Real-time Domain Adaptation in Semantic Segmentation

Vercellone Romeo
Politecnico di Torino
s341967@studenti.polito.it

Nespolo Elena
Politecnico di Torino
s345176@studenti.polito.it

Mallo Giuseppe
Politecnico di Torino
s346884@studenti.polito.it

Lanzillotta Andrea
Politecnico di Torino
s343438@studenti.polito.it

Abstract

This report explore the field of Semantic Segmentation and studies several papers published in recent years. We begin by making a comparison between two models with distinct design goals: DeepLab, which is focused on enhancing segmentation accuracy through advanced convolutional techniques, and BiSeNet, which aims for high-speed inference while maintaining strong segmentation performance. Then, we focus on real-time semantic segmentation on two benchmark datasets: GTA5 and CityScapes. Creating a dataset of real-world images providing high-quality, pixel-level annotations, like Cityscapes, is highly costly and labor-intesive, whereas GTA5 offers synthetic data that is easier to generate at large scale. Therefore, we believe that the development of new Semantic Segmentation models that are able to shift from synthetic to real-world domains is crucial, and this motivates the use of "Unsupervised Domain Adaptation" (UDA) techniques. We evaluate several methods including Data Augmentation, Image-to-Image Domain Adaptation through "Fourier Domain Adaptation" (FDA) and "Domain Adaptation via Cross-domain Mixed Sampling" (DACS), and "Rare Class Sampling" (RCS). Finally we explore a method to try to improve the model accuracy via a technique called "Context-Aware High-Resolution Domain-Adaptive Semantic Segmentation" (HRDA). The code is available at https://github.com/OTOTOTO02/MLDL2024_project1.

1. Introduction

Semantic Segmentation is a field of computer vision, whose purpose is to assign a semantic label to each pixel of an image or a frame of a video. These algorithms allow computers to understand and interpret the context of visual data. In particular, Semantic Segmentation algorithms can be used for enabling the safe navigation of autonomous vehicles, assisting in medical diagnostics through image analysis, enhancing automation in agriculture, improving scene understanding in photography, among others.

The rise of Deep Learning models [3], corresponding to the introduction of Fully Connected Network and then the Encoder-Decoder architecture, has made Semantic Segmentation computationally feasible and cost-effective. These advancements have led to new state-of-the-art (SOTA) models tailored to specific goals:

- **High accuracy**: Deep Convolutional Neural Networks provides better performances with the respect to classical Machine Learning models, but they still struggle in varying aspects, such as difficulties in preserving fine spatial details and object contours, detecting objects at different scales, and capturing precise spatial relationships between pixels in complex scenes.

- **Real-time performance**: Relying solely on input cropping, channel pruning, or stage dropping leads often to poor performances. Then it is essential to enable high-quality segmentation with low latency for real-time applications.

- **Domain adaptability**: A common strategy is to depend on synthetic data, that are easier to generate and to annotate than real-world datasets. However, models trained on such data usually return low accuracy when applied to real-world images: this phenomenon is known as *domain shift*. Several approaches have been proposed, such as Data Augmentation or techniques such as Fourier Domain Adaptation (FDA) [8], Domain Adaptation via Cross-domain Mixed Sampling (DACS) [7], Rare Class Sampling (RCS) [4].

- **GPU memory consumption**: especially for UDA techniques that relies on both source and target domain, Semantic Segmentation models require a lot of

memory. So a framework like Context-Aware High-Resolution Domain-Adaptive Semantic Segmentation (HRDA) is able to balance good performance with efficient GPU memory usage by selecting both High Resolution and Low Resolution crops.

2. Related Work

The DeepLabV2 model [1] enhances spatial resolution, adapts to varying object scales and finds a trade-off between localization accuracy and classification performance thanks to the use of *atrous convolution*, *Atrous Spatial Pyramid Pooling* and *Conditional Random Fields*.

Instead, BiSeNet [9] introduces a dual-path approach, combining a spatial path and a context path, to achieve fast yet accurate segmentation.

In unsupervised domain adaptation (UDA), models must generalize from a labelled source domain to an unlabelled target domain, as summarized in [10]. Among recent methods, FDA [8] applies the Fourier transform to the source domain images, and replace their low-frequency components with those from the target domain, transferring low-level features such as texture and illumination. Similarly, [7] introduces a mixing strategy in which pseudo-labelled target objects are added to source images, called DACS. The work made by [4] is focused on a Transformer architecture but with innovative training strategies such as Rare Class Sampling. Finally, [5] aimed at reducing GPU memory consumption during training. Their approach combines both high-resolution and low-resolution crops to capture global context and to preserve fine details, resulting in the High-Resolution Domain-Adaptive framework.

