Master Degree in Artificial Intelligence

Deep Learning & Computer Vision Project

Segmentation on GTA-V and MMD Domain Adaptation to Cityscapes

Academic Year: 2024/2025

Enrico Loda - VR522872
Luca Panariello - VR518122

University of Verona
Department of Computer Science

Contents

1 Introduction

1.1 Motivation and rationale

Semantic segmentation is an artificial intelligence task that categorises each pixel in an image into a specific class. It is a type of **supervised multi-class classification** task.

This task is important and relevant today, with many applications including autonomous driving and tumour detection. However, the difficult part is not the training, but annotating the ground truth.

Annotating images with pixel-wise semantic labels is an extremely time-consuming task that requires a huge amount of human effort. It is often impractical for large datasets, which are often required for better model generalisation. This phenomenon is called the **Curse of Annotation** [1].

To overcome this, we can train a model on an easier-to-annotate **synthetic** dataset and use an **unsupervised domain adaptation** technique to adapt the model to a harder-to-annotate **real** dataset [2].

1.2 Data Provenance

The datasets used for the task are:

- **GTA-V Dataset** [2] consists of frames from the eponymous video game, which has a realistic, open-world urban setting that resembles real-world environments. The annotations are obtained in a semi-automatic manner. This will be used as a **synthetic** dataset.
- **Cityscapes** [3] consists of urban street scenes primarily from German cities. This will be used as a **real** dataset.

The GTA-V dataset is easier to annotate since the environment is a video game that can be controlled and modified. By modifying meshes and rendering, it is possible to automatically annotate the selected frames [1] [2].

1.3 State of the Art

There are many models that can perform semantic segmentation; one example is the **SegFormer** [4], which has an encoder-decoder structure: the encoder is a **Transformer** and the decoder is an **MLP**.

Another is the **High Resolution Network** [5], which uses CNNs to perform downsampling and upsampling in parallel. The parallel sub-networks exchange information, maintaining detail throughout the network and producing a high-resolution image.

Although this network was created for **pose estimation**, studies have achieved good results in semantic segmentation too [6].

For unsupervised domain adaptation in semantic segmentation, many models could be used; one of the best architectures are the **Visual Foundation Models**, which are flexible networks, such as **visual transformers**, that are trained on a multitude of massive, diverse datasets in order to learn general image features, effectively creating a base model that can be fine-tuned or adapted for any task.

Various advanced techniques are paired with Visual Foundation Models; one example is **VFM-UDA++** [7]. It combines minimising the distance between feature embeddings produced by the student and teacher models, and pseudo-labelling, which is a self-supervised method. This method attains performance remarkably close to fully-supervised learning in the same adaptation from GTA-V to Cityscapes.

1.4 Objectives

This project is divided into two phases:

1. **Segmentation on Synthetic**: First, two models were fine-tuned to perform semantic segmentation on the **synthetic dataset**: **DeepLabV3** and **U-Net**. The impact of input sizes and initial weights has also been studied.

2. **Adaptation on Real**: After selecting the best **DeepLabV3** and **U-Net** models, these can be adapted to segment images from the **real dataset** using an **unsupervised domain adaptation** technique. The chosen technique is **Maximum Mean Discrepancy**.

Mean Intersection over Union (mIoU) has been used to measure the quality of the models.

The objective is to determine whether completely **unsupervised domain adaptation** can achieve satisfactory results. For this reason, only the **synthetic dataset** will be used in the first phase, despite the common practice of mixing real and synthetic images prior to training [2]. The aim is to achieve at least 60% mIoU on the **test set**, which is the standard for simple fine-tuned models.

In the second part, the best models from the first part are selected using the **validation set**. These models are then trained using images and labels from the synthetic dataset, in the same way as in the first phase. This ensures that the model retains the segmentation task when adapting to the real dataset **without forgetting**.

The adaptation will use images from both datasets for training, but the real dataset labels can be discarded since an unsupervised approach was chosen. However, for comparison purposes, they will be used to select the best model through validation to determine whether a completely unsupervised training approach would produce results comparable to those of a supervised approach.

In this phase, the goal will be to achieve at least half of the previous threshold (30% mIoU), since **MMD** is not an advanced DA technique.

2 Methodology

This project was written in **Python**, and the deep neural networks rely on **torch** frameworks, also the following libraries have been used: torch, torchvision, numpy, cityscapesscripts, backbones_unet, matplotlib, tqdm.

2.1 Dataset Analysis

Both datasets contain RGB images and implement the same 60/10/30 split for train, validation and test sets. The segmentation process considered a total of 19 different classes (Figure 1).

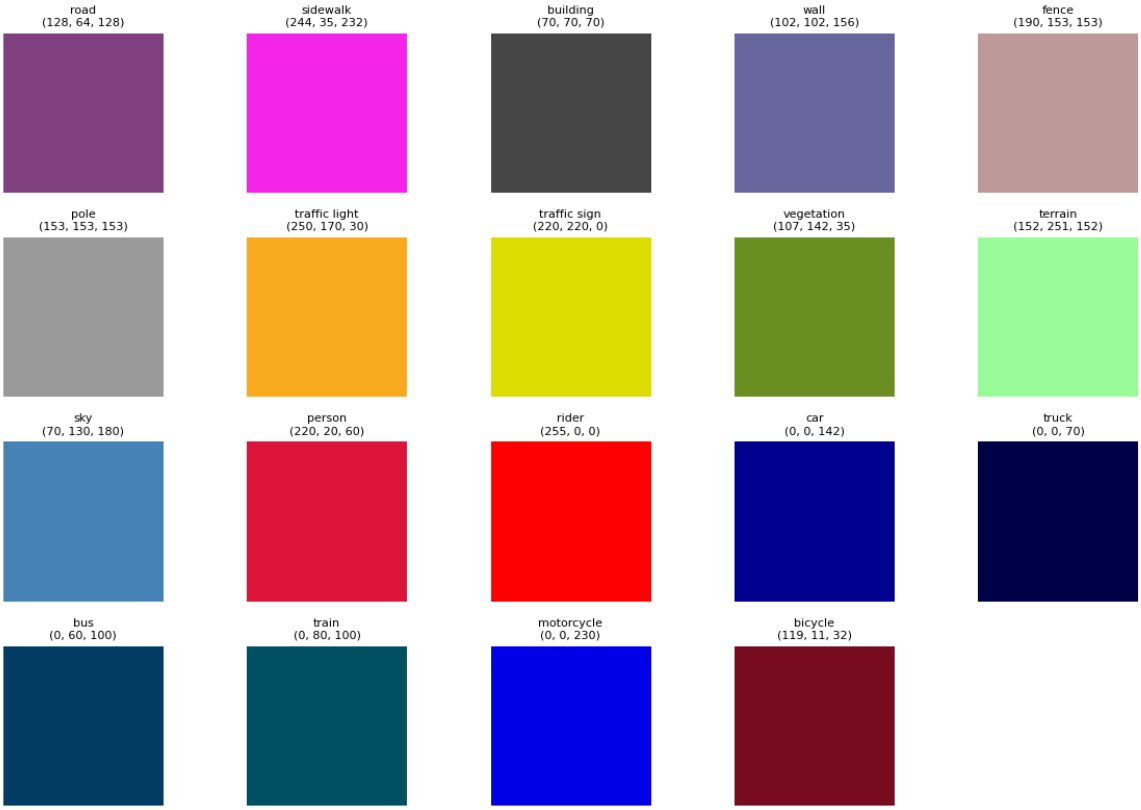

Figure 1: Labels with corresponding colours.

