Real-time Domain Adaptation in Semantic Segmentation with Adversarial Learning

1st Zepeng Li
Politecnico di Torino
Turin, Italy
s327196@studenti.polito.it

2nd Felipe Miamoto
Politecnico di Torino
Turin, Italy
s321003@studenti.polito.it

3rd Alexandre Casarin
Politecnico di Torino
Turin, Italy
s306940@studenti.polito.it

***Abstract*—The present paper aims to train and evaluate a real-time deep neural network in order to provide semantic segmentation for street-view images. This task holds a big relevance in many fields, such as in autonomous driving, where the response time is extremely important. Collecting real-world labeled data, though, is an extensive work. To deal with it, a common solution is to use synthetic data to train the model, which naturally comes with a decrease in the performance. To partially avoid this effect, we employed some domain adaptation solutions to reduce the gap between the domains, such as data augmentation. Additionally, we utilized an adversarial learning approach to align the distributions of segmentation made on target and source images. The results show that both, the inclusion of augmented data and the use of adversarial learning, can lead to great improvements in the performance. Our findings highlight the efficacy of adding domain adaptation methods to deal with the domain shift problem, which is crucial for developing an effective and lightweight model for real-time semantic segmentation.**

***Index Terms*—semantic segmentation, domain adaptation, adversarial learning, real-time deep neural networks**

I. Problem overview

Semantic segmentation [6], a fundamental task in computer vision, involves the process of partitioning an image into semantically meaningful segments, each labeled with a specific class. Models like DeepLabV2 [4] have set high standards in terms of accuracy for this task. However, their complexity often impedes their deployment in real-time applications where the ability to quickly and accurately recognize various scenes, such as roads, cars, sidewalks and pedestrians, is crucial. To address these limitations, the BiSeNet (Bilateral Segmentation Network) [3] model has been developed. BiSeNet strikes a balance between speed and accuracy, achieving real-time performance while maintaining a level of accuracy comparable to more complex models like DeepLabV2.

Additionally, training a semantic segmentation model typically requires a large amount of labeled data, which is often time-consuming, labor-intensive, and costly to annotate. To mitigate this, synthetic data, such as the GTA5 [1] dataset, is used for training. However, this approach introduces domain shift, where significant differences between synthetic training data and real-world testing data distributions lead to poor generalization on real-world data. Even with annotated real-world data, domain shift persists due to variations in environmental conditions, such as weather changes, and differences in sensor data capture methods, further complicating the challenge of achieving robust performance across diverse real-world scenarios.

To mitigate this issue, researchers have introduced Unsupervised Domain Adaptation (UDA) techniques. UDA aims to bridge the gap between the source (synthetic) and target (real-world) domains without requiring labeled data from the target domain. In our assignment, we use GTA5 as the source dataset and Cityscapes [2] as the target dataset to simulate real-world conditions. Given the need for real-time performance, we selected BiSeNet as our semantic segmentation model.

To achieve domain alignment between the source and target datasets, we adopted an adversarial approach to UDA. This approach encourages the semantic segmentation model to learn feature representations that are invariant to the domain shift, thereby improving the model's generalization performance on the target data. Additionally, we conducted a series of baseline experiments (more details in methods part) to establish a reference for evaluating the effectiveness of our adversarial UDA approach. Our comprehensive experimental setup and results demonstrate the feasibility of using BiSeNet in conjunction with adversarial UDA to achieve both real-time performance and robust semantic segmentation across different domains.

II. Related works

A. Dataset

1) Cityscapes: This dataset contains high-quality pixel-level annotations for 5000 images from street scenes across 50 cities. It includes 2975 training images, 500 validation images, and 1525 test images, covering 30 classes with 19 main classes used for evaluation (e.g., cars, pedestrians, buildings). Cityscapes is crucial for semantic segmentation, instance segmentation, and object detection in urban environments, driving advancements in autonomous driving and urban scene understanding.

2) GTA5: Derived from the game Grand Theft Auto V, the GTA5 dataset provides approximately 24,966 high-resolution images depicting diverse urban environments with varied perspectives, weather conditions, and environmental changes. These images serve as synthetic data, mimicking real-world scenarios and effectively reducing the high costs associated with manual annotation. Annotations include detailed pixel-level labels for roads, vehicles, pedestrians, buildings, vegetation, and sky.

B. Models

1) DeepLabV2: DeepLabV2 is a classic deep learning model for semantic image segmentation. It employs atrous convolution (dilated convolution) to expand the network's receptive field without increasing parameters or computational costs, capturing multi-scale contextual information. The model features Atrous Spatial Pyramid Pooling (ASPP), using multiple atrous convolutions with varying dilation rates to effectively capture object details at different scales (More details are shown in Fig. 1). DeepLabV2 utilizes pre-trained networks like VGG-16 or ResNet-101 for robust feature extraction and processes input images at multiple scales for comprehensive object handling. It employs Fully Connected Conditional Random Fields (CRFs) in post-processing to refine segmentation by ensuring spatial label consistency, resulting in sharper boundaries and reduced noise. The architecture integrates these components to maintain high-resolution feature maps and enhance segmentation accuracy across various object sizes.

