Domain Adaptation Strategy for efficient Semantic Segmentation

Alessandro Ciarrocchi
s344430

Arianna Lezzi
s347096

Diletta Ceschini
s333117

Emanuela Piga
s333550

June 25, 2025

Abstract

In this project, we explore the task of Domain Adaptation in the context of Semantic Segmentation. The study focuses on adapting models trained on the synthetic GTA5 dataset to effectively segment images from the Cityscapes dataset, which represents real urban scenarios. The aim is to gain a deeper understanding of the limitations and capabilities of real-time networks and to evaluate the impact of lightweight adaptation strategies on model generalization. To this end, several experiments are conducted using commonly adopted architectures, such as DeepLab and BiSeNet, with an emphasis on assessing their performance under domain shift conditions. Multiple strategies are then implemented to reduce the domain gap, including data augmentation techniques and adversarial training approaches. In the final phase of the project, more advanced methods are introduced, while also addressing class imbalance issues. The outcomes obtained by solving the class imbalance problem represent a promising basis for deeper exploration within this domain. Code is available at `https://github.com/ariannalezzi/MLDL.git`

1. Introduction

We are dealing with one of the core tasks in computer vision that involves assigning a semantic label to every pixel in an image. This pixel-level classification allows for fine-grained understanding of visual scenes by differentiating between a variety of semantic regions that, in our case, are part of an urban setting (e.g. car, roads, people, vegetation, and sky). Although semantic segmentation has achieved notable performance on benchmark datasets, deploying these models in real-world scenarios remains challenging due to the discrepancy between the data distribution of the training set and that of the target deployment environments. This domain gap often arises from variations in illumination, texture, resolution, or overall scene composition.

While earlier research on domain adaptation has mainly focused on high-capacity architectures with extensive resources, more recent efforts aim to bring these techniques to lightweight models suitable for real-time applications. Among these, one that certainly deserves a mention is BiSeNet, a segmentation architecture developed to balance inference speed and accuracy. Our work begins by training the chosen real-time segmentation network (in this case BiSeNet) on the Cityscapes dataset to establish an upper performance bound under ideal conditions. Then, we train the same model on a synthetic dataset (GTA5) and directly evaluate its performance on Cityscapes to quantify the domain shift effect. A proposed naive solution to improve the generalization ability of a segmentation network trained on a synthetic domain is the application of combined data augmentation techniques. Another method used to tackle the problem of the domain shift is the adversarial approach, which is based on a competitive mechanism between two components of the model, respectively known as feature extractor and discriminator. Additionally, we experiment with a simple approach to solve the class imbalance problem by using a weighted sampler, adopted to create a training dataset that includes a higher number of images containing underrepresented classes. To further address the domain gap problem, we also relied on approaches such as Constructive Unpaired Translation (CUT), which generates a new dataset by translating images into the target domain's style. This approach enabled us to train BiSeNet using GTA5 images transformed to closely resemble the visual characteristics of the Cityscapes dataset. Finally, we propose a solution based on the redesign of U-Net's architecture. In summary, through this paper we seek to understand the difficulties of using real-time segmentation in challenging conditions and to evaluate how well lightweight adaptation methods work, also proposing some extensions in order to further improve the performance of the domain adaptation task.

2. Related Works

2.1. Semantic Segmentation

The growing impact of semantic segmentation, along with practical demands and technical constraints, has driven a shift from low-level approaches to more precise and complex methods. A major turning point in this evolution was marked by the introduction of Fully Convolutional Networks (FCNs) by Long et al. in 2015 [4], who repurposed traditional Convolutional Neural Networks used for classification to generate dense, pixel-wise predictions. This ground-braking work has led to many advances that improved segmentation for more accurate and refined analysis [6], [5].

In recent years, multiple solutions have been proposed to address challenges related to computational efficiency and the degradation of spatial resolution. These challenges have been addressed, to some extent, through the development of the DeepLab architecture and the use of atrous convolutions [1], which offer a way to enlarge the filters' field of view. Unlike in traditional pooling operations, this allows spatial information to be retained without increasing the parameters or the overall computational load.

Current advancements have focused on improving both accuracy and generalization capabilities of semantic segmentation networks. Architectures such as BiSeNet (Bilateral Segmentation Network) [11] have been specifically designed to balance inference speed and accuracy.

