Networks (DANN) algorithm to optimize a gradient reversal layer to achieve domain invariance. Long *et al*. [19] considered aligning conditional distribution across domains and proposed Conditional Domain Adversarial Networks (CDAN), which condition the discriminator on both features and classifier predictions. These methods align the marginal distributions and conditional distributions to improve adaptation performance.

Adversarial Generative Models. Adversarial generative models leverage generative adversarial networks (GANs) [10] to create domain-invariant representations. For example, CycleGAN-based methods [11,29,35] enforce cycle consistency to ensure that the translated images retain semantic content, facilitating better alignment at both pixel and feature levels. Methods like SimGAN [24] refine synthetic data to make it more realistic, preserving the annotations while enhancing domain adaptation.

Unsupervised Adversarial Domain Adaptation. Unsupervised adversarial domain adaptation (UADA) is particularly relevant for semantics segmentation, where labeled target data is often unavailable. Techniques such as adversarial training with Generative Adversarial Networks (GANs) [10] have shown promise in creating robust models that can generalize well across different environments. Methods like Cycle-Consistent Adversarial Domain Adaptation (CyCADA) [11] adapt representations at both pixel and feature levels, leveraging cycle consistency and task-specific losses to enhance segmentation performance.

3. Algorithmic Overview

The algorithms used in this report are already present in the literature. In particular, the Segmentation Network and Discriminator Network are taken from [31] and [30], respectively.

3.1. Segmentation Network

The paper "Rethinking BiSeNet For Real-time Semantic Segmentation" [31] introduces an innovative method to enhance the efficiency of real-time semantic segmentation by addressing the limitations of existing networks such as BiSeNet [4]. The core of the proposed method is the Short-Term Dense Concatenate (STDC) module, which offers a novel approach to balance segmentation accuracy and processing speed.

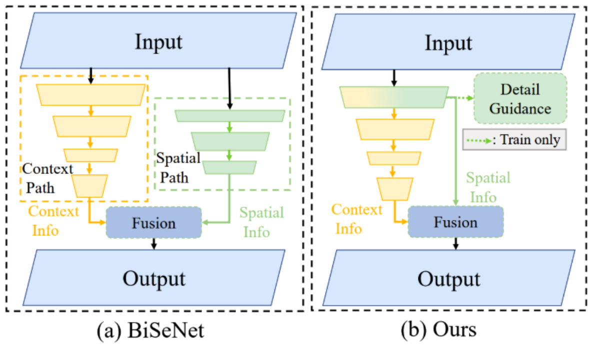


Figure 3. Illustration of architectures of BiSeNet [4] and our proposed approach. (a) presents Bilateral Segmentation Network (BiSeNet [4]), which uses an extra Spatial Path to encode spatial information. (b) demonstrates our proposed method, which uses a Detail Guidance module to encode spatial information in the low-level features without an extra time-consuming path.

3.1.1 Bilateral Segmentation Network (BiSeNet)

BiSeNet V1 [4] introduced a dual-path architecture for real-time semantic segmentation, utilizing an additional spatial path to encode spatial details while the context path focused on semantic information. However, this architecture has shown significant limitations:
Structural Redundancy: The additional spatial path introduces excessive complexity, which can slow down the inference process.
Computational Cost: The spatial path requires additional computational resources, making the network less efficient in contexts where speed is crucial. To address these issues, the proposed method eliminates the need for a separate spatial path by integrating spatial detail capabilities directly into the lower levels of the STDC module. This approach not only reduces computational complexity but also improves network efficiency while maintaining or enhancing segmentation accuracy.