3. Methodology

3.1. DeepLab

DeepLabV2 is a convolutional neural network architecture for semantic image segmentation that introduced key innovations to improve dense prediction accuracy, particularly in challenging scenes with varying object scales. Building upon its predecessor DeepLabV1, DeepLabV2 incorporates atrous convolutions to effectively control the resolution of feature maps without increasing computational cost, enabling dense feature extraction at multiple scales. Furthermore, it employs *Atrous Spatial Pyramid Pooling* (ASPP) to capture multi-scale contextual information, and leverages fully connected *Conditional Random Fields* (CRFs) for refined boundary localization. These enhancements make DeepLabV2 a compelling model for evaluating the effectiveness of deep learning techniques in fine-grained scene understanding tasks. In this study, we explore the capabilities of DeepLabV2 analyzing its performance, robustness, and limitations when applied to the Cityscapes dataset.

3.2. Bisenet

While DeepLabV2 achieves high segmentation accuracy through rich contextual modeling, its computational demands limit its applicability in real-time or resource-constrained settings. To address this trade-off between accuracy and speed, BiSeNet was proposed. BiSeNet introduces a dual-branch architecture consisting of a *Spatial Path*, which preserves spatial information, and a *Context Path*, which rapidly encodes semantic context using a lightweight backbone. These branches are fused via a *Feature Fusion Module* (FFM), enabling BiSeNet to balance detailed spatial accuracy with efficient contextual understanding. Designed for real-time semantic segmentation, we will explore how BiSeNet perform both in accuracy and in inference speeds, the so called speed-accuracy trade-off.

3.3. Image-to-Image domain adaptation and Data Augmentation

Building upon the upper bounds established in the previous sections, we now address the challenge of domain shift and evaluate techniques aimed at mitigating its impact. Specifically, we assess the performance of the BiSeNet network trained on the GTA5 dataset and tested directly on the validation split of the Cityscapes dataset, which represents a distinct domain from the training data.
This cross-domain evaluation introduces a significant distribution gap between the source (GTA5) and target (Cityscapes) domains. To bridge this gap, domain adaptation methods are typically employed to enhance the model's generalization to the target domain. In particular, we focus on image-to-image domain adaptation, where the objective is to implement transformations that map input images from the source domain to the target domain's style or distribution, while preserving essential semantic information. This approach allows for reducing the domain discrepancy at the image level, facilitating more effective feature learning and improving downstream task performance.

3.3.1 Domain shift

As a baseline to quantify the effect of domain shift, we adopt a direct transfer setting in which the BiSeNet model is trained exclusively on the source domain and evaluated on the target domain without any form of domain adaptation or data augmentation. This naïve transfer setup represents the lower bound of performance, as the model has not been exposed to the target domain's distribution during training. The resulting degradation in performance illustrates the impact of domain shift and motivates the need for adaptation techniques that can bridge the gap between source and target domains. This baseline serves as a reference point against which the effectiveness of domain adaptation strategies can be measured.

3.3.2 Domain shift with data augmentations

The introduction of image augmentationsh enable the model to learn more general and less domain dependent information, like object-specific colors or relative positions in the image. Three different data augmentation techniques have been explored: blurring, color and brightness changing, and noise addition to the image.
The application of Gaussian Blur appears to smoothen the features, which helps the model to better identify and delineate large and less intricate objects.
Color Jitter alters the color characteristics of the images, which can help the model generalize better by simulating different lighting conditions and learn less color-dependent features.
Noise instead can lead the model to be more robust to artifacts present in the image, since it learns not to depends on all inputs pixels.

3.3.3 Domain Adaptation via Fourier Domain adaptation

Looking at the images in the datasets, it's clearly possible to recognize whether they belong to Cityscapes or GTA5, and this is a reason for the poor results of the UDA problem. Therefore, our aim is to make the visual features of the source domain and of the target domain more similar to each other. In the context of Image-to-Image, the source domain is the set of images on which the training is performed while the target domain the one on which both the validation and testing are performed and it is usually unlabelled. In particular, GTA5 is considered a source domain and the Cityscapes a target domain of which the labels are not considered.

FDA approach is indeed based on the Fourier transform:

$$\mathcal{F}_{m,n}(x) = \sum_{h=1}^{H}\sum_{w=1}^{W} x(h,w)e^{-\imath 2\pi\left(\frac{h}{H}m+\frac{w}{W}n\right)} \qquad (1)$$

It applies the Fourier transform to the tensor representing the 3 channels of an image, converting them into their frequency domain representation. [8] shows that low-level frequencies' amplitude are dependent from the characteristics of the sensor, the illuminant, or other low-level sources of variability. Therefore, the solution considered is to substitute the lowest frequencies from a source sample with those from a target image.

Given the source sample $\{x_s, y_s\} \sim D^s$ and the target sample $\{x_t\} \sim D^t$, we define a mask function M_β that assigns a value of 1 to the center of the image, whose size depends on the parameter β and 0 elsewhere. The amplitude and phase components of the Fourier transform $\mathcal{F}$ are denoted as $\mathcal{F}^A$ and $\mathcal{F}^P$ respectively, and they are the input for the Fourier inverse $\mathcal{F}^{-1}$. Finally, everything is defined for the formalization of FDA:

$$x_{s\to t} = \mathcal{F}^{-1}([M_\beta \circ \mathcal{F}^A(x_t) + (1-M_\beta)\circ \mathcal{F}^A(x_s), \mathcal{F}^P(x_s)]) \qquad (2)$$

This transformation is applied on each of the three channels of an image, resulting in a new tensor whose size is the same as the one from the original tensor.