2.2. Domain Adaptation

Due to the use of different datasets for training and validation in semantic segmentation, this mismatch leads to a domain shift and a subsequent drop in performance.

Earlier studies primarily relied on Unsupervised Domain Adaptation (UDA) techniques, which do not require annotations from the target domain. However, they typically depend on a substantial number of samples to train the model effectively. More recently, with the advancement of deep learning, these methods have evolved into end-to-end models capable of learning domain-invariant features [12]. This progression has led to improved adaptability when transferring knowledge from synthetic to real data. In this work, we train on synthetic GTA5 data and evaluate on real Cityscapes images to analyze the effects of domain shift on real-time segmentation models.

2.3. Real Time Semantic Segmentation

While many domain adaptation approaches prioritize high-performing yet computationally intensive architectures, recent research emphasizes real-time semantic segmentation. BiSeNet represents a seminal contribution in this direction, enabling rapid inference while maintaining competitive accuracy. However, balancing speed and precision remains a central challenge: lightweight models typically struggle in complex scenarios and exhibit limited domain generalization capabilities. Consequently, integrating domain adaptation techniques into real-time architectures has become a critical area of investigation. Our analysis contributes to this field by assessing the effectiveness of domain adaptation strategies in enhancing the performance of real-time networks such as BiSeNet [11], trained on synthetic dataset like GTA5 and evaluated on real-world images from Cityscapes.

2.4. Adversarial learning

The concept of Adversarial learning was first introduced by GoodFellow et al. in 2014 [3] who described it as a competitive process with the aim of generating realistic results. Initially applied to tasks like image generation [10] and synthesis, in recent years it has been extended to semantic segmentation with the purpose of overcoming the domain shift. In this context, we propose an alternative to the work conducted in [9], by using a real-time network (BiSeNet) as a feature extractor.

3. Methodology

3.1. Dataset

We evaluate our method on two datasets that are able to capture the main details of an urban environment. Below are described the specifics of each dataset.

- **GTA5** [8] is a synthetic dataset of images collected from the homonymous videogame. It contains 2500 images with resolution 1280×720 and includes 19 classes from city settings.
- **Cityscapes** [2] is made up of around 2000 real-life images from different German cities, recorded by a camera fixed to a car, with a resolution of 2048×1024. It consists of 19 classes representing different objects of urban scenarios. In our experiments we use a resized version with resolution 1024×512.

3.2. Metrics

To evaluate the performance we use *mean Intersection over Union* (mIoU) and *Intersection over Union* per class, to measure the overlap between the predicted segmentation and the ground truth. *Latency* assesses the time delay between providing an input to the model and receiving the corresponding output. We employ *Frame Per Seconds* to estimate the inference speed of a model, which indicates how many frames the model can process per second. *FLOPs* quantifies the computational complexity of a model by counting the number of floating-point operations required to process a single input. *Params* refers to the total number of learnable weights in a model. This metric reflects the model's capacity and memory footprint.

3.3. Network Details

In the following section we describe the approach used to conduct our studies, providing the adopted configurations, the employed networks and the main stages of their training and evaluation process. **Deeplab** is the first framework we embrace for our evaluations. It belongs to a family of CNNs particularly effective in processing high-resolution images and capturing fine-grained spatial details, making them well-suited for semantic segmentation. Specifically, we adopt DeepLabV2, which consists of a backbone network (ResNet-101 in our case) pre-trained on the ImageNet dataset, whose task is to extract feature maps from the input images. In order to preserve spatial resolution and enlarge the receptive field without increasing the computational resources, the extracted feature maps are processed using *atrous convolutions*. This technique allows the network to modulate the resolution of feature responses while preserving a broad contextual understanding of the scene. Subsequently, the features are further refined through the *Atrous Spatial Pyramid Pooling (ASPP)* module, which applies parallel atrous convolutions with multiple dilation rates, allowing the model to capture information at various spatial scales. This enables the simultaneous recognition of both small-scale and large-scale objects, as well as the integration of contextual information from different receptive fields. The resulting feature maps are then upsampled to restore the original resolution of the input image, and used to generate the final segmentation map. Optionally, a fully connected Conditional Random Field (CRF) can be applied as a post-processing step to refine the segmentation results. We train DeepLab using Cityscapes dataset, which was already partitioned into training and validation sets. All images are resized to a resolution of 1024 for both training and validation phases. The hyperparameters employed during training are detailed in Table 2. The final performance metrics, including mean Intersection over Union (mIoU), floating-point operations per second (FLOPS), and inference latency, are summarized in Table 1. Considering as baseline the work [1], where the validation mIoU on Cityscapes reaches 66%, our model achieves a slightly lower performance, reaching a mIoU equal to 61%. This divergence can be mainly explained by differences in experimental setup, as the baseline model employed the full-resolution images and the entire dataset, whereas our approach rely on resized images and potentially fewer data samples.