3.1.2 Short-Term Dense Concatenate (STDC)

The STDC module is designed to optimize feature extraction efficiently. It consists of several blocks, each applying a series of convolutional operations, followed by batch normalization and ReLU activation. The structure of the module is such that the response maps from various blocks are concatenated, enabling the aggregation of information from different scales of representation. This approach allows for scalable receptive fields and multi-scale feature representation without adding significant computational load.
Specifically, each block within the STDC module applies convolutions with progressively smaller kernel sizes. This allows the module to capture fine details at lower levels and semantic information at higher levels. The gradual reduction in kernel size is intended to balance the reduction of computational complexity with the maintenance of segmentation quality.
The proposed method is articulated in several phases, combining STDC modules into a unified STDC network. This network is designed to maximize computational efficiency while maintaining high performance in terms of segmentation accuracy.
The STDC network uses multiple STDC modules distributed across various levels. Each module is responsible for capturing and aggregating information at different scales, providing a robust and multi-scale representation of image features.
A key element of the method is the Detail Aggregation module, which guides the learning of spatial details in the lower levels of the network. This module generates a "ground truth" of details, which is then used to optimize learning through binary cross-entropy and dice loss. This approach ensures that spatial details are learned efficiently without adding complexity during inference time.
At the end of the decoding process, spatial information from the lower levels is fused with semantic information from the higher levels to produce the final segmentation results. This fusion ensures that the network maintains fine details and semantic accuracy, delivering high-quality segmentation results.
More information about the segmentation network are available in the reference paper [31].

3.2. Discriminator Function

A discriminator function implemented as in [30] is used to perform Unsupervised Adversarial Domain adaptation. The network comprises five convolutional layers with a 4 × 4 kernel and a stride of 2. The number of channels for each layer is 64, 128, 256, 512, 1. Except for the final layer, each convolutional layer is followed by a leaky ReLU activation function with a negative slope of 0.2. An up-sampling layer after the last convolutional layer is added to resize the output to the input dimensions. Batch normalization layers are not included since we train the discriminator and the segmentation network together using a small batch size.

3.3. Training Procedure

Semantic Segmentation Training: For the first experiments to evaluate the segmentation network, we used the model without considering the discriminator function, implementing a training loop that trained the model on the respective training folders of Cityscapes and GTA5 datasets. Then we performed some data augmentation techniques by modifying, during image loading, some properties of the images. At this stage, we also performed Domain Shift evaluation by training the model on GTA5 and evaluating it on the Cityscapes validation set.

Domain Adaptation Training: To perform domain adaptation, the training loop previously used was modified to accommodate the discriminator function. Our domain adaptation algorithm comprises two components: a segmentation network (**G**) and a discriminator (**D**). We have two sets of images, representing the source and target domains, denoted as I_s and I_t. First, we process the source image (I_s), which includes annotations, through the segmentation network to optimize **G**. Next, we generate the segmentation softmax output (P_t) for the target image (I_t), which lacks annotations. Our objective is to make the segmentation outputs P for both the source and target images (P_s and P_t) similar. To achieve this, we input these two predictions into the discriminator (**D**) to determine if the input is from the source or target domain.

By applying an adversarial loss to the target prediction, gradients are propagated from **D** to **G**, encouraging **G** to produce segmentation distributions in the target domain that resemble those in the source domain.