The images were resized to **466x256** and normalized using ImageNet1K parameters. These dimensions were chosen to match the original aspect ratio of the synthetic dataset 2.1.1. Resizing the labels was implemented using **Nearest Neighbour Interpolation** to ensure the validity of all pixels iD.

There is also an alternative architecture that requires an input resolution of **480x256**; more information can be found in the section on architectures 2.2. All architectures were tested on both aspect ratios.

2.1.1 Synthetic dataset

The entire dataset comprises **24,966** densely labelled frames from GTA-V. Each frame has an original resolution of **1914×1052** pixels.

A semi-automated pipeline was used to generate pixel-accurate semantic labels. To achieve this, the rendering pipeline and resource management were exploited to propagate labels efficiently.

On average, it took 7 seconds to annotate each image in this dataset, which is 771 times faster than the per-image annotation time for Cityscapes [2].

For this project, only **12,500** images were used for training to reduce the time required for each epoch.

2.1.2 Cityscapes

The dataset consists of **5,000** finely annotated images with an original resolution of **2,048×1,024**. There are a total of 30 different classes, but only 19 were considered; the rest were labelled as background (id 255). This was done to align with the labels of the synthetic dataset.

All 5,000 fine-annotated images were used for domain adaptation.

2.2 Architectures

All of the architectures implement a pre-trained **ResNet50** as their backbone.

2.2.1 U-Net

The **U-Net** architecture is one of the most popularly implemented in semantic segmentation tasks. It consists of a U-shaped **encoder-decoder** network with a symmetric structure: a contracting path (encoder) to capture context, and an expanding path (decoder) to enable precise localisation [8].

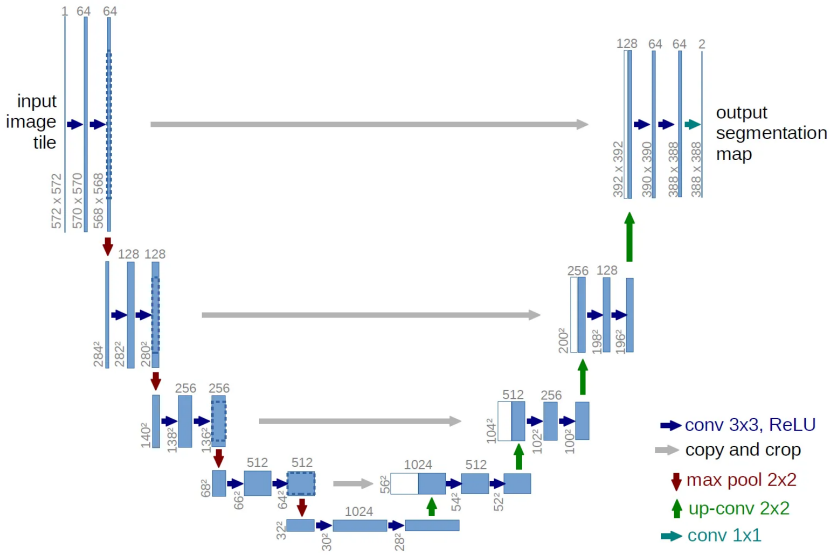

Figure 2: Base U-Net architecture [8]

In this project, the **U-Net**:

- Implements five encoder and decoder blocks, resulting in six total depth levels;
- Does not use a 'copy and crop' connection for the embeddings at the same depth level from the encoder to the decoder; these connections are replaced with normal skip connections, assuming matching dimensions at the same depth level.

With this configuration, an issue arose with the 466×256 resolution since 466 is not divisible by 32, leading to a **mismatch** in dimensions between the encoder and decoder at the same depth level. To overcome this, a **custom U-Net (Padded U-Net)** was implemented to enforce the same dimensions by padding the embeddings before concatenation.

Different variations were implemented to determine the impact of these paddings:

- **Padded U-Net** with a backbone pre-trained on ImageNet1K;
- **U-Net** with a backbone pre-trained on ImageNet1K, with input images sized **480×256**.

Size 480 was chosen because it is the number closest to 466 that is divisible by 32.

2.2.2 DeeplabV3

DeepLabV3 is a **Fully Convolutional Neural Network (FCN)** designed for semantic segmentation. It consists of a ResNet-50 (or ResNet-101) backbone that has been pre-trained

using either the **ImageNet** or **COCO** datasets. In the deeper layers, standard convolutions are replaced with **atrous (dilated) convolutions**. An **Atrous Spatial Pyramid Pooling (ASPP)** module is then added on top of the backbone to capture multi-scale contextual information. This is followed by an upsampling step that restores the input image resolution.

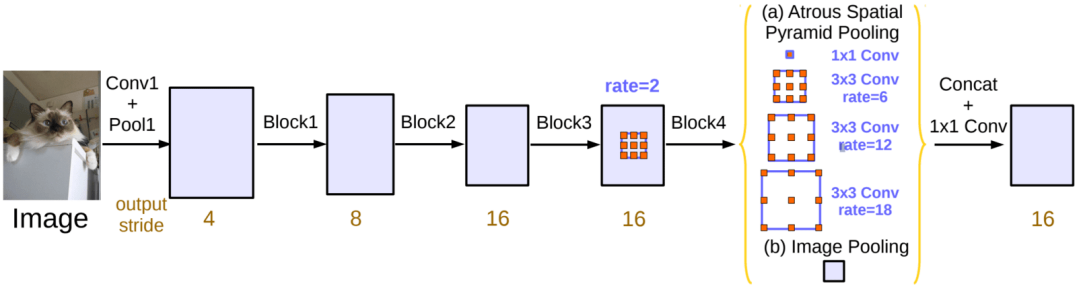

Figure 3: General Deeplab architecture [9]

The core concept of DeeplabV3 are **Atrous convolutions** and **Atrous Spatial Pyramid Pooling**:

- **Atrous convolutions** are a generalisation of standard convolutions, where the kernel is applied with 'holes' (controlled by the dilation rate). This allows the receptive field to be expanded without reducing the resolution of the feature map. .

  In this project, **Atrous convolutions** are applied to the final two blocks of the ResNet backbone, replacing strided convolutions.