A traditional Cross-Entropy loss is not sufficient for the training since it would only see how the model performs on the source domain. Our purpose is indeed to taking account of target domain, even if we don't have the label of it, and this is perform adding an entropy minimization function weighted through a parameter λ defined as follows:

$$\begin{aligned}\mathcal{L}(f_\theta) =& \mathcal{L}_{ce}(f_\theta; x_{s\to t}, y_s) + \lambda \mathcal{L}_{ent}(f_\theta; y_t) \\ =& -\sum_i \langle y_i^s, \log(f_\theta(x_i^{s\to t})\rangle \\ &+ \lambda \sum_i \rho(-\langle f_\theta(x_i^t), \log(f_\theta(x_i^t))\rangle)\end{aligned} \qquad (3)$$

where $\rho(x) = (x^2 + 0.001^2)^\eta$ is the Charbonnier penalty, λ is a chosen empirically and f_θ is the segmentation model with parameters θ.

3.3.4 Domain Adaptation via Cross-domain Mixed Sampling

Unsupervised Domain Adaptation (UDA) deals with the problem where labelled data is available for a source domain, but only unlabelled data is available for the target domain. One of the proposed techniques is DACS [7] which utilize the notion of pseudo-labelling that are predictions of the target dataset used during training. Due to the tendency to conflate to the easier-to-predict classes [7], DACS compensate this problem by mixing the pseudo-labels with part of the semantic ground truth of the source images, ensuring that during the training, pixels coming from one domain will border the ones coming from the other domain.
In particular, given the source samples $\{x_s, y_s\} \sim D^s$, the target samples $\{x_t\} \sim D^t$, the pseudo-label generated by the current model f_θ as $y_t = f_\theta(x_t)$: the mixed images are created by selecting a portion of the classes present in y_s. A binary mask M for the selected classes is created, and finally the mixed label is $y_m = M \odot y_s + (1-M) \odot y_t$ with the same mask applied to the images as well to create x_m.
To perform back-propagation the following loss function is used:

$$\mathcal{L}(f_\theta) = \mathcal{L}_{ce}(f_\theta; x_s, y_s) + \lambda \mathcal{L}_{ce}(f_\theta; x_m, y_m) \qquad (4)$$

Where $\mathcal{L}_{ce}$ is the cross-entropy loss, and λ is a hyperparamenter that decides how much the unsupervised part affect the training. λ is chosen in a dynamic way and it represents the proportion of pixels where the prediction of the model on those images is above a threshold.

3.3.5 Rare Class Sampling

Rare Class Sampling (RCS) is a training strategy introduced in [4] to address the gap between the distribution of classes of the source domain and the target domain. This imbalance can lead to poor learning of rare classes which tends to reinforce predictions of the most common categories.

[4] notes that images having pixels belonging to rare classes usually don't appear at the beginning of the training procedure and therefore the model starts to learn them too late in order to get a satisfying evaluation. Hence, a solution would be to calculate prior to the model deployment the frequencies of the classes. We can set a temperature T, that permits to define how impactful are the frequencies of the rarest classes. When we want to generate a sample from a dataset, we condition this extraction by forcing the selected class to be present in the sample.
Defining f_c as the frequency in which the class c appear in the dataset D:

$$f_c = \frac{\sum_{i=1}^{|D|} \sum_{j=1}^{H} \sum_{k=1}^{W} \mathbb{1}(y_i[j,k] = c)}{|D| \cdot H \cdot W} \tag{5}$$

The sampling probability of class c is:

$$P(c) = \frac{e^{(1-f_c)/T}}{\sum_{c'=1}^{C} e^{(1-f_{c'})/T}} \tag{6}$$

Therefore rarer classes will have higher sampling probability. Lowering the temperature will increase the probability of extracting rarer classes, whilst increasing T will lead the distribution to be similar to a uniform distribution. It's easy to show that even if we extract multiple times rare classes while ignoring common classes such as *road*, a co-occurrence may happen. Meaning images that contain rare classes will contain common classes too, so the model can still learn the common classes even if not extracted by P.

3.3.6 Context-Aware High-Resolution Domain-Adaptive Semantic Segmentation

A common problem in UDA is the intensive use of GPU memory since to perform training, images from multiple domains or additional networks may be used. So the solution is to scale down the images. The problem is that low-resolution (LR) inputs fail to recognize smaller objects like traffic lights or poles, but high-resolution (HR) images overload the GPU. On the contrary HR inputs correctly classify small objects, but they are not able to identify long distance relations that LR inputs maintain (for example walls or sidewalks). So the objective of this technique is to utilize both of these inputs and make them compensate for each other. HRDA is then a multi-resolution framework for UDA in semantic segmentation [5].
Firstly, HRDA uses a LR crop, referred as the context crop, to learn large objects and long-distance relations, then from the context a HR crop, known as the detail crop, is extracted, which is used to classify small or fine objects without much need of the environment. Finally these two crops are merged with a self-attention mechanism that learns how much to trust LR or HR prediction in that region.
The paper of reference utilized the DAFormer architecture and DeepLabV2 as baselines, while we utilize BiSeNet to evaluate performance differences with other UDA techniques that have been explained.