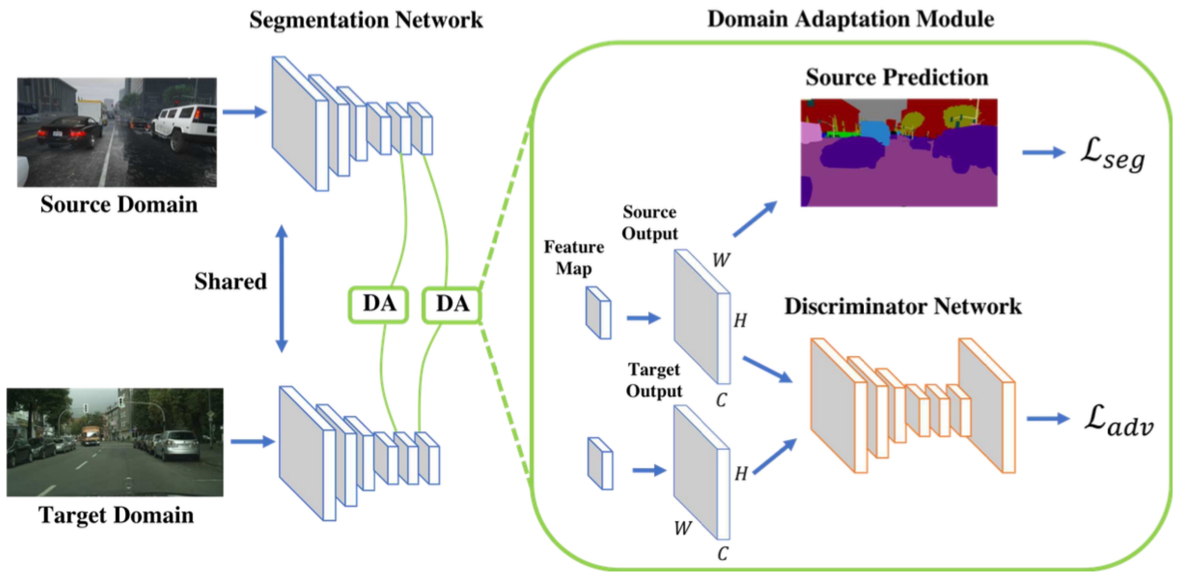


Figure 4. Given images of size $H \times W$ from both source and target domains, we pass them through the segmentation network to generate output predictions. For source predictions with C categories, a segmentation loss is computed using the source ground truth. To align target predictions more closely with the source predictions, we employ a discriminator to distinguish whether the input is from the source or target domain. An adversarial loss is then calculated on the target prediction and back-propagated to the segmentation network. We refer to this process as an adaptation module.

Discriminator Training: We also describe the training objective for the discriminator. Given the segmentation softmax output $P = G(I)$, we forward P to a fully convolutional discriminator D using a cross-entropy loss L_d for the two classes (i.e., source and target).
The loss can be written as:

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

where $z = 1$ if the sample is drawn from the target domain, and $z = 0$ if the sample is from the source domain.
Segmentation Network Training: First, we define the segmentation loss for images from the source domain as the cross-entropy loss:

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

where Y_s is the ground truth annotations for source images and $P_s = G(I_s)$ is the segmentation output.

Second, for images in the target domain, we forward them to G and obtain the prediction $P_t = G(I_t)$. To make the distribution of P_t closer to P_s, we use an adversarial loss L_{adv}:

$$\mathcal{L}_{\text{adv}}(I_t) = -\sum_{h,w}\log(D(P_t)^{(h,w,0)}). \tag{3}$$

This loss is designed to train the segmentation network and fool the discriminator by maximizing the probability of the target prediction being considered as the source prediction. A reference image of the training procedure is shown in Figure 4

4. Experimental Results

The methods used in this paper were tested on two datasets: Cityscapes [7] and GTA5 [20] in order to evaluate the effectiveness of the segmentation network and loss due to the domain shift problem as well as evaluate domain adaptation techniques. We first introduce the datasets that were used along with the metrics used to evaluate the models and then we show the results obtained for Semantic Segmentation and Domain Adaptation jointly with some parameter optimization experiments. For all the following experiments, we used the parameters reported in the table 1.

Hyper-parameter	Value
backbone	STDCNet813
optimizer	Adam
crop height	512
crop width	1024
epochs	50
batch size	8
learning rate	0.0001
num classes	19

Table 1. Default values for model hyperparameters

4.1. Datasets and Metrics

Cityscapes: Cityscapes [7] is a semantic scene parsing dataset, which is taken from a car perspective. It contains 2252 fine annotated images and is split into training and validation with 1752 and 500 images respectively. The annotation includes 30 classes, 19 of which are used for semantic segmentation tasks. The images have a high resolution of 2,048 × 1,024 which size will be reduced to 1024 x 512 before passing images to the model.

GTA5: GTA5 [20] dataset contains 2500 synthetic images with pixel-level semantic annotation. The images have been rendered using the open-world video game Grand Theft Auto 5 and are all from the car perspective in the streets of American-style virtual cities. The dataset has been split in order to obtain a training set of 2000 images and a validation set of 500. There are 19 semantic classes that are compatible with the ones of the Cityscapes [7] dataset.
Evaluation Metrics: For all the tests, Pixel Accuracy and Mean Intersection over Union (mIoU) metrics were used in addition to the average training time per epoch used to evaluate the performances in terms of time of the model.