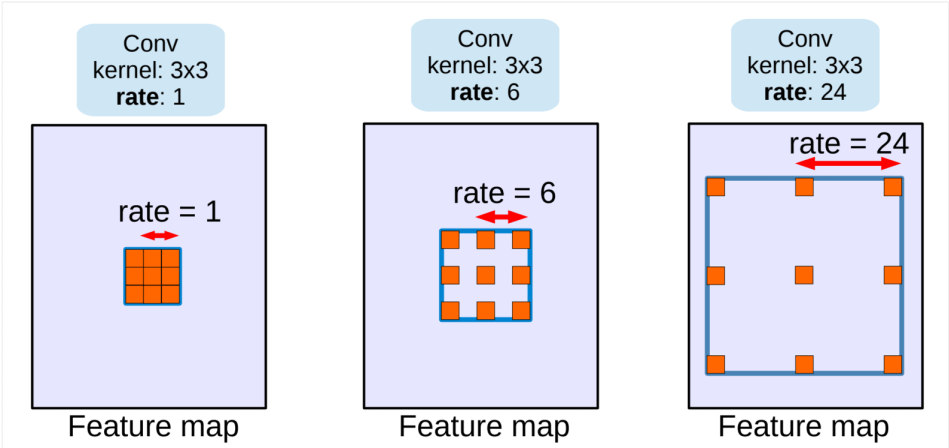

Figure 4: Atrous convolution [9].

- **Atrous Spatial Pyramid Pooling** consists of four Atrous convolutions with different dilation rates, plus one image pooling branch, all of which are performed in parallel (Figure 3).

  The outputs of the five branches are then concatenated to allow the network to capture multi-scale context before projection.

Different variations were implemented in this project:

- Backbone pretrained on **COCO**;
- Backbone pretrained on **ImageNet1K**;
- Backbone pretrained on **ImageNet1K**, input images of size **480x256** (for comparison with the U-Net).

2.3 Loss Functions and Metrics

The main metric used for evaluation is the **Mean Intersection over Union** (mIoU). It measures the proportion of correctly classified pixels out of the total number of pixels for each class. This value ranges from 0 to 1.

2.3.1 Segmentation Loss

Since semantic segmentation is a **multi-class classification** and the output of the networks is a vector of probabilities, a natural choice for the loss is the **Cross-Entropy** $\mathcal{L}_{\text{CE}}$.

However, due to the nature of the dataset, class pixels are not evenly distributed. Objects such as bicycles, people and traffic signals are far less prevalent than roads, trees and buildings in an urban setting. There is a risk that the model will ignore these classifications since a small percentage of misclassified pixels would not influence the cross-entropy value. **Dice Loss** $\mathcal{L}_{\text{Dice}}$ has therefore been added to penalise this behaviour.

The dice loss is a distance that takes inspiration from the mIoU, and it is defined as follows:

$$\mathcal{L}_{\text{Dice}}(\mathbf{p}, \mathbf{g}) = 1 - \sum_{c=1}^{C} \frac{L_c(\mathbf{p}, \mathbf{g})}{C}$$

$$L_c(\mathbf{p}, \mathbf{g}) = \frac{2 \sum_i^N p_i \delta_{g_i,c} + \epsilon}{\sum_i^N p_i + \sum_i^N \delta_{g_i,c} + \epsilon}$$

For each pixel i:

- p_i is the class probability prediction and g_i is the ground truth label.

- The **Kronecker delta** function $\delta_{g_i,c}$ is 1 if the pixel i belongs to class c ($g_i = c$), and 0 otherwise.
- The **smooth parameter** ϵ, avoids division by 0 and ensures non-zero loss return.
- N represent the number of pixels.
- C represent the number of classes.

The final loss for the segmentation task is therefore:

$$\mathcal{L}_{\text{Seg}}(\mathbf{p}, \mathbf{g}) = (1 - \lambda_{\text{Dice}})\mathcal{L}_{\text{CE}}(\mathbf{p}, \mathbf{g}) + \lambda_{\text{Dice}}\mathcal{L}_{\text{Dice}}(\mathbf{p}, \mathbf{g})$$

The parameter λ_{Dice} controls the strength of the regularization. Due to the time-consuming training process, this parameter was fixed at 0.3. An ablation study could be conducted to assess the impact of this parameter.

Some studies have shown that this addition helps stabilise the training process [10].

2.3.2 Domain Adaptation Loss

The idea behind **Maximum Mean Discrepancy Domain Adaptation** is to introduce a loss function that measures the distance between the feature embeddings of synthetic and real data. By minimising this loss, the model is trained to produce the same embeddings, leading to similar segmentation in an unsupervised manner, as the ground truth of the target dataset is not required.

The original study of this type of domain adaptation [11] choose a **CNN** as model, where the deeper layers are Fully Connected layer and contain the most tailored features for the task. For this reason, the authors chose to minimise the loss of the embeddings of these layers.

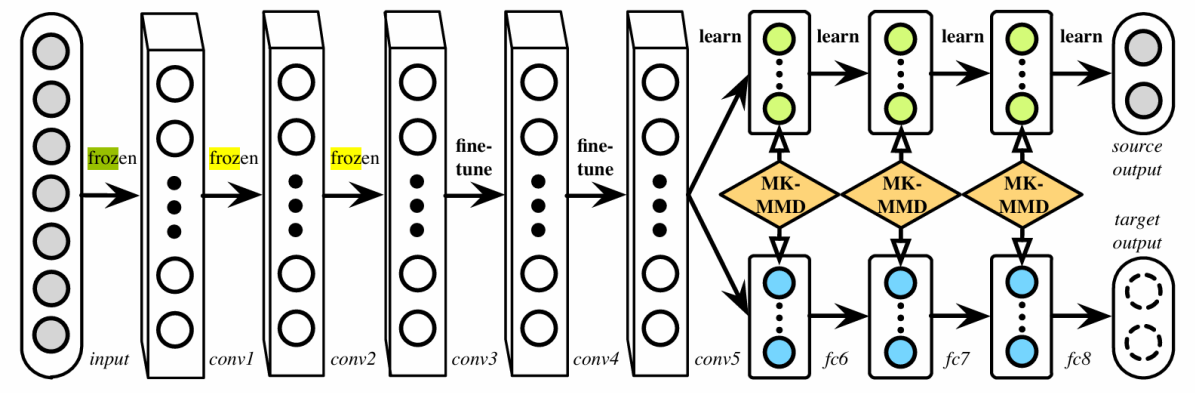

Figure 5: MMD Domain Adaptation. Image from [11]

In this case, the models used are **DeepLabV3** and **U-Net**. The former is a **Fully Convolutional Network** while the latter has an **encoder-decoder** architecture. Neither has FC layers.

The embeddings taken into consideration are those of the **backbones**, which are the most informative layer as they contain the most compacted features of the architecture, making them the most specific to the task in question.

Ideally, the MMD loss would be computed on the embeddings of the backbones and adjacent layers, but for comparison sake only the backbones were considered.