Bisenet is the second architecture used in our project. It is a real-time segmentation network able to overcome the trade-off between spatial resolution and inference speed. It consists of two main modules: the Spatial Path and the Context Path. The *Spatial Path* is primarily responsible for preserving high-resolution spatial information, like shapes and small objects. It consists of three through convolutional layers that perform a moderate downsampling, allowing to encode rich spatial information while maintaining computational efficiency.

Model	DeepLabV2	BiSeNet
mIoU (%)	61.61	55.81
Latency (ms)	65.7611	6.8413
FPS	15.2089	146.1955
FLOPs	37.5×10^{10}	25.78×10^{9}
Params	43.901×10^{6}	12.582×10^{6}

Table 1. Comparison of DeepLabV2 and BiSeNet in terms of mIoU, latency, inference speed (FPS), FLOPs, and number of parameters.

In contrast, the *Context Path* is designed to provide a large receptive field for capturing high-level semantic context. This is obtained using a lightweight backbone that rapidly downsamples the input, enabling the extraction of global features, in our case we used ResNet18. To further improve the contextual representation, the Context Path incorporates global average pooling, which aggregates global information and improves the network's understanding of the scene as a whole. Additionally, we include an Attention Refinement Module (ARM) within the Context Path that restrains less informative features and enhances the most informative ones.

Finally, BiSeNet uses a *Feature Fusion Module* (FFM), to concatenate the output features of the Spatial Path and Context Path, since it is not possible to sum them because of their difference in level of features. To ensure consistency in attribute scaling and distribution, batch normalization is applied to the concatenated features. Subsequently, global pooling condenses the information into a compact vector, from which a weight vector is derived. This is then used to re-weight the fused representation, allowing the network to selectively emphasize informative elements while suppressing less relevant ones.

As for DeepLab, we trained BiSeNet with images from Cityscapes. Hyperparameters are shown in Table 2, while metrics are illustrated in Table 1. We achieve a validation mIoU of 55.81%, differently from our reference, [11], that reaches 78.6%. Unlike the original implementation, that uses random 1024×1024 crops, our tests employ 720×1024 crops. This reduced spatial resolution likely contributed to the performance gap by limiting the model's ability to capture fine details and full object context in large or complex scenes.

4. Domain Shift

Acquiring manually annotated images for semantic segmentation tasks is often both time-consuming and computa-

Hyperparameters	
epochs	50
classes	19
batches	2
base L.R.	$2.5e^{-4}$

Table 2. Hyperparameters used for DeepLabV2 and BiSeNet

tionally demanding. A widely adopted approach to mitigate this challenge involves the use of synthetic datasets, artificially generated images produced within simulated environments. In this phase of the project, we leverage the synthetic data provided by the GTA5 dataset to train BiSeNet according to what already defined in section 3.3. The trained model is subsequently evaluated on the Cityscapes dataset, in order to assess its generalization capabilities across domains. Performance assessment was conducted using the mIoU. The obtained results are reported in the first row of the table (3).

With respect to table 1 the mIoU is significantly lower when training on GTA5 and validating on Cityscapes due to a domain gap between synthetic and real-world data. GTA5, being a synthetic dataset, differs substantially from Cityscapes in visual appearance, texture realism, lighting conditions, and object representation. These differences affect the model's ability to generalize.

4.1. Data Augmentation Techniques

To enhance the generalization capability of the segmentation network, we repeated the previous experimental procedure by incorporating several data augmentation strategies aimed at both increasing the effective size of the training set and reducing the visual gap.