Given the samples $\{x \in R^{H \times W \times 3}, y \in R^{H \times W \times 1}\} \sim D^s$, we define the bilinear downsample operator as $\zeta(x, 1/s) \in R^{\frac{H}{s} \times \frac{W}{s} \times C}, C = 1, 3$ for $s \geq 1$. We note that ζ is also an upsample operator when applied as $\zeta(x, s)$. From the image a context region crop $x_{c,HR}$ is extracted, it is then downsampled to obtain the context crop x_c. The detail crop x_d is a section extracted from the context region. Generating the right context and detail crop is open to discussions since different scales and crop proportions of the original image can be utilized:

$$x_{c,HR} = x[a_c : (a_c + h_c), b_c : (b_c + w_c)] \tag{7a}$$
$$x_c = \zeta(x, 1/s) \tag{7b}$$
$$x_d = x[a_d : (a_d + h_d), b_d : (b_d + w_d)] \tag{7c}$$

Where a_c, b_c, a_d, b_d are chosen from a uniform distribution calibrated to correctly generate crops of desired sizes.
Next we have to decompose the underlying model f_θ in a feature encoder f^E, a semantic decoder f^D, and an attention decoder f^A. The separation into three components is necessary since f^A is not computed on the final prediction of the model but on an internal representation with the goal of easily making the gradient flow on the internal layers. With the use of BiSeNet, we define the output of its *Feature Fusion Module* [9] as the output of the encoding head, and the rest of the model as the decoding head. The forward pass then becomes:

$$\hat{y}_c = f^D(f^E(x_c)) \in \mathbb{R}^{\frac{h_c}{s} \times \frac{w_c}{s} \times 1} \tag{8a}$$
$$\hat{y}_d = f^D(f^E(x_d)) \in \mathbb{R}^{h_d \times w_d \times 1} \tag{8b}$$

The detail prediction is aligned with the context region by padding it with zeros to match sizes and relative position of the detail in the context region itself, we call the obtain tensor $\hat{y}'_d$.
The heads f^E and f^D are so shared between the two inputs, not only it saves memory usage but it increases the robustness of the model against multi resolution inputs.
Then for the fusion of the predictions the f^A head learn to predict how much to trust LR and HR inputs via a self attention:

$$a_c = \sigma(f^A(f^E(x_c)) \in [0, 1]^{\frac{h_c}{s} \times \frac{w_c}{s} \times 1} \tag{9}$$

where $\sigma(\cdot)$ being the sigmoid function. a_c is then masked resulting in a'_c and upsampled: the initial resolution of the context region is restored and only the part aligned with x_d is maintained, outside the attention is set to 0.
Finally the fused prediction:

$$\hat{y}_f = \zeta((1 - a'_c) \odot \hat{y}_c), s) + \zeta(a'_c, s) \odot \hat{y}'_d \quad (10)$$

The main difference with [5] is that the attention we created is only on one channel and then broadcasted to be compatible with the predictions. We choose this since it's more explainable, and directly corresponds to a dynamic average of the detail and context predictions.

The model can be trained with $\hat{y}_f$ and $\hat{y}_d$, so the formalization of the loss is:

$$\mathcal{L}_{hrda}(f_\theta) = (1-\lambda_d)\cdot\mathcal{L}_{ce}(f_\theta; \hat{y}_f, y_{c,HR})+\lambda_d\cdot\mathcal{L}_{ce}(f_\theta; \hat{y}_d, y_d) \quad (11)$$

Where $y_{c,HR}$ is the label corresponding to $x_{c,HR}$ and y_d to x_d. A component supervising LR inputs can be added but it's already been covered by the fused prediction, and we decided not to use it.

This loss can be easily applied to the unsupervised part, but only $\mathcal{L}_{hrda}$ can be directly applied for the source images with prediction and ground truth and a component dependent on the unsupervised part may be introduced as in 3.3.3 or 3.3.4:

$$\mathcal{L}(f_\theta) = \mathcal{L}_{hrda}(f_\theta) + \lambda\mathcal{L}_u(f_\theta) \quad (12)$$

where $\mathcal{L}_u$ is a component evaluating the unsupervised part of training, and it's dependent on the the technique used (DACS or FDA).

3.4. Datasets

The results we presented are based on two popular datasets for semantic segmentation: GTA5 and Cityscapes. The source dataset GTA5 [6], contains 2500 synthetic labelled images and with 19 classes of size 1052×1914. The target dataset, Cityscapes, is composed of 1572 training images and 500 validation images taken from urban environments and labelled with 29 classes [2] with resolution of 1024×2048.
19 of the classes of the training dataset are compatible with the ones from the source dataset, all the other are considered as "void" and do not contribute in the evaluation. Results are reported in terms of Intersection over Union (IoU) per class and the mean across them (mIoU).