The MMD loss is defined as follows:

$$\mathcal{L}_{\text{MMD}}(\mathbf{x}, \mathbf{y}) = \frac{1}{n^2} \sum_{i,j} k(x_i, x_j) - 2 \sum_{i,j} k(x_i, y_j) + \sum_{i,j} k(y_i, y_j)$$

where $\mathbf{x}$ and $\mathbf{y}$ are the embeddings, and $k(\cdot, \cdot)$ is the kernel function used to project the embeddings into a higher-dimensional space due to the effect of the **kernel trick**. The kernel function used for this loss is the **Gaussian Radial Basis Function**, which is a common choice if no particular characteristic is known about the feature:

$$k(x, y) = \frac{-\|x - y\|_2^2}{2\sigma^2}$$

In this case, the value of σ can be modified to improve feature casting. However, for the aforementioned reason, this parameter will be fixed at 1.

Ultimately, the MMD loss was integrated with the segmentation loss, acting as a regularization term that penalises the loss if the embeddings are not close:

$$\mathcal{L} = \mathcal{L}_{\text{Seg}}(\mathbf{p}, \mathbf{g}) + \lambda_{\text{MMD}} \mathcal{L}_{\text{MMD}}(\mathbf{s}, \mathbf{r})$$

Here $\mathbf{s}$ are the embeddings of the synthetic images computed by the backbones and $\mathbf{r}$ are the embeddings of the real images, also computed by the backbones. The parameter λ_{MMD} controls the strength of the MMD loss; different values of this parameter were tested in an ablation study.

A multi-level MMD has also been implemented in DeeplabV3 by applying the MMD loss to the ASPP block output to align the multi-scale contextual features. The new version of the loss is as follows:

$$\mathcal{L} = \mathcal{L}_{\text{Seg}}(\mathbf{p}, \mathbf{g}) + \lambda_{\text{MMD}} \frac{\mathcal{L}_{\text{MMD}}(\mathbf{s}_B, \mathbf{r}_B) + \mathcal{L}_{\text{MMD}}(\mathbf{s}_D, \mathbf{r}_D)}{2}$$

Where $\mathbf{s}_B, \mathbf{r}_B$ are the embeddings of the backbone, and $\mathbf{s}_D, \mathbf{r}_D$ are the embeddings of the ASPP block.

3 Training

ADAM was choose as the **optimizer**, this is a common choice in the literature. The **learning rate** was set to `1-e4` for both task, which provides a good compromise between convergence and descent steepness. No **scheduler** has been applied to the learning rate.

Even though these parameters are crucial for training, no ablation studies have been conducted on the learning rate or any other optimiser parameters in this project due to time constraints.

3.1 Segmentation

The dataset was loaded into DataLoaders with a batch size of 8, which was deemed to be a good compromise between update frequency and memory storage. **Shuffle** was set to `true` in order to avoid biases. The number of training **epochs** chosen for this task was 25 and and the best models were selected based on the highest mIoU in the validation set.

The aim of this task was to compare different architectures with multiple input resolutions that employed backbones pre-trained on various datasets. To ensure a fair comparison, the architectures were trained with as few differences as possible.

3.2 Domain adaptation

DataLoaders with a batch size of 4 were deployed: one for the GTA-V dataset and one for the Cityscape dataset. This led to a total of 8 images at each step.

Shuffle was set to `true` in order to avoid biases since the dataset is ordered by frames and cities. The number of **epochs** chosen was reduced to 10 in order to perform an ablation study on λ_{MMD} initially with values: 0.2, 0.1, 0.05, 0.01.

Following the adaptation of the first DeeplabV3, 0.05 and 0.01 were selected as the most appropriate values for the remaining models.

For each variation two different *best* models were saved: one based on the highest mIoU on the **source** validation, and one based on the highest mIoU on the **target** validation. Although the implemented method is unsupervised, this approach was chosen to validate the best possible outcomes given this project's constraints.

3.3 Hardware Specifications

All the training scripts were run on a M4 MAC MINI with the following specifications:

- **CPU**: M4 10-core
- **GPU**: M4 10-core
- **Unified Memory**: 16 GB

- **OS**: macOS Sequoia
- **Python version**: 3.11.9

Training the models locally was preferred, and this machine was chosen for its energy efficiency and unified memory architecture. It is also important to note that the backend implemented for training was MPS and not CUDA due to the Apple silicon architecture.

Due to the limitations of the machine, training sessions could last for hours. For this reason, **checkpoints** were employed to manage time and resources. The training process is usually divided into sessions of five epochs.

4 Inference

During the inference and visualisation phase, two functions were applied to render the segmentation results: the **softmax** function was applied since the model outputs were logits and not probability distributions. The **max** function was applied to retrieve the class in which the model was most confident.

4.1 Background Classification

A small test was also performed. Since some labels were removed from the dataset and considered background, the model was not trained to classify those pixels. Therefore, their confidence should be low.

The goal of the test is to assess the effectiveness of the models in understanding areas for which they have not been trained, by measuring their level of uncertainty. To achieve this, a combined approach was employed that considered both **entropy** and the **confidence of the predictions**.

Entropy measures the amount of information that a distribution contains. If the entropy is high, then the distribution contains a lot of information and resembles a spike or constant distribution (for example, a one-encode distribution). On the other hand, a distribution with no information would be flat (for example, the uniform distribution). The formula applied was as follows:

$$H(\mathbf{p}) = \sum_{i=1}^{19} p_i \log(p_i + \epsilon)$$

ϵ is a small number to avoid log of zero. After measuring the entropy for each pixel, this value was compared with a threshold value $\xi_{\text{entropy}} = 1$. If the entropy was higher, the pixel was deemed informative and shown correctly in the visualisation; otherwise, the pixel was blacked out.

The **confidence of the predictions** is simply a threshold on the maximum probability of the softmax of the logits. This threshold was set to $\xi_{\text{confidence}} = 0.5$, meaning that all pixels with a probability of belonging to a class lower than 50% were blacked out.

The final approach combined these two information, leading to blacking out all the pixels with an **entropy** too high or **confidence of the predictions** too low.

Important note: due to the simplicity of this method, this test has only been implemented in the visualisation phase to assess the quality of the results obtained. Neither loss nor metrics take this formulation into account.

5 Experiments & Results

This section presents the results of the different architectural variations implemented in terms of mIoU in table form. Graphs illustrating losses and mIoU during training are provided for the variation that achieved the highest mIoU in the validation set.

5.1 Segmentation on Synthetic Dataset

5.1.1 DeepLabV3

	COCO	**Imagenet1k**	**480x256**
Train mIoU	0.6684	0.6815	0.6840
Validation mIoU	0.5830	0.5857	0.5857
Test mIoU	0.5900	0.5892	0.5448

Table 1: Loss and Metric Comparison in DeepLabV3 model

All variations of this architecture were trained for 25 epochs, with each epoch taking an average of $\sim$ 47 minutes. This totalled to $\sim$ 62 hours of training. It is important to note that this time was inflated due to the longer sessions of 15 epochs at the beginning of the project. These longer sessions resulted in some epochs taking up to 1 hour to complete due to some memory leakage problems in the MPS backend. By contrast, each epoch in the smaller sessions of 5 epochs took around $\sim$ 38 minutes.