Initially, these transformations were applied individually in separate training runs to evaluate their impact on performance; subsequently, the augmentations that produced the most promising results were combined into single training pipelines. During training, each data augmentation was applied with a probability of 0.5, meaning that, for every training image, there was a 50% chance that a given transformation would be excluded. This probabilistic application was intended to introduce diversity in the training data without applying the same transformation to every image. Specifically, the following data augmentation techniques were implemented:

- **Random Crop**: randomly selects and crops a subregion of the input image, simulating partial views and promoting spatial invariance, thereby helping the model become less sensitive to the exact location of features within the image.
- **Random Brightness**: adjusts the brightness of the image by a random factor, helping the model handle varying lighting conditions.
- **Color Jitter**: randomly changes the brightness, contrast, saturation, and hue of the image, enhancing the model's robustness to color variations.
- **Gaussian Blur**: applies a Gaussian filter to smooth the image, simulating low-focus or de-focused conditions in real-world scenarios.
- **Motion Blur**: introduces a directional blur to replicate the visual effect of camera or object motion during image capture.
- **Gaussian Noise**: adds random noise sampled from a Gaussian distribution to the image, emulating sensor noise and real-world imperfections.

4.2. Results

The baseline model (without any augmentation) achieved a mIoU of 17.93, reflecting the performance gap due to domain shift between synthetic and real data.

Among all individual augmentations, Color Jitter (CJ) yielded the most significant improvement, raising mIoU to 23.52, followed by Gaussian Blur (GB) at 19.36, Motion Blur at 18.71 and Random Brightness (RB) at 18.23. These results can be attributed to the fact that such augmentations simulate realistic variations in lighting conditions, color distribution, and image sharpness, factors in which GTA5 and Cityscapes differ significantly. Consequently, combining Color Jitter, Gaussian Blur, and Random Brightness, yields the best overall performance, with a mIoU of 27.17, marking a $+9.24$ improvement over the baseline.

In contrast, augmentations like Random Crop (RC) and Gaussian Noise (GN) are less effective in this setting. Random crop can disrupt the spatial integrity of the scene by removing essential contextual information. This is especially problematic in urban driving datasets like GTA5 and Cityscapes, where the camera maintains a fixed, forward-facing viewpoint from a vehicle. As a result, objects tend to appear in consistent regions of the image, and the model learns to rely on these spatial patterns. Gaussian Noise, on the other hand, introduces low-level pixel disturbances that are not representative of the target domain: since Cityscapes images are captured with high-quality sensors under controlled conditions, adding synthetic noise may mislead the model by forcing it to learn visual patterns that are unlikely to appear in real-world scenarios.

To better understand the impact of data augmentation at a finer level, we categorize the semantic classes into four groups based on their final IoU scores and relative improvements with respect to the model obtained through the combination of Color Jitter, Gaussian Blur and Random brightness. The values in parentheses represent the final IoU score for each class, followed by the relative percentage improve-

Augmentation	**mIoU**	road	sidewalk	building	wall	fence	pole	light	sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motocycle	bicycle
Baseline	17.93	43.56	7.16	59.31	9.39	15.50	0.01	0.63	0.13	72.69	5.07	62.31	20.02	1.11	52.17	1.71	3.25	0.02	0.61	0.00
Random Crop (RC)	14.83	21.51	8.04	57.60	2.76	7.62	0.00	2.46	0.10	52.74	2.05	38.13	32.51	0.00	42.69	6.22	1.42	0.04	5.88	0.00
Gaussian Noise (GN)	16.03	17.68	8.70	59.70	1.93	0.59	0.00	0.72	0.46	65.96	2.97	71.63	22.64	0.07	41.77	3.48	0.80	0.00	5.29	0.10
Motion Blur (MB)	18.71	37.77	6.34	57.43	7.39	4.81	0.01	0.62	0.20	76.05	4.97	65.51	37.95	6.83	40.33	5.08	2.12	0.46	1.59	0.07
Color Jitter (CJ)	23.65	70.88	19.43	68.04	14.39	2.28	0.01	0.82	0.39	77.20	25.85	78.78	32.70	1.04	48.91	4.37	2.31	0.45	1.50	0.00
Gaussian Blur (GB)	19.36	41.93	7.49	65.22	11.74	13.33	0.01	0.85	0.48	74.62	4.61	68.07	36.27	0.72	38.03	2.63	0.76	0.10	1.01	0.00
Random Brightness (RB)	18.23	49.78	15.87	51.55	8.79	2.51	0.09	0.61	0.19	63.04	5.72	70.62	24.11	0.00	49.31	2.56	0.29	0.06	1.25	0.00
GN+CJ	20.64	51.17	13.38	63.07	7.70	2.70	0.02	0.40	0.24	72.66	5.42	79.74	26.14	1.96	44.87	11.54	1.02	1.18	8.46	0.52
GB+RB	22.97	69.30	16.37	71.72	12.84	9.76	0.01	1.26	0.16	71.47	6.71	71.58	36.41	2.42	55.74	5.12	1.60	0.00	3.91	0.12
CJ+GB+RB	**27.17**	80.06	27.90	74.07	20.85	12.83	0.02	0.92	0.32	77.45	12.76	80.33	40.81	5.78	65.28	8.59	2.81	0.06	5.44	0.01