3.5. Network details

We employ the following method:

- DeepLab with a ResNet-101 backbone pre-trained on ImageNet, similarly as in [1]. To perform training we utilize stochastic gradient descent, initial learning rate of 1×10^{-3} with polynomial decay with power 0.9 at every epoch with the formula $initial\_lr \left(1 - \frac{epoch}{max\_epoch}\right)^{power}$. We use momentum of 0.9 and weight decay of 5×10^{-4}, batches are composed of 3 images due to resource limitations.

- BiSeNet with a ResNet-18 backbone pre-trained on ImageNet. As suggested in [9], the initial learning rate is 2.5×10^{-2} with polynomial decay with power 0.9 at every epoch with the same previous formula, momentum is 0.9 and weight decay is 1×10^{-4}. When it is not indicated otherwise, the number of elements per batch is 4 and the stochastic gradient descent is used.

4. Experiments and results

DeepLab The training process for the DeepLabV2 model was notably slow, achieving approximately 2 epochs per hour, which suggests potential limitations in available computational resources. The model demonstrated robust performance. The validation mIoU converged to approximately 61.95%, confirming its efficacy as a segmentation model, even with the observed training inefficiencies. The training mIoU converged around 80%, exhibiting a marginal yet consistent upward trend. This indicates that while further training might yield minor improvements, the model was approaching its performance ceiling within the given parameters.

As shown in Table 1 classes corresponding to *wall*, *fence*, *pole*, *light*, *rider*, *train* and *motocycle* proved most difficult for the model during validation, with mIoU values consistently below 50%. It is noteworthy that class 5 (*pole*) also showed a comparatively low mIoU of 45.34% even in the training phase, suggesting an inherent difficulty in its accurate segmentation. A positive observation was the presence of strong semantic relationships in common misclassifications. For instance, class 16 (*train*) was misclassified as class 15 (bus) in about 21% of cases. Such errors, where objects are confused with visually or functionally similar entities, are often preferable to entirely unrelated misclassifications, indicating a partial understanding of the object's context.

BiSeNet During the BiSeNet experiment the training loop proved markedly faster than with DeepLab V2: a full epoch required roughly 10–15 minutes. This lightweight dual-path design therefore offers a clear computational advantage whenever real-time inference or rapid prototyping is a priority.

The quantitative picture, however, reveals a speed–accuracy trade-off. On the training set the network converged swiftly to a mean IoU of 86 %, but the validation curve plateaued much earlier, peaking at 53.9 %. The widening gap between the two curves, together with a

Model	**mIoU**	Road	SW	Build	Wall	Fence	Pole	Light	Sign	Veg.	Terr.	Sky	Person	Rider	Car	Truck	Bus	Train	MC	Bike
DeepLabV2-R101	**61.95**	**96.51**	73.74	**87.28**	**43.73**	**40.10**	33.05	**39.06**	**53.92**	87.39	**52.76**	90.41	**64.69**	**43.12**	**89.58**	**64.00**	**63.67**	**48.12**	**44.97**	**61.04**
BiSeNet-ResNet18	53.92	96.39	**74.02**	85.94	32.82	32.52	**36.14**	31.69	51.10	**87.65**	52.66	**90.42**	63.71	30.42	86.96	24.22	35.30	24.46	28.28	59.73

Table 1. Validation mIoU (%) per class on Cityscapes after 50 epochs for the two baseline networks.

Model	mIoU	Flops $\times 10^9$	Mean Latency (ms)	Mean FPS	Num. Param $\times 10^6$
DeepLabV2	**61.95**	374.66	234.37	4.63	43.79
BiSeNet	53.92	**25.77**	**14.62**	**79.77**	**12.58**

Table 2. Model comparison in terms of deployment characteristics

validation loss that stabilised around 0.32, might indicates over-fitting: additional epochs keep improving memorisation of the source data without translating into better generalisation.

A closer look at the class-wise scores clarifies the performance ceiling. BiSeNet segments large, texture-rich regions extremely well, like *road* 96 %, *sky* 90 %, *vegetation* 88 %, *building* 86 %, confirming the effectiveness of its context path in capturing global cues, Table 1. Conversely, thin structures and low-frequency objects remain challenging: *wall* 33 %, *fence* 33 %, *pole* 36 %, *light* 32 %, along with the minor rider/vehicle classes (*rider*, *motorcycle*, *train*, *truck*) whose IoU hovers between 24 % and 35 %. The validation confusion matrix highlights semantically plausible mix-ups (e.g., *wall* frequently labelled as *building*, *truck* merged with *car*), suggesting that finer localisation rather than semantic comprehension is the limiting factor.

In summary, BiSeNet trades roughly eight percentage points of validation mIoU for an order-of-magnitude gain in throughput compared with DeepLabV2, in Table 2. If deployment speed outweighs the need for precise delineation of small or rare classes, this compromise may be acceptable; otherwise, further regularisation (early stopping, stronger data augmentation) or rare-class-aware training strategies will be necessary to narrow the accuracy gap without eroding its real-time edge.

Data Augmentation In Table 3 there are the outcomes from the different configurations possible with the chosen data augmentation.

Gaussian Blur reduces high-frequency noise and makes the model less sensitive to small imperfections or variations in image details, improving the mIoU of classes as *Fence, Pole, Person* and *Truck*.