2) BiSeNet: BiSeNet is a real-time semantic segmentation model known for its efficiency and balanced performance. It adopts a dual-path architecture comprising a Spatial Path and a Context Path. The Spatial Path preserves high-resolution spatial details crucial for precise segmentation, while the Context Path employs feature down-sampling and dilated convolutions to capture global context and semantic information. BiSeNet integrates a Feature Fusion Module (FFM) to combine detailed spatial features from the Spatial Path with context-rich features from the Context Path, thereby enhancing segmentation accuracy. An attention mechanism further refines segmentation by focusing on critical regions and improving boundary precision. This design makes BiSeNet well-suited for applications such as autonomous driving, video surveillance, and augmented reality, where both speed and accuracy are essential. Similarly to DeepLab, BiSeNet can also be based in different models as a backbone, including ResNet18, ResNet101, Xception39 and VGG-16. The full architecture of BiSeNet is showen in Fig. 2.

C. Techniques used to perform domain adaptation

1) Data augmentation: Data augmentation plays a crucial role in alleviating domain shift by artificially expanding the diversity of the source domain data to better match the characteristics of the target domain. This approach involves applying transformations such as rotation, scaling, cropping, flipping, color jittering, or adding noise to the source domain data. By doing so, the augmented data not only increases the variability and richness of the training set but also introduces variations that are more representative of the target domain.

2) Adversarial learning approach: Adversarial domain adaptation [5] is a method in machine learning aimed at aligning feature distributions between different domains. It addresses the challenge of transferring knowledge from a source domain (with ample labeled data) to a target domain (with limited or no labeled data), while accounting for differences in their data distributions. This technique leverages adversarial training, where a feature extractor is trained alongside a domain discriminator. The feature extractor learns to generate domain-invariant features, beneficial for the task at hand across both domains, while the domain discriminator attempts to differentiate between features from the source and target domains. By minimizing differences in feature distributions, adversarial adaptation enhances model performance and robustness when applied to unseen data from the target domain, making it particularly useful in scenarios where labeled data in the target domain is scarce

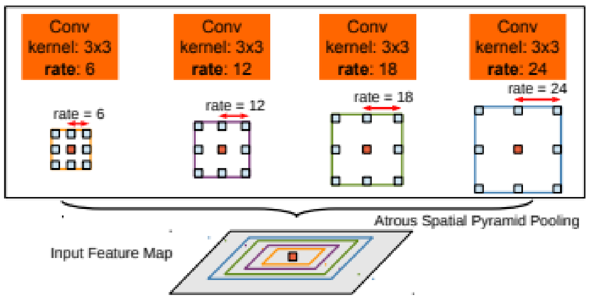
Conv kernel: 3x3 rate: 6
Conv kernel: 3x3 rate: 12
Conv kernel: 3x3 rate: 18
Conv kernel: 3x3 rate: 24
rate = 6
rate = 12
rate = 18
rate = 24
Atrous Spatial Pyramid Pooling
Input Feature Map

Fig. 1: Atrous Spatial Pyramid Pooling (ASPP)
Source: [4]

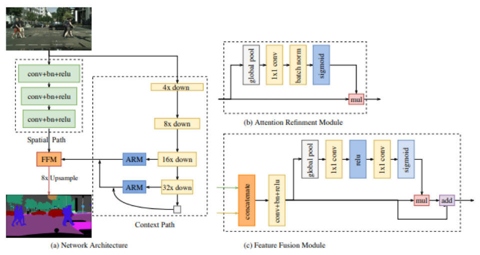

Fig. 2: BiSeNet architecture
Source: [3]

III. Methods

A. Classic semantic segmentation network

Classical semantic segmentation models and real-time models often exhibit differences in performance and inference speed. These differences significantly influence model selection, especially in fields such as autonomous driving, where real-time performance is crucial. To quantify these differences accurately, it is essential to establish a baseline experiment using a classical semantic segmentation model.

In this experiment, we adopt the DeepLabV2 framework with a ResNet-101 [7] model pre-trained on ImageNet as our segmentation baseline. However, it is noted that the current and subsequent experiments are conducted in the Google Colab environment, utilizing T4 GPU computational resources. Due to computational constraints and considerations for training efficiency, we did not implement the fully connected Conditional Random Field (CRF) proposed in the original DeepLabV2 paper [4], which is used to further smooth the noisy segmentation map.

We utilized the Cityscapes dataset to simulate real-world traffic data for training and validating our baseline network.

Considering computational resource limitations, we did not use the entire Cityscapes dataset. Instead, our training set comprised 1,572 samples from the original training dataset, and our validation set included 500 samples from the original validation dataset. To balance computational efficiency and fidelity, the resolution of the samples was downsampled from the original 2048x1024 to 1024x512 using nearest neighbor interpolation.