Table 3. Per-class IoU and mIoU (%) across different data augmentation techniques applied to semantic segmentation.

ment compared to the baseline model without augmentation.

- **High final mIoU with strong improvement**: *Road* $(80.06, +83.8\%)$. Its frequent presence in consistent spatial positions makes it highly learnable and the augmentations help simulate real-world variability.
- **High final mIoU with moderate improvement**: *Building* $(74.07, +24.9\%)$, *Vegetation* $(77.45, +6.5\%)$, *Sky* $(80.33, +28.9\%)$, *Car* $(65.28, +25.1\%)$. These classes are well-defined and visually prominent in both datasets, and the model already generalizes well to them.
- **Large relative improvement but moderate final mIoU**: *Person* $(40.81, +103.8\%)$, *Sidewalk* $(27.90, +289.7\%)$, *Wall* $(20.85, +122.0\%)$, *Terrain* $(12.76, +151.7\%)$, *Truck* $(8.59, +402.3\%)$, *Motorcycle* $(5.44, +791.8\%)$. Despite their relative improvement, these classes remain difficult for the model to learn: the augmentation helps to recognize them, but performance is still limited by the inherent complexity and less consistent visual appearance.
- **Persistently low mIoU despite augmentation**: *Fence* $(12.83, -17.2\%)$, *Pole* (0.02), *Traffic Light* (0.92), *Traffic Sign* (0.32), *Bus* (2.81), *Train* (0.06), *Bike* (0.01). These classes remain poorly segmented even after augmentation. They are either small, thin, infrequent, or surrounded by confusing context (e.g. poles next to trees, fences bleeding with buildings). In these cases, the relative improvement is not reported, as it is not meaningful due to the extremly low baseline values.

5. Adversarial Learning

In this part of the project, we keep focusing on the issue of domain adaptation, by leveraging an alternative approach based on Adversarial Learning, inspired by the work in [9]. Adversarial learning is a competitive process between two learning agents, characterized by one model generating data and another distinguishing it, leading both components to improve through their opposing objectives. In our domain adaptation scenario for semantic segmentation the two agents are the *segmentation network* **G** and the *discriminator* **D**.

The **Segmentation Network** (also called "feature extractor") is the previously introduced BiSeNet. The role of this module is taking an input image from GTA5 (*source domain*) and returning a segmentation map. The segmentation network is trained to fool the discriminator by making images from Cityscapes (*target domain*) look like the source domain predictions.

The **Discriminator** is a small CNN that tries to distinguish wether the segmentation network's output comes from the source domain or the target domain . If the predictions on target domain images differ significantly from those on source domain images the discriminator will learn how to detect these differences. On the other hand, the segmentation network will try to produce more confident target prediction that resemble the source domain. This adversarial dynamic pushes the model to generalize to the target domain.

5.1. Results

Table 4 shows the results obtained in terms of mIoU and IoU for each class. We achieve a mIoU of 29.67%, that compared with the best result reported in table 3 shows a moderate increase. Most of the classes seems to perform equal or better with respect to the best result of previous section, with some of them almost doubling their performance (*truck* +6,49% iou). However, some classes were penalized by this method, in particular *pole* and *Traffic Sign* perform worse, with a really low Iou. small.