4.2. Semantic Segmentation Baseline

First, we evaluate the segmentation network's performance separately on each dataset by training on the training set and testing on the validation set. This provides an upper bound for our problem

Experiment	Accuracy(%)	mIoU(%)	Training Time (avg per epochs)
STDC 1 - 50 ep (Cityscapes)	**77.8**	**46.98**	**1.56**

Table 2. Results of STDC network (pre-trained on ImageNet) on Cityscapes training and validation folder.

Experiment	Accuracy(%)	mIoU(%)	Training Time (avg per epochs)
STDC 1 - 50 ep (GTA5)	**79.5**	**56.3**	**2.15**

Table 3. Results of STDC network (pre-trained on ImageNet) on GTA5 training and validation folder.

The results show that good results can be obtained by training the model directly on the dataset consisting of real images. This ideal scenario will provide our upper bound for subsequent experiments.

4.3. Domain Shift

The second experiment is aimed at evaluating the domain shift that is obtained when the synthetic image dataset, thus GTA5, is used to train the model, evaluating the performance on the Cityscapes dataset. What we expect from this result, is a very noticeable drop in segmentation performance since although both datasets represent images of cities, the two datasets have several differences in terms of lighting and environment representation in addition we have to remember that GTA5 images are synthetic being extracted from a video game.

Experiment	Accuracy(%)	mIoU(%)
STDC 1 - 50 ep (GTA5 to Cityscapes)	**46.8**	**15.1**

Table 4. Results of STDC network (pre-trained on ImageNet) on GTA5 training and Cityscapes validation folder

As we expected, the drop in performance is significant for both Accuracy and mIoU due to the gap between artificial images and real images.

4.3.1 Domain Shift with Data Augmentation

One attempt to slightly improve the performance of our model is to apply data augmentation techniques to the model. This could increase the generalization of our training dataset, resulting in an increase in the performance of the model.
Flips and changes in image characteristics such as brightness, saturation, and contrast were used as data augmentation techniques, with a probability to apply augmentation setted to 0.5.

Experiment	Accuracy(%)	mIoU(%)	Training Time (avg per epochs)
STDC 1 - 50 ep	**44**	**19.4**	**2.15**

Table 5. Results of STDC network (pre-trained on ImageNet) on GTA5 training and Cityscapes validation folder, applying data augmentation to GTA5 images

The use of such techniques actually produces an improvement in model performance, however, the gap with our upper bound (2) remains clear.

4.4. Unsupervised Adversarial Domain Adaptation

The next experiment, is to attempt to improve the performance drop due to domain shift, using Adversarial Domain Adaptation techniques specifically by implementing a discriminator function. The results we expect are an increase in performance compared to the previous experiment on Domain Shift, however, the results will still be lower than the upper bound evaluated in the first experiment.

Experiment	Accuracy(%)	mIoU(%)	Training Time (avg per epochs)
STDC 1 - 50 ep	**49.4**	**21.0**	**3.40**

Table 6. Results of STDC network (pre-trained on ImageNet) on GTA5 training and Cityscapes validation folder with 50 epochs of training. Default parameters have been used for batch size, learning rate, etc.

As we expected, the introduction of the discriminator function produces an improvement in the performance of our model by partially resolving the performance drop introduced by the Domain Shift

4.5. Hyper-Parameter Optimization

As a final attempt to improve the performance of our model again by application of the discriminator function, we try parameter optimization by changing some parameters of our model that are not subject to the training. An example of these parameters are learning rate, batch size, number of epochs, weight decay, and the adversarial loss. For this experiment, we used the parameters of section 4.4 as a baseline, by alternatively modifying one of the untrainable parameters of the model.

4.5.1 Adversarial Loss Optimization

The first test to be performed is to find an optimal value for λ_{adv} that is a parameter to weight the Adversarial Loss and the Segmentation Loss. Four values of λ_{adv} were tested, remembering that the default one used in the previous step is 0.001

λ_{adv}	Accuracy (%)	mIoU (%)
0.001	**49.4**	**21.0**
0.0005	51.8	22.5
0.0002	50.8	21.5
0.0001	**52.7**	**22.9**

Table 7. Results of Adversarial Domain Adaptation Task with different values for the adversarial loss parameter λ_{adv}

By bringing the λ_{adv}v to 0.0001 we are able to achieve a performance increase of about 2% mIoU.