As can be seen from Table 1, the **480x256 DeepLabV3** model with backbone pretrained on ImageNet1K achieves the best results during training, despite performing worst on the test set. However, note how small the difference is between this model and the base **DeeplabV3** with backbone pretrained on ImageNet1K, which makes it clear that the small increase in input size did not impact performance during validation.

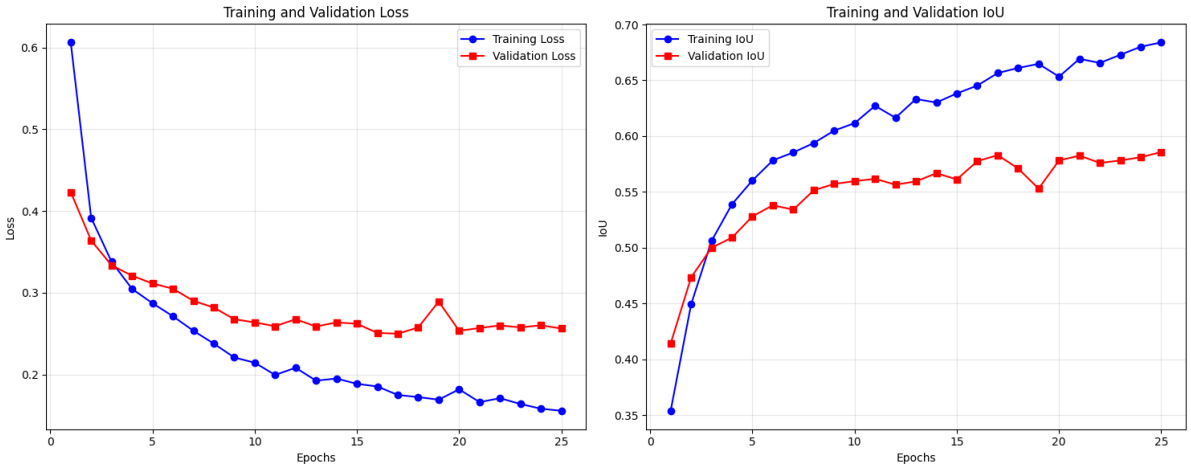

Figure 6: Loss and mIoU trend of the 480x256 DeepLabV3 model

It is also evident from Figure 6 that the loss and mIoU of the **480x256 DeepLabV3** model demonstrate consistent improvement throughout training. It is reasonable to assume that training the model for additional epochs would yield even better results.

The **COCO DeepLabV3** model and the **ImageNet1K DeepLabV3** metrics show almost identical trends, revealing that the choice of weight leads to only a small change in the final results (see Table 1), with no significant difference in **convergence speed**.

5.1.2 U-Net

	Padded U-Net	**480x256 U-Net**
Train mIoU	0.6952	0.7035
Validation mIoU	0.5690	0.5983
Test mIoU	0.5760	0.6040

Table 2: Loss and Metric Comparison in U-Net Model

Both variations of this architecture were trained for 25 epochs, with an average epoch taking $\sim$ 17 minutes each, totalling to $\sim$ 14 hours of training.

As can be seen from Table 2, the **480x256 U-Net** model achieved the best results during training. Unlike the DeeplabV3 model, there was a visible improvement in the validation set, probably due to the simpler encoder-decoder architecture and the removal of the padding in the skip connections.

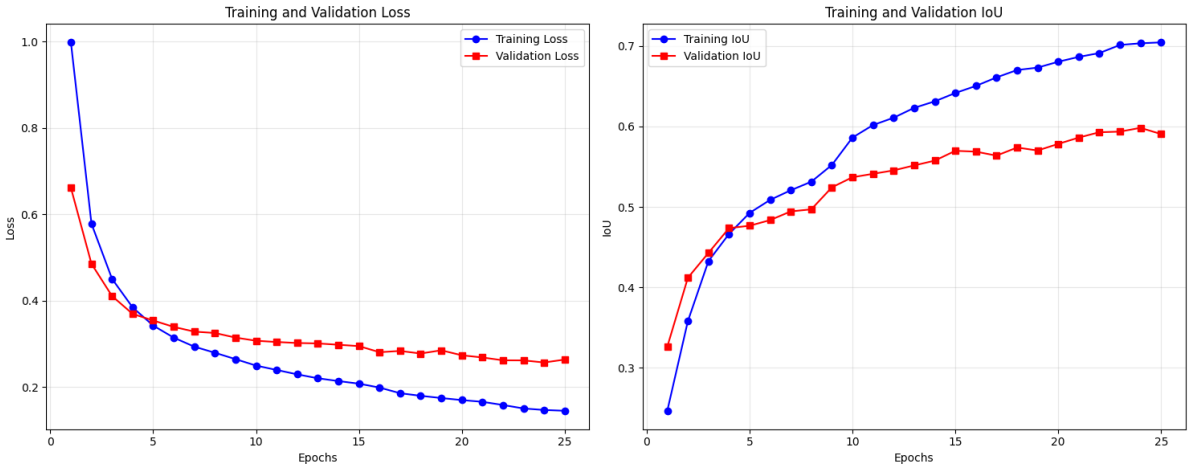

Figure 7: Loss and mIoU trend of the 480x256 U-Net model

The metrics are constantly improving in this case, too. It shows greater stability than the **DeepLabV3** trend. As before, increasing the number of epochs could lead to an even better outcome.

5.1.3 Segmentation Comparison

	DeepLabV3 480x256	**U-Net 480x256**
Train mIoU	0.6840	0.7035
Validation mIoU	0.5857	0.5983
Test mIoU	0.5448	0.6040
Train Time (hours)	16.65[a]	6.76

Table 3: Loss, Metric and Time Comparison between DeepLabV3 and U-Net models

(a) This estimate is based on the 5 epoch session to ensure a fair comparison

As it can be seen in Table 3a, the **U-Net** model outperforms the **DeepLabV3** model in every aspect, despite having a simpler architecture. This is probably due to the small number of epochs and the limited dataset. Given enough training steps, it can be expected that the **DeepLabV3** model would surpass the **U-Net**. Both models will be evaluated in the **domain adaptation** phase.