6. Experiments

6.1. U-Net

In the later stage of our study, we explore some changes in the BiSeNet architecture by replacing the context path of our Network with a U-Net encoder-decoder structure illustrated in 6.1.1. This idea came from the core concept of feature fusion, which is central to the BiSeNet architecture, that effectively combines spatially detailed features from a

Adversarial	**mIoU**	road	sidewalk	building	wall	fence	pole	light	sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motocycle	bicycle
BiSeNet	29.67	86.74	24.68	79.10	24.50	16.07	0.01	1.01	0.19	80.05	24.83	82.78	40.56	5.67	72.21	15.08	3.350	1.420	4.980	0.400

Table 4. Per-class IoU and mIoU (%) when applying adversarial learning

high-resolution spatial path with semantically rich features from a low-resolution context path. We posit that integrating a U-Net backbone could further enhance the quality of deep semantic features and improve the overall discriminative power of the fused feature representation. Indeed, U-Net is widely recognized for its ability to capture fine details in medical image segmentation. This modification aims to leverage the complementary strengths of both architectures: the semantic richness representation of U-Net and the efficient dual path design of BiSeNet. To this end, we compare the classical **U-Net** [6] with a modified version, **ResUNet** [13] which substitutes U-Net's default encoder with a ResNet18 or ResNet34 encoder pretrained on ImageNet, aiming to extract more meaningful features through residual learning.

To ensure a fair comparison, we also trained ResNet18 as a standalone segmentation model, consistent with its role in BiSeNet's context path. All three architectures were trained by adapting a set of pre-trained weights from ImageNet. This approach allowed us to save significant training time, as we did not need to train the models from scratch. Furthermore, by leveraging the same pretraining source, we ensured a fair comparison across models, avoiding variability due to differing initialization strategies.

Results

As we can see from the results in the table 5, the standalone ResNet18 model achieves the best overall performance across all evaluated metrics. Specifically, ResNet18 attains a mIoU of 49.39%, outperforming both ResUNet variants, which achieve 46.34% and 44.65% for ResUNet18 and ResUNet34 respectively. Thus, ResNet18 result in the highest Accuracy, 96.16%, confirming its superior segmentation capability in our experimental setup.

Given these results, continuing with the U-Net integration may not be justified, as ResNet18 alone already outperforms the U-Net based variants across all key metrics. For completeness, we present qualitative results in figure 6.1.2, comparing the segmentation masks produced by ResUNet18, ResUNet34, and ResNet18 on selected GTA5 samples. Although the outputs are visually similar, ResNet18 tends to capture more relevant structural features without the excessive detail observed in the U-Net variants, which may introduce unnecessary complexity detecting too many details. This suggests that, in opposition with our initial assumption, the effectiveness of ResNet18 makes it a more suitable context encoder for our application than U-Net.

Model	ResNet18	ResUNet18	ResUNet34
mIoU (%)	**49.39**	46.34	44.65
Accuracy	**96.16**	95.92	95.42

Table 5. Comparison between the three models

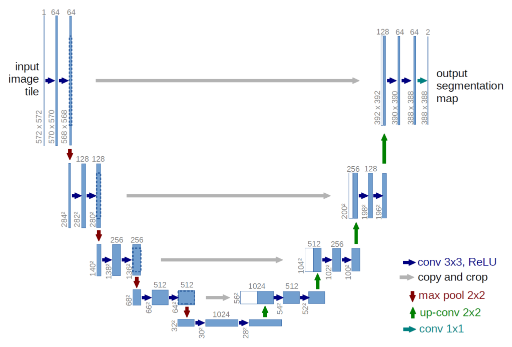

Figure 6.1.1. Standard Unet architecture

6.2. Constractive Unpaired Translation

Another approach we investigate in order to reduce the domain gap between GTA5 and Cityscapes is based on Constractive Unpaired Translation (CUT) [7]. CUT is a model that transforms images from one visual domain to another in situations where corresponding image pairs are not available. It compares local patches from the input image and the desired output, encouraging the model to retain important structural details while translating the image to a different domain. CUT's architecture is composed of three main components that work together to translate images from one domain to another. First, the *Generator* takes an input image and produces a translated version that reflects the style of the target domain. Next, a different *Feature Extractor Network* is used to pull out meaningful features from both the original image and the generated one. These features are then compared using a *PatchNCE* loss, which operates on small patches from each image to ensure that the content and structure of the input are preserved in the translation. Lastly, the *Discriminator* evaluates the translated image and tries to determine whether it looks like a real image from the target domain. This encourages the generator to produce outputs that are visually realistic.