4.5.2 Epochs Optimization

We also tried to change the number of training epochs looking for possible overfitting problems. Remember that an epoch is the complete passage of the training algorithm along the entire training set.

# epochs	Accuracy (%)	mIoU (%)
10	37.6	16.7
20	38.5	19.5
30	45.8	20.9
40	**49.4**	**21.1**
50	**48.8**	**21.0**

Table 8. Results of Adversarial Domain Adaptation Task with different values for training epochs

Changing the number of epochs shows that increasing the number of epochs improves the performance of the model, so the default value of 50 epochs should be considered optimal. It might be useful to use a lower number of epochs to reduce the training time without affecting the performance of the model.

4.5.3 Model Learning Rate Optimization

Another attempt is to change the initial value of the learning rate passed to the optimizer used by the model.

We can imporve performance changing the default value to 0.0002.

learning rate	Accuracy (%)	mIoU (%)
0.001	46.2	15.4
0.0005	53.1	18.8
0.0002	**53.1**	**21.5**
0.0001	**48.8**	**21.0**

Table 9. Results of Adversarial Domain Adaptation Task with different values for the learning rate of the model

4.5.4 Batch Size Optimization

An additional hyperparameter that we tried to change is the batch size. Remember that batch size indicates the number of training samples that are passed to the model before the gradients are computed and the model weights are updated.

batch size	Accuracy (%)	mIoU (%)
8	**48.8**	**21.0**
16	**55.7**	**24.0**
32	53.0	22.1

Table 10. Results of Adversarial Domain Adaptation Task with different values for the batch size λ_{adv}

As we see from the table, changing batch size to 16 can improve performances.

4.5.5 Discriminator Learning Rate Optimization

The last Hyperparameter to optimize is the learning rate of the discriminator.

learning rate D	Accuracy (%)	mIoU (%)
0.001	25.8	3.8
0.0005	**56.4**	**22.6**
0.0002	54.6	21.8
0.0001	**48.8**	**21.0**
0.00005	50.0	20.9

Table 11. Results of Adversarial Domain Adaptation Task with different values for the learning rate of the discriminator λ_{adv}

4.5.6 Best Configurations

Considering the best hyperparameters obtained from the previous steps, we tried to train the model with the same to improve the results even more. We used a batch size of 16, with a learning rate for the model and the discriminator of 0.0002 and 0.0005 respectively. We also used λ_{adv} 1e-4 training the model for 60 epochs.

Experiment	Accuracy (%)	mIoU (%)
STDC with discriminator	**61.8**	**23.3**

Table 12. Results of Adversarial Domain Adaptation Task with different best hyperparameters λ_{adv}

Using all the best hyperparameters we get a slight deterioration in mIoU compared with the best value obtained from our tests, but we get a significant imporvement in accuracy.

5. Conclusion

In this report, we examined how domain adaptation techniques affect semantic segmentation using synthetic datasets. We evaluated the STDC network trained on GTA5 and tested it on Cityscapes.

Our initial experiments showed a significant performance drop due to domain shift when applying a model trained on synthetic data to real-world data. Data augmentation techniques like flips, and adjustments to brightness, saturation, and contrast slightly improved performance but still left a notable gap.

Adversarial domain adaptation, using a discriminator function, improved performance by reducing discrepancies between synthetic and real data. Further enhancements came from hyper-parameter optimization, including tuning learning rate, batch size, number of epochs, and adversarial loss.

Overall, our findings highlight the importance of domain adaptation techniques in closing the performance gap between synthetic and real datasets. While adversarial adaptation and hyper-parameter tuning yielded significant improvements, more advanced adaptation techniques and data augmentation strategies are needed to achieve optimal performance. However, the Domain Adaptation problem remains an open and challenging issue in the world of deep learning, and other Domain Adaptation methods can be implemented to try to further improve performance.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017. 2

[2] G. Brazil, X. Yin, and X. Liu. Illuminating pedestrians via simultaneous detection & segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4950–4959, 2017. 1