Color Jitter, on the other hand, helps the model learn to segment correctly even under varying lighting conditions, not capturing small details from classes as *Fence* and *Pole* or specific classes as *Sidewalk, Terrain* and *Train*.

Gaussian Noise introduces random perturbation to the image, simulating typical noise conditions from images captured by cameras in less-than-ideal environments, where mIoU of only few classes increase.

Furthermore, the simultaneous use of these techniques provides a greater variety of data during training, virtually expanding the training dataset size and forcing the model to learn more general and less dataset-specific representations. In Table 3, the final data augmentation configuration ("All Aug."), produces the best results overall improving the model's robustness and enhancing its generalization on unseen data. The performance for dynamic and less structured classes, such as *Bicycle* and *Motorcycle*, benefits the most, suggesting that the added noise and color variation provide valuable learning opportunities for the model. This explains why we chose to test the combination of all three types of data augmentation, in order to improve the model's performance by leveraging the beneficial aspects of each of the previous-mentioned methods.

FDA As in the other experiments, we used a batch size of 4 for the source domain, and 2 for the target domain. The problem arises from the difference in dataset sizes: the source domain contains less than twice the number of images as the target domain. As a result, some target images must be reused for FDA, however this does not result in duplicate or redundant training data.

We chose the parameters $\eta = 1.5$ and $\lambda = 0.005$ as recommended in [8] and with this setting the results obtained are shown in Table 4. The first attempt was to apply data augmentation, specifically the best setting found during experimentation from Section 3.3.2, after FDA. FDA must indeed be performed on the original images because applying it to already augmented images could distort their frequency characteristics, potentially undermining the effectiveness of the adaptation. With parameter β set to 0.05, we made a comparison with a model trained on data applied to only FDA, but for the second model the are significantly better with a mIoU 11% higher. This is the reason why we gave up on adding the Data Augmentation after FDA.

Then other values tried for β are 0.01 and 0.09. When $\beta = 0.01$, the results are quite similar to the previous case, with a mIoU of 30.58% against 30.96% and the only class' mIoU significantly lower is for *Person* and *Rider*. Instead for the second choice of β, the performance achieved a mIoU of 27.98% which we consider worse with respect to

Method	mIoU	Road	SW	Build	Wall	Fence	Pole	Light	Sign	Veg.	Terr.	Sky	Person	Rider	Car	Truck	Bus	Train	MC	Bike
No Aug.	18.90	20.07	5.17	61.02	5.50	8.38	10.69	1.78	1.61	70.40	5.55	57.68	31.87	0.05	49.47	1.03	0.63	**9.23**	0.03	0.00
Blur	25.24	24.30	15.71	54.82	8.90	**16.95**	15.76	3.68	1.45	74.79	4.36	67.20	35.55	0.00	40.78	12.91	1.39	0.00	0.00	0.00
Color Jitter	20.53	24.77	3.20	65.40	7.55	5.95	6.43	**13.62**	**5.57**	70.56	2.88	72.30	**35.84**	**3.36**	49.11	2.26	0.12	0.00	0.69	0.01
Noise	21.15	18.67	1.55	64.05	1.10	9.49	14.30	10.95	5.49	60.55	1.96	66.25	27.29	1.36	43.07	9.97	0.00	0.00	2.26	0.00
All Aug.	**30.03**	**86.37**	**19.56**	**74.56**	**20.16**	5.81	**18.94**	11.80	4.83	**76.42**	**22.87**	**80.60**	33.88	1.42	**70.57**	**21.87**	**6.03**	3.72	**10.74**	**0.48**

Table 3. Validation mIoU per class with BiSeNet trained on augmented images of GTA5 and validated on CityScapes

Method	mIoU	Road	SW	Build	Wall	Fence	Pole	Light	Sign	Veg.	Terr.	Sky	Person	Rider	Car	Truck	Bus	Train	MC	Bike
All Aug., $\beta = 0.05$	19.73	56.66	29.17	62.19	5.37	1.18	13.97	2.82	1.10	46.82	9.40	66.82	15.01	0.91	25.50	5.85	1.41	**10.93**	0.04	0
No Aug., $\beta = 0.05$	**30.96**	**86.38**	32.15	**76.53**	17.57	**9.16**	20.69	10.39	**7.31**	**75.08**	14.25	75.67	**38.07**	**9.51**	66.49	9.74	3.50	0	**4.39**	**0.47**
No Aug., $\beta = 0.01$	30.58	85.82	27.97	76.32	**21.54**	7.13	**22.96**	**12.48**	5.46	74.82	**15.90**	77.40	34.54	1.35	**66.83**	10.37	5.29	0	4.16	0.13
No Aug., $\beta = 0.09$	27.98	85.55	**32.36**	73.95	14.16	8.15	19.78	8.03	6.41	72.02	10.88	**78.36**	30.30	1.04	64.08	**13.90**	**7.49**	1.11	3.54	0.43

Table 4. Result of applying FDA for GTA5 $\rightarrow$ Cityscapes

the configuration with $\beta = 0.05$. Again, the most underclassified classes are *Person* and *Rider*, but the model can detect better *Truck, Bus* and *Train*. A larger β indicates that the area of modified frequencies is larger as well, therefore the outcome presented shows that it is not necessary to have a large β, but only few of the low-frequencies are useful to alter the global style of the image achieving effective domain adaptation.