In our project we fed CUT with unpaired GTA5 and Cityscapes images and we obtained a dataset with GTA5

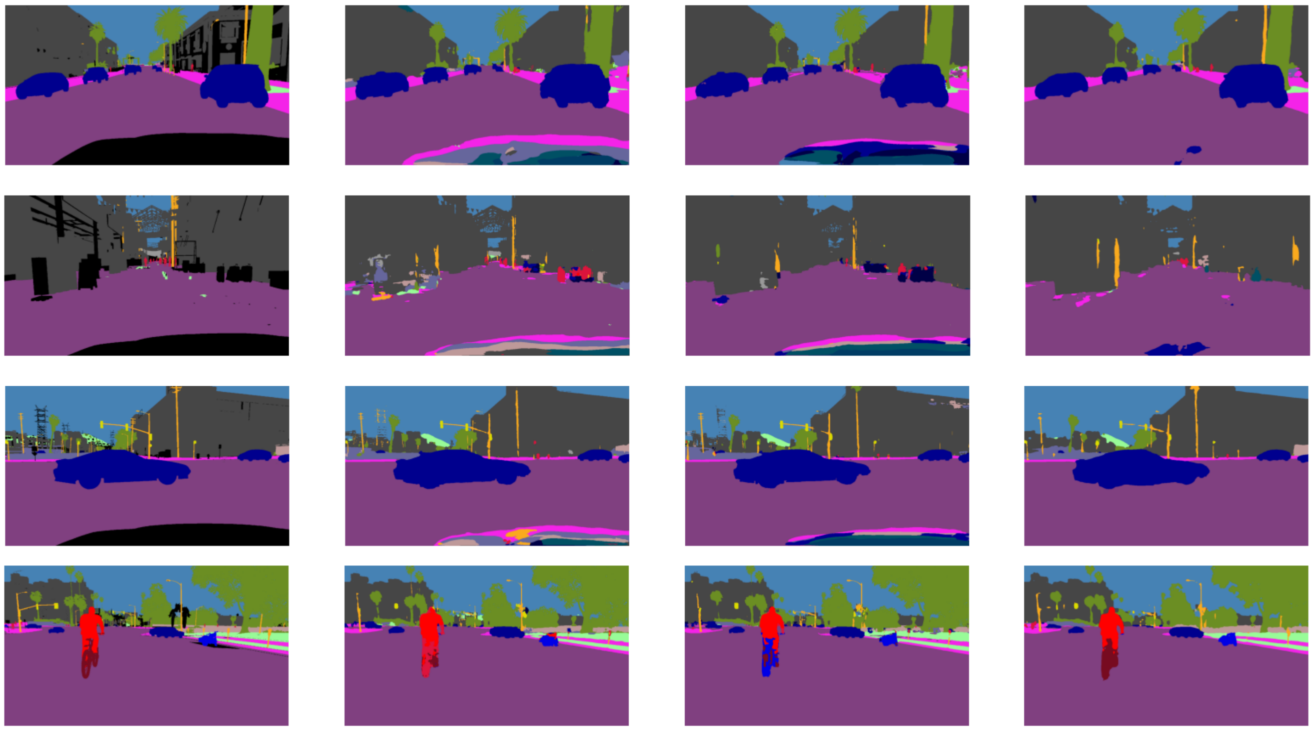

Figure 6.1.2. Ground Truth ResUNet18 ResUNet34 ResNet18

images whose color palette, lighting and textures resemble Cityscapes. This new translated dataset was used for training our segmentation network BiSeNet 3.3 in the same way as explained in previous sections. We have also tried another approach based on CUT, named *FastCUT*. This is a faster version of CUT that, instead of leveraging a new network for doing feature extraction, uses the Generator itself. We were expecting to find a lower training time, and slightly worse results in terms of mIoU. However, the performances were almost identical to CUT, thus we reported only the results obtained with CUT. Table 6 shows the comparison between the baseline model (trained on GTA5 and validated on Cityscapes) and the model trained with the translated GTA5 dataset resulting from CUT. We can see that the second one performs poorly compared to the baseline model, reaching a mIoU equal to 10.29%. The reason behind this low performance is probably due to the fact that the loss function is patch-based, that means it compares only small portions of the images. Similarly to what happens with RandomCrop (introduced in section 4.1), CUT often leads to the omission of several classes during translation, as the generated images tend to preserve only a subset of the original content. Therefore, when these images are forwarded to BiSeNet, the feature extractor sees less samples from certain classes. Additionally, these translated images are very small and have a poor resolution, thus small objects (like traffic lights, pedestrians, lane markings) become blurred or lost entirely.

6.3. Class imbalance

The datasets used so far exhibit strong class imbalance, where common classes such as ”road” or ”building” dominate the pixel distribution, while classes like ”pole” or ”rider” are extremely underrepresented. This imbalance can bias the learning process, leading to poor generalization on rare classes. To mitigate this issues, we implement a class-aware sampling strategy that adjusts the training data distribution at the image level. Our strategy is applied to the GTA5 dataset before training and it consists of three parts.

1. **Class Frequency Estimation**: we start by counting how many pixels belong to each class across the whole training dataset. This tells us how common or rare each class is overall.
2. **Per-Image Class Mapping**: for every image, we check which classes appear in it.
3. **Weighted Sampling Strategy**: for each picture, we take the highest class score, i.e. the rarest class in it. Using that score the model decides how likely that image is to be picked for training: images with rare classes will get sampled more often and images with only common classes will be sampled less.