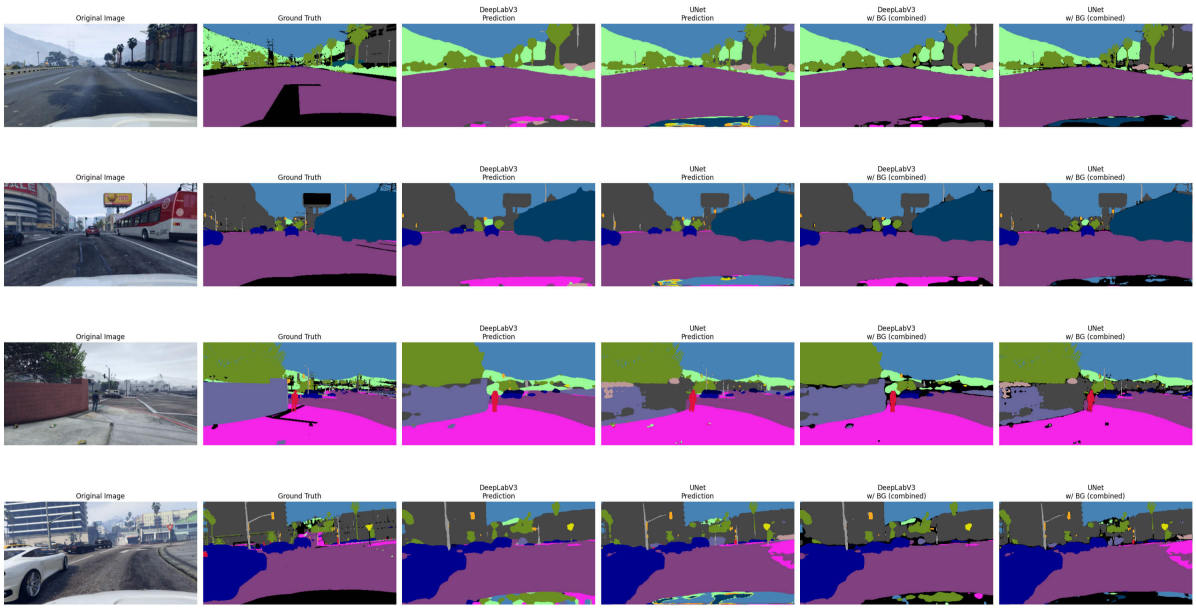

Figure 8: Some examples of segmentation on test set

As Figure 8 shows, the **U-Net** produces a clearer delineation of the **boundaries** of the segmented object, which is probably due to its encoder-decoder architecture with skip connections.

5.2 Domain Adaptation on Real Dataset

The following tables show the results of the domain adaptation using different λ_{MMD} values. The results calculated with the models **before domain adaptation** are shown as $\lambda_{\text{MMD}} = 0$. Graphs illustrating losses and mIoU during training are provided for the λ_{MMD} value that achieved the highest mIoU in the validation sets of the source and the target datasets.

5.2.1 DeepLabV3

λ_{MMD}	0	0.2	0.1	0.05	0.01	$0.05^{(\text{mul})}$	$0.01^{(\text{mul})}$
Source Train mIoU	0.6815	0.6389	0.6502	0.6401	0.6545	0.6508	0.6590
Source Val mIoU	0.5857	0.5131	0.5351	0.5447	0.5474	0.5531	0.5523
Target Val mIoU	0.2693	0.2544	0.2595	0.2918	0.2937	0.2990	0.2940

Table 4: Metric Comparison of multiple λ_{MMD} values for DeepLabV3 model, best results in **source** validation; (mul) refers to the multilevel MMD approach.

$\lambda_{\mathbf{MMD}}$	0	0.2	0.1	0.05	0.01	$0.05^{(\mathbf{mul})}$	$0.01^{(\mathbf{mul})}$
Source Train mIoU	0.6815	0.6302	0.6448	0.6463	0.6570	0.6451	0.6501
Source Val mIoU	0.5857	0.5033	0.5306	0.5337	0.5423	0.5307	0.5375
Target Val mIoU	0.2693	0.2897	0.3010	0.2990	0.3006	0.3093	0.2974

Table 5: Metric Comparison of multiple λ_{MMD} values for DeepLabV3 model, best results in **target** validation; (mul) refers to the multilevel MMD approach.

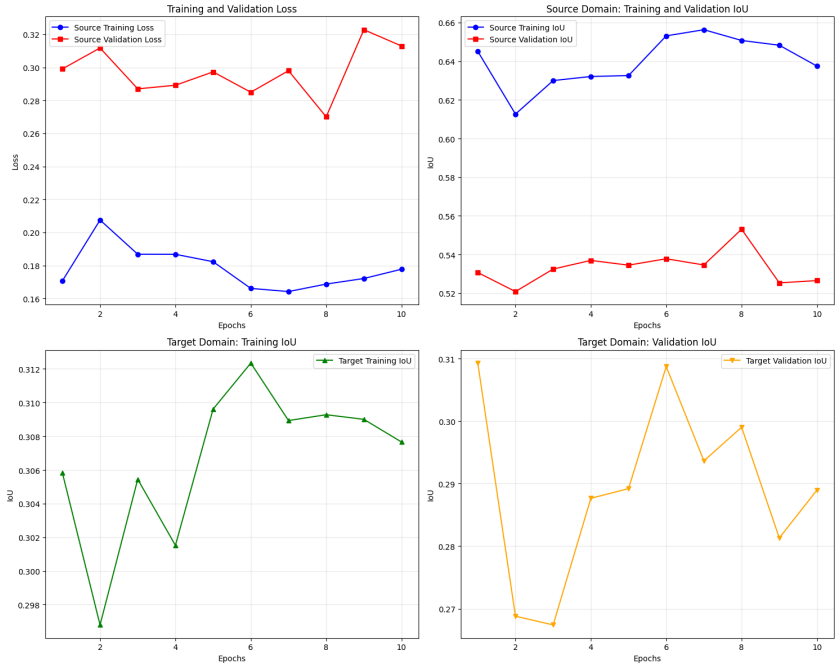

Figure 9: Results of training process of best model in both **source** and **target** validation (multilevel $\lambda_{\text{MMD}} = 0.01$)

As shown in Tables 4 and 5, higher values of λ_{MMD} lead to worse result. This is probably due to the greater amount of perturbation in the loss. For this reason, the multilevel approach and the **U-Net** adaptation were only tested for the two smallest λ_{MMD} values.

The **multilevel approach** results in better performance in both the selection methods, confirming the benefits of also considering the output of the **ASPP** block for the MMD loss.

Figure 9 shows that performance in the source domain is not disrupted too much while achieving moderate, albeit inconsistent, growth in the target domain.