DACS Each batch is composed by 8 images, where half of it are samples form D^s and the other half are mixed images. Differently from the default settings, source images are resized to 720×1280 and target images to 512×1024, then crops of 512×1024 are extracted. Also we use the Stochastic Gradient Descent with Nesterov acceleration and initial learning rate of 2.5×10^{-4}. Deciding where to apply augmentations for model generalization is difficult and very laborious since we can: apply augmentation and then mixing or the reversed order, we can augment the target images, or augment the mixed images. With the thought of maintaining the pseudo-labels the more confident possible, we decided not to augment the target images. We first trained using all augmentations from the best configuration of Section 3.3.2 and applying them to the source images before mixing, reaching 26.65% mIoU, but with poor performance with respect to semantically similar classes (*Sidewalk* with *Road, Truck* with *Car*), and finer objects (*Fence, Light, Sign*), Table 5. So we tried the complete opposite, where no augmentation were performed at all, but had similar behaviour and 28.74% mIoU, with *Bus* and *Person* being recognized by the model, and with less performance on the other classes. Applying instead only blur and color jittering before mixing led to an improvement, 32.82% mIoU with an overall improvement in the already present classes except person not being recognized anymore. Applying these two augmentations after the mixing decreased the mIoU to 30.64%, with *Person* reappearing and *Fence* improving. Applying these augmentations only to the mixed images instead led to 33.86% mIoU, with *Bus* and *Train* recognized. Having said all this, we decided to augment both source images after mixing and mixed images due to how on one hand peoples were recognized and on the other, busses and trains, this did not work, reaching a peak of 32.48% mIoU.
An important observation, is that even at the latest epochs the model was still learning and improving, and motivated by the fact that we did not use DeepLabV2 as in [7], we modified the learning rate to the value of 2.5×10^{-2} indicated in [9], obtaining a huge improvement of 40.62% mIoU, the row "Other Lr." of Table 5. This suggest that the main difficulty of DACS is just slow convergence due to how much images are perturbed because of augmentations, and how the mask for the mixing is created, since it may happen that logical relations (car on the road) may not be preserved.

HRDA We used Stochastic Gradient Descent with Nesterov acceleration, source images are resized to 720×1280 and target images to 512×1024, then crops of 512×1024 are extracted. Batches are composed of 4 source images.
We applied this framework with different techniques, firstly we established a upper bound with a BiSeNet model both trained and validated on the target dataset to establish the power of HRDA. Then similar to our standard process, the lower bound with direct transfer from source to target dataset, and gradually introducing image-to-image adaptations, namely augmentations, FDA, and DACS.
Training on the target dataset led to an 55.91% mIoU, a difference of -6.04% and $+1.99\%$ from our results of DeepLab and BiSeNet respectively. After, we evaluated the domain shift, resulting in 23.62% mIoU, increasing the baseline imposed with the original model by 4.72 percentage points. With augmentations 32.47% mIoU has been reached, $+2.74\%$, Table 6. With the use of UDA techniques, the loss to be minimized changed to be similar to the ones specified before in Section 3.3.3 and 3.3.4:

$$\mathcal{L}_{hrda,fda} = \mathcal{L}_{hrda}(x_{s\rightarrow t}, y_s) + \lambda \mathcal{L}_{ent}(x_t) \quad (13a)$$

$$\mathcal{L}_{hrda,dacs} = \mathcal{L}_{hrda}(x_s, y_s) + \lambda \mathcal{L}_{hrda}(x_m, y_m) \quad (13b)$$

Method	mIoU	Road	SW	Build	Wall	Fence	Pole	Light	Sign	Veg.	Terr.	Sky	Person	Rider	Car	Truck	Bus	Train	MC	Bike
All Aug. Before M.	26.65	81.07	4.60	66.93	14.89	0.47	3.69	0.00	0.00	74.94	26.08	49.40	0.05	0.00	64.04	10.26	1.08	0.32	0.00	0.00
No Aug.	28.74	76.88	1.85	68.88	7.28	0.41	4.21	0.00	0.00	72.53	10.47	49.39	8.15	0.00	72.24	13.10	13.53	3.50	0.00	0.00
No Noise Before M.	32.83	81.85	2.53	74.30	15.29	4.25	5.91	0.00	0.00	74.70	18.13	**67.89**	0.03	0.00	74.85	20.77	9.37	9.75	0.00	0.00
No Noise After M.	30.64	81.19	2.91	73.13	12.57	12.14	2.21	0.00	0.01	70.119	15.99	60.02	6.29	0.00	73.49	18.08	0.00	0.62	0.00	0.00
Aug. on Mixed Img	33.86	81.94	2.37	74.72	15.98	9.80	4.12	0.00	0.00	76.99	25.85	67.08	0.01	0.00	73.56	15.83	11.21	14.74	0.00	0.00
Aug. after M. and on M.	32.48	81.62	4.92	63.58	13.47	15.56	8.39	0.00	0.00	75.32	25.20	67.58	0.23	0.13	75.85	17.54	12.58	**15.22**	0.00	0.00
Other Lr.	**40.62**	**88.82**	**29.56**	**78.67**	**21.45**	**22.66**	**30.41**	**11.92**	**26.96**	**83.92**	**32.02**	66.28	**48.33**	**7.88**	**80.81**	**25.69**	**27.06**	0.00	**8.15**	0.00