Technique	**mIoU**	road	sidewalk	building	wall	fence	pole	tl	ts	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bike
Baseline	17.9	43.56	7.16	59.31	9.39	15.50	0.01	0.63	0.13	72.69	5.07	62.31	20.02	1.11	52.17	1.71	3.25	0.02	0.61	0.00
CUT	10.29	70.7	3.04	44.79	2.17	0.02	0.62	0.35	0.01	33.98	5.25	19.78	0.03	0.08	12.51	1.02	1.23	0.00	0.00	0.00

Table 6. Per-class IoU and mIoU (%) comparison between baseline model and CUT

Technique	**mIoU**	road	sidewalk	building	wall	fence	pole	tl	ts	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bike
Baseline	17.9	43.56	7.16	59.31	9.39	15.50	0.01	0.63	0.13	72.69	5.07	62.31	20.02	1.11	52.17	1.71	3.25	0.02	0.61	0.00
Weighted Sampler	20.38	56.53	6.69	66.33	5.81	5.67	0.09	1.13	0.39	74.09	5.62	69.86	25.97	1.94	54.70	2.29	0.87	1.00	6.48	1.09

Table 7. Per-class IoU and mIoU (%) comparison between baseline model and model with weighted sampler applied

Table 7 shows that this method improves performance on rare classes compared to the baseline model.

7. Conclusions and future works

In conclusion, this work addresses the domain shift problem by first introducing a straightforward approach based on data augmentation techniques, then progressing towards more advanced solutions such as adversarial learning. Additionally, we suggest several possible extensions to further mitigate the domain gap when working with synthetic datasets. These include addressing class imbalance in the data, experimenting with style transfer methods and exploring the use of architectures like U-Net. Such directions could provide a valuable starting point for improving performance in real-world applications.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, 2017.

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[4] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015.

[5] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation, 2015.

[6] Philipp Fischer Olaf Ronneberger and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany*, 2015.

[7] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation, 2020.

[8] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games, 2016.

[9] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker. Learning to adapt structured output space for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] Zhengwei Wang, Qi She, and Tomás E. Ward. Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Comput. Surv.*, 54(2), Feb. 2021.

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

[12] Sicheng Zhao, Xiangyu Yue, Shanghang Zhang, Bo Li, Han Zhao, Bichen Wu, Ravi Krishna, Joseph E. Gonzalez, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia, and Kurt Keutzer. A review of single-source deep unsupervised visual domain adaptation, 2020.

[13] zhoudaxia233. Pytorch-unet. `https://github.com/zhoudaxia233/PyTorch-Unet/blob/master/resnet_unet.py`, 2019.