5.2.2 U-Net

	Pre-Adaptation	$\lambda_{\text{MMD}} = 0.05$	$\lambda_{\text{MMD}} = 0.01$
Source Train mIoU	0.7035	0.6679	0.6860
Source Validation mIoU	0.5983	0.5627	0.5644
Target Validation mIoU	0.2793	0.2922	0.2751

Table 6: Metric comparison of λ_{MMD} values for U-Net model, best results in **source** validation

	Pre-Adaptation	$\lambda_{\text{MMD}} = 0.05$	$\lambda_{\text{MMD}} = 0.01$
Source Train mIoU	0.7035	0.6838	0.6842
Source Validation mIoU	0.5983	0.5568	0.5660
Target Validation mIoU	0.2793	0.3063	0.3035

Table 7: Metric comparison of λ_{MMD} values for U-Net model, best results in **target** validation

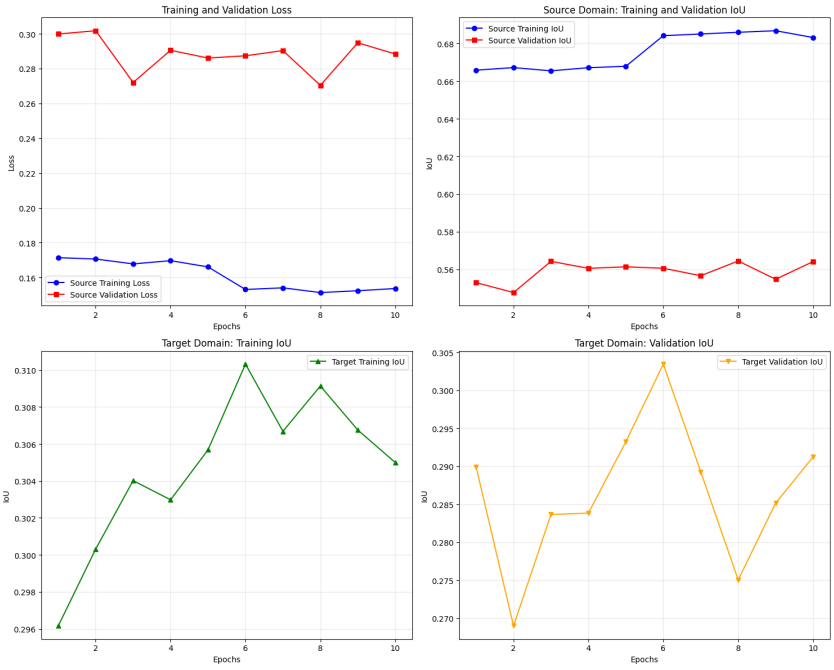

Figure 10: Results of training process of best model in **source** validation ($\lambda_{\text{MMD}} = 0.01$)

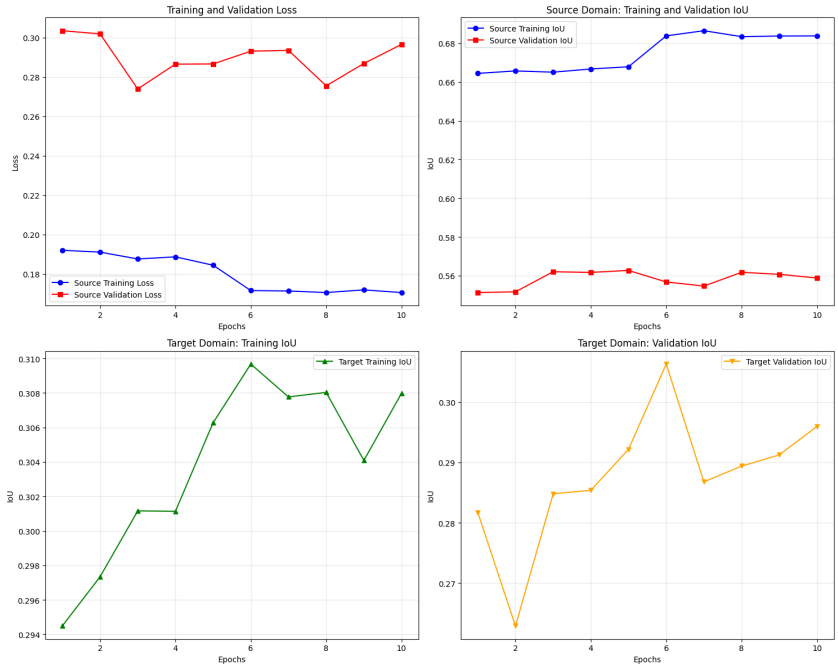

Figure 11: Results of training process of best model in **target** validation ($\lambda_{\text{MMD}} = 0.05$)

Even in this case, the regularization effect of λ_{MMD} on the **segmentation loss** is evident. Using a higher λ_{MMD} increases the mIoU on the target, but decreases the same metric on the source set. As previously mentioned, this is a predictable outcome given the structure of the **MMD** framework. Figures 10 and 11 demonstrate that performance in the source domain remains consistent, whereas the target domain shows a better growth trend compared to **DeepLabV3**.

5.2.3 Domain Adaptation Comparison

	DeepLabV3		**U-Net**	
	(a) best source	(b) best target	(a) best source	(b) best target
	multilevel$\lambda_{\text{MMD}} = 0.05$		$\lambda_{\text{MMD}} = 0.01$	$\lambda_{\text{MMD}} = 0.05$
Source Train mIoU	0.6508	0.6451	0.6860	0.6838
Source Validation mIoU	0.5531	0.5307	0.5644	0.5568
Target Validation mIoU	0.2990	0.3093	0.2751	0.3063
Training Time (hours)	8.1		2.8	

Table 8: Comparison considering the best results in both source and target validation

(a) Best model selected using **source validation** (DeepLabV3 in Table 4 - U-Net in Table 6)

(b) Best model selected using **target validation** (DeepLabV3 in Table 5 - U-Net in Table 7)

Even during the domain adaptation phase, the **U-Net** model outperformed the **DeepLabV3** model. Of the models selected based on their performance in the source validation set, the former achieved better results in all areas except for a lower target validation mIoU.

The same applies to the models chosen based on performance in the target validation set; however, in this case the difference was almost insignificant, as can be seen in 8b. Of the two architectures, **U-Net**showed the greatest improvement in performance between the two selection cases, while the performance of **DeepLabV3** was less affected.

It is important to note that even after the adaptation, the models perform well in the source validation set, as intended by the domain adaptation framework.

The obtained results in the target validation, while not excellent, are satisfactory considering the simplicity and the constrains of the implementation, leading to some decent segmentations.