Table 5. Result of applying DACS for GTA5 → Cityscapes

Method	mIoU	Road	SW	Build	Wall	Fence	Pole	Light	Sign	Veg.	Terr.	Sky	Person	Rider	Car	Truck	Bus	Train	MC	Bike
Direct Transfer	23.62	59.97	6.23	62.50	9.80	5.88	10.07	2.22	2.19	66.81	13.83	72.98	16.19	0.00	32.00	11.47	1.39	4.45	0.00	0.00
Aug.	32.47	86.42	20.36	75.34	23.82	11.29	19.33	**6.52**	5.91	75.76	16.79	**78.86**	33.37	5.67	63.51	20.15	6.62	**26.75**	**8.00**	0.00
FDA	26.97	86.44	19.14	73.10	16.81	7.07	16.90	4.11	8.34	72.11	13.55	74.64	25.31	3.14	68.35	13.54	4.89	3.26	1.67	0.00
DACS + Aug.	32.14	79.38	21.82	67.16	20.28	14.79	**24.58**	4.99	7.47	**78.78**	16.95	61.63	**47.11**	**13.02**	75.32	18.32	22.35	0.76	3.90	0.00
DACS + Aug. + RCS	**34.65**	**87.86**	**24.84**	**75.38**	**24.18**	**14.48**	13.84	4.50	**18.17**	77.92	**35.15**	67.93	22.98	8.49	**78.98**	**28.64**	**28.59**	0.00	7.17	**4.54**
Upper Bound	55.91	96.26	71.61	85.36	44.14	35.43	29.03	26.62	46.12	86.23	49.31	89.47	59.40	30.36	86.54	42.32	46.92	46.94	34.96	55.32

Table 6. Result of HRDA on GTA5 → Cityscapes, small variability in result may happen due to HR crop generated at random

The performances of these configuration is reported in table 6. Applying FDA led to a general decrease in performance 26.97% compared to a 30.96%. For DACS alone, the model behave similar to when HRDA is not involved with major degradation in *light* and *sign*. Adding the support of RCS, did not have much effect on the aforementioned classes, but finally *Bike* class (the rarest) appeared.

5. Conclusions

Our analysis highlights the significant impact of model architectures, data augmentations, and training frameworks on domain shift. With respect of the model selection, we demonstrated the accuracy-speed trade-off of DeepLabV2 and BiSeNet models. We then showed that augmentations alone are not sufficient for accurate segmentation, and more advanced techniques are necessary. FDA is mathematically interesting and appears to have great potential for an effective performance. Although the performance outperforms the baseline, the results shows the absence of improvement with respect the performance achieved from the data augmentation setting. DACS achieved the best results overall, and incorporating RCS on top of it did not substantially degrade performances on common classes. So if the objective is better semantic separation rather than better refined classification, RCS is a viable addition. The HRDA framework, made the model generally more robust when trained on the target dataset, the addition of the high resolution input improved the segmentation of smaller objects, while the low resolution input still learned to classify the rest of the image. Notably, even if the resources at disposal were a fraction of SOTA approaches, the results obtained are still satisfactory. A notable limitation has been the lack of computational power and data available for the training.

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, May 2017. arXiv:1606.00915 [cs]. 2, 5

[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, Apr. 2016. arXiv:1604.01685 [cs]. 5

[3] Shijie Hao, Yuan Zhou, and Yanrong Guo. A Brief Survey on Semantic Segmentation with Deep Learning. *Neurocomputing*, 406:302–321, Sept. 2020. 1

[4] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. DAFormer: Improving Network Architectures and Training Strategies for Domain-Adaptive Semantic Segmentation, Mar. 2022. arXiv:2111.14887 [cs]. 1, 2, 4

[5] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. HRDA: Context-Aware High-Resolution Domain-Adaptive Semantic Segmentation, July 2022. arXiv:2204.13132 [cs]. 2, 4, 5

[6] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for Data: Ground Truth from Computer Games, Aug. 2016. arXiv:1608.02192 [cs]. 5

[7] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. DACS: Domain Adaptation via Cross-domain Mixed Sampling, Nov. 2020. arXiv:2007.08702 [cs]. 1, 2, 3, 7

[8] Yanchao Yang and Stefano Soatto. FDA: Fourier Domain Adaptation for Semantic Segmentation, Apr. 2020. arXiv:2004.05498 [cs]. 1, 2, 3, 6

[9] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation, Aug. 2018. arXiv:1808.00897 [cs]. 2, 4, 5, 7

[10] 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, Sept. 2020. arXiv:2009.00155 [cs]. 2