[3] G.J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2), 2009. 2

[4] Chao Peng Changxin Gao Gang Yu Changqian Yu, Jingbo Wang and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *In Proceed- ings of the European conference on computer vision (ECCV)*, page 325–341, 2018. 3, 4

[5] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017. arXiv preprint arXiv:1706.05587. 2

[6] Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. Fasterseg: Searching for faster real-time semantic segmentation. In *International Conference on Learning Representations*, 2020. 1, 3

[7] 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. In *CVPR*, 2016. 1, 2, 5, 6

[8] F. Flohr and et al. D. Gavrila. Pedcut: an iterative framework for pedestrian segmentation combining shape models and multiple data cues. In *BMVC*, 2013. 1

[9] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030, 2016. 3

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014. 3

[11] 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. In *ICML*, pages 1994–2003, 2018. 3

[12] Ping Hu, Fabian Caba, Oliver Wang, Zhe Lin, Stan Sclaroff, and Federico Perazzi. Temporally distributed networks for fast video semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8818–8827, 2020. 2

[13] C.-Y. Lee, T. Batra, M. H. Baig, and D. Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *CVPR*, pages 10285–10295, 2019. 3

[14] B. Li, S. Liu, W. Xu, and W. Qiu. Real-time object detection and semantic segmentation for autonomous driving. In *Proceedings of the Multispectral Image Processing and Pattern Recognition: Automatic Target Recognition and Navigation*, volume 10608, page 106080P. International Society for Optics and Photonics, 2018. 1

[15] Hanchao Li, Pengfei Xiong, Haoqiang Fan, and Jian Sun. Dfanet: Deep feature aggregation for real-time semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9522–9531, 2019. 1, 2

[16] Xin Li, Yiming Zhou, Zheng Pan, and Jiashi Feng. Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pages 9145–9153, 2019. 2

[17] Peiwen Lin, Peng Sun, Guangliang Cheng, Sirui Xie, Xi Li, and Jianping Shi. Graph-guided architecture search for real-time semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4203–4212, 2020. 1

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 2, 3

[19] M. Long, Z. Cao, J. Wang, and M. I. Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, pages 1640–1650, 2018. 3

[20] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 1, 2, 5, 6

[21] G. Ros, S. Ramos, M. Granados, A. Bakhtiary, D. Vázquez, and A.M. López. Vision-based offline-online perception paradigm for autonomous driving. In *WACV*, 2015. 2

[22] S. Roy, A. Siarohin, E. Sangineto, S. R. Bulo, N. Sebe, and E. Ricci. Unsupervised domain adaptation using feature-whitening and consensus loss. In *CVPR*, pages 9471–9480, 2019. 3

[23] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: A database and web-based tool for image annotation. *IJCV*, 77(1-3):157–173, 2008. 1

[24] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, pages 2242–2251, 2017. 3

[25] B. Sun, J. Feng, and K. Saenko. Correlation alignment for unsupervised domain adaptation. In *Domain Adaptation in Computer Vision Applications*, pages 153–171, 2017. 3

[26] X. Tao, D. Zhang, W. Ma, X. Liu, and D. Xu. Automatic metallic surface defect detection and recognition with convolutional neural networks. *Applied Sciences*, 8(9):1575, 2018. 1

[27] M. Thoma. A survey of semantic segmentation, 2016. arXiv preprint arXiv:1602.06541. 1

[28] Y.-H. Tseng and S.-S. Jan. Combination of computer vision detection and segmentation for autonomous driving. In *2018 IEEE/ION Position, Location and Navigation Symposium*, pages 1047–1052. IEEE, 2018. 1

[29] E. Tzeng, K. Burns, K. Saenko, and T. Darrell. Splat: semantic pixel-level adaptation transforms for detection, 2018. arXiv:1812.00929. 3

[30] Samuel Schulter Kihyuk Sohn Ming-Hsuan Yang Manmohan Chandraker Yi-Hsuan Tsai, Wei-Chih Hung. Learning to adapt structured output space for semantic segmentation. 2018. 3, 4

[31] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation, 2020. arXiv preprint arXiv:2004.02147. 1, 3, 4

[32] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018. 3

[33] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–420, 2018. 2, 3

[34] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 2

[35] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2223–2232, 2017. 3