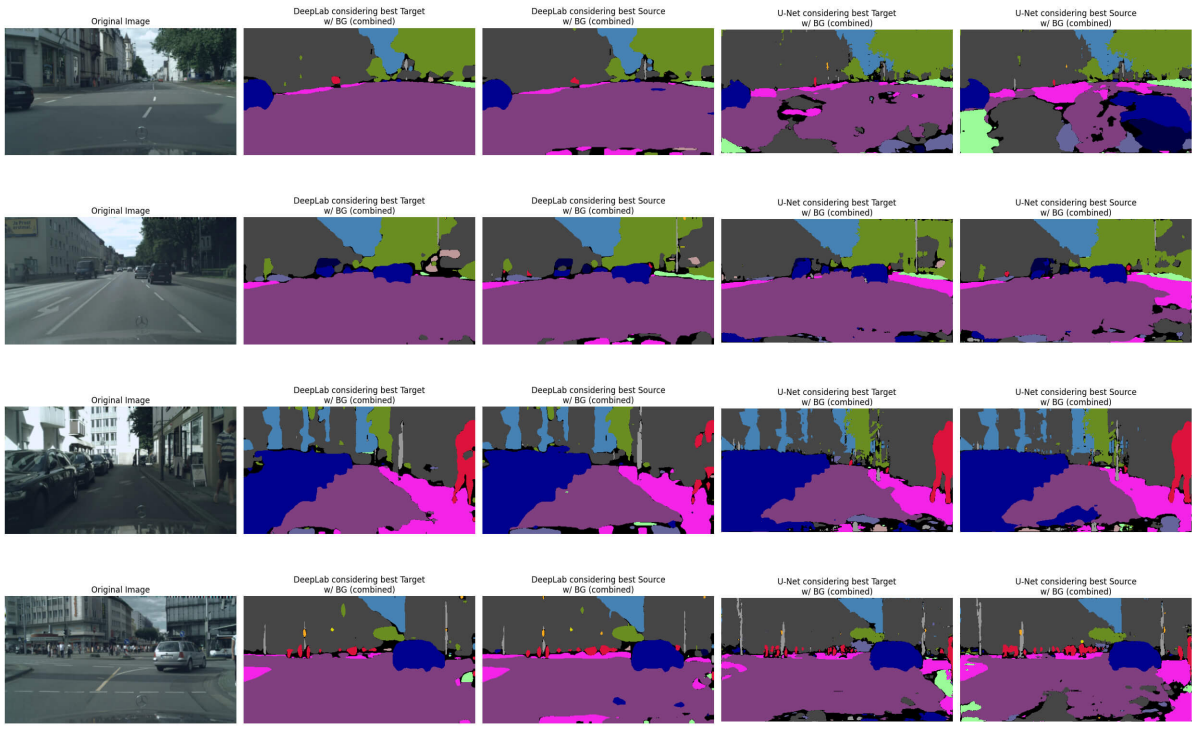

Figure 12: Some examples of segmentation on test set of Cityscapes

The examples in Figure 12 are important for a qualitative evaluation of the results:

- **DeepLabV3** seems to be more coherent with objects that cover most of the image, such as the road and the sidewalk compared to **U-Net**,
- **U-Net** on the other hand seems to produce highly detailed outlines of smaller objects, such as the vegetation and people, even at longer distances; whereas **DeepLabV3** produces coarser, region-based segmentations.

6 Conclusion

The objective of this project was to train different pre-trained deep neural network networks to perform **semantic segmentation** on images from a synthetic environment (**GTA-V**). The trained models were then adapted to perform the same task on images from a real environment (**Cityscapes**) in an unsupervised manner. To achieve this, normalisation and resizing were applied, and several versions of the **U-Net** and **DeepLabV3** architectures were tested.

The initial resolution chosen was **466x256** to maintain the original aspect ratio. However, resizing the images to **480x256** seems to be a valid choice, as both model versions achieve the highest metrics. The **U-Net** probably achieves the greatest performance boost due to the padding drop in the skip connections. **U-Net** is clearly the better architecture in this phase, achieving a higher mIoU in every partition of the dataset in a fraction of the total training time 3a.

As expected, some improvements have been made in the **adaptation phase**, albeit modest ones. This is probably due to the low number of training epochs, the high intensity induced by λ_{MMD}, and the MMD Adaptation itself, since this method was designed for **Feed Forward Networks** with **Fully Connected Layers** [11].

In conclusion all objectives have been met, but there are many improvements that can be made:

- Modification on **U-Net** and **DeepLabV3**, changing the number of filters, layers, kernel size or dilation rate in case of **Atrous convolutions**,
- Experimentations on other range of hyperparameters, like **learning rate**, **momentum** λ_{Dice}, λ_{MMD}; increasing the number of epochs or adding a **scheduler for the learning rate**,
- Implementations of different architecture such as **DeepLabV3+** [12], this in particular would probabily benefit in the quality of the outlines due to its encoder-decoder architecture,
- Implementations of more advanced Domain Adaptation techniques specific for the segmentation task, such as method implementing adversarial networks [13] [14] or State of The Art techinques [7].

References

[1] Jun Xie et al. *Semantic Instance Annotation of Street Scenes by 3D to 2D Label Transfer*. 2016. arXiv: 1511.03240 [cs.CV]. URL: https://arxiv.org/abs/1511.03240.

[2] Stephan R. Richter et al. *Playing for Data: Ground Truth from Computer Games*. Ed. by Bastian Leibe et al. 2016. URL: https://link.springer.com/chapter/10.1007/978-3-319-46475-6_7.

[3] Marius Cordts et al. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. https://www.cityscapes-dataset.com/examples/. Accessed: 2025-08-16. 2016.

[4] Enze Xie et al. *SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers*. 2021. arXiv: 2105.15203 [cs.CV]. URL: https://arxiv.org/abs/2105.15203.

[5] Ke Sun et al. *Deep High-Resolution Representation Learning for Human Pose Estimation*. 2019. arXiv: 1902.09212 [cs.CV]. URL: https://arxiv.org/abs/1902.09212.

[6] Ke Sun et al. *High-Resolution Representations for Labeling Pixels and Regions*. 2019. arXiv: 1904.04514 [cs.CV]. URL: https://arxiv.org/abs/1904.04514.

[7] Brunó B. Englert and Gijs Dubbelman. *VFM-UDA++: Improving Network Architectures and Data Strategies for Unsupervised Domain Adaptive Semantic Segmentation*. 2025. arXiv: 2503.10685 [cs.CV]. URL: https://arxiv.org/abs/2503.10685.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV]. URL: https://arxiv.org/abs/1505.04597.

[9] Liang-Chieh Chen et al. *Rethinking Atrous Convolution for Semantic Image Segmentation*. 2017. arXiv: 1706.05587 [cs.CV]. URL: https://arxiv.org/abs/1706.05587.

[10] Chen Shen et al. *On the influence of Dice loss function in multi-class organ segmentation of abdominal CT using 3D fully convolutional networks*. 2018. arXiv: 1801.05912 [cs.CV]. URL: https://arxiv.org/abs/1801.05912.

[11] Mingsheng Long et al. *Transferable Representation Learning with Deep Adaptation Networks*. 2019. DOI: 10.1109/TPAMI.2018.2868685.

[12] Liang-Chieh Chen et al. *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*. 2018. arXiv: 1802.02611 [cs.CV]. URL: https://arxiv.org/abs/1802.02611.

[13] Munan Ning et al. *Multi-Anchor Active Domain Adaptation for Semantic Segmentation*. 2021. arXiv: 2108.08012 [cs.CV]. URL: https://arxiv.org/abs/2108.08012.

[14] Yi-Hsuan Tsai et al. *Learning to Adapt Structured Output Space for Semantic Segmentation*. 2020. arXiv: 1802.10349 [cs.CV]. URL: https://arxiv.org/abs/1802.10349.