Additionally, due to constraints on computational power, we set the mini-batch size to 2. This is significantly different from the batch size of 20 mentioned in the original DeepLabV2 paper [4]. To mitigate the impact of this change, we adjusted the initial learning rate to 1e-4 (1e-3 for the last classifier layer), which is one-tenth of the rate suggested in the original DeepLabV2 paper [4]. Our optimization method is stochastic gradient descent (SGD) with a momentum of 0.9 and a weight decay of 5e-4. Regarding loss function, we chose Cross-entropy loss. According to the original DeepLabV2 paper [4], employing a polynomial learning rate decay policy can yield better optimization results compared to a step learning rate policy. Thus, in this experiment, we decided to use the polynomial learning rate decay policy with max-iter of 50 (same as the number of epochs) and with a power of 0.9.

Finally, we trained the baseline network on the Cityscapes training dataset for 50 epochs and evaluated its performance on the Cityscapes validation dataset using several key metrics: Mean Intersection over Union (mIoU), latency, FLOPs, FPS, and the number of parameters. These results serve as reference benchmarks for the performance and inference speed of classical semantic segmentation models.

B. Real-time semantic segmentation network

In this experiment, our focus shifted towards the evaluation of real-time semantic segmentation models. We quantitatively compared their performance and processing speed against traditional semantic segmentation models, contrasting these results with those obtained in our previous experiment. In related work, we highlighted BiSeNet as an efficient and lightweight semantic segmentation model. BiSeNet's architecture, which integrates a Spatial Path and a Context Path, allows for a large receptive field while retaining essential spatial information, rendering it highly suitable for real-time applications. For our experiment, we adopted the ResNet-18 [7] model pretrained on ImageNet as the baseline network within the BiSeNet framework. Similarly, owing to insufficient computing resources, we did not utilize the auxiliary loss function suggested in the original BiSeNet paper [3] for model optimization.

Maintaining consistency with our previous experiment, we used the downsampled Cityscapes dataset for both training and validation of our baseline model. The model was trained for 50 epochs, and its performance was evaluated on the Cityscapes validation dataset using metrics such as Mean Intersection over Union (mIoU), latency, FLOPs, FPS and the number of parameters. The mIoU metric is considered the upper bound for performance in subsequent experiments.

In terms of hyperparameter selection, we chose a batch size of 8, which is half of the recommended size in the original BiSeNet paper [3]. Although this reduction could potentially increase the variance in gradient calculations, it did not significantly impact the final optimization results in this context. Moreover, maintaining the initial learning rate at 2.5e-2, as suggested by BiSeNet, facilitated faster convergence to a certain extent. Cross-entropy loss was employed alongside the SGD optimizer, with momentum set to 0.9 and weight decay to 1e-4. For learning rate scheduling, we used a polynomial decay policy with max-iter of 50 (equivalent to the number of epochs) and a power of 0.9.

C. Evaluating the domain shift problem in Semantic Segmentation

After comparing real-time to classic segmentation networks, it is important to evaluate the impact of the domain shift when training a model.

Domain shift refers to the discrepancy between the data from the source domain, where a model is trained, and the one coming from the target domain, where the model is deployed. In the context of semantic segmentation, this problem is particularly pronounced when training on synthetic datasets (such as GTA5) and deploying on real-world datasets (like Cityscapes). This shift can significantly impact the performance of a model due to differences in visual characteristics like lighting, textures, colors and object appearances between synthetic and real-world images.

One example, noticeable in the top two parts of Fig. 8, is that synthetic datasets often have exaggerated colors and lighting conditions that do not match real-world scenarios. This may lead to a model that learns features that are not applicable to real-world images, thus causing the model performance to drop when evaluated on the target domain.

As mentioned in the previous section, BiSeNet is a lighter model compared to DeepLabv2. Therefore, we continued with this architecture to perform the next experiments, considering also our computational resources limitations.

First, the objective was to determine a lower bound mIoU. By training on GTA5 and evaluating on Cityscapes (validation split) without using any domain adaptation technique, it was possible to measure the impact of the domain shift on the performance of the model. This was important to establish a baseline, serving as a reference point to quantify the effectiveness of applying the techniques that will be discussed in the following sections.

In this step, we used a batch-size of 8, with cross-entropy loss and SGD optimizer. The parameters considered for the optimizer were: initial learning-rate = 2.5e-2, momentum = 0.9, weight-decay = 1e-4. Furthermore, it was included a polynomial learning-rate decay, decaying every epoch with max-iter = number of epochs (50) and power = 0.9.

It is also important to mention that we are using 2500 images from GTA5 dataset, which represents only a subset of the original one (24966 images) and the images were resized to 1280x720 (from 1914x1052) to optimize operations computationally. The validation was performed using Cityscapes dataset, specifically on the validation split. This same configuration was used to perform the next experiments.

D. Data augmentations to reduce the domain shift

It is not a secret that domain shift affects negatively the segmentation network's performance. Then, to enhance the generalization capability of the model trained on a synthetic domain, one of the possible naïve approaches is to use data augmentations during training. This technique artificially modifies the visual appearance of source images to make them more similar to target ones.

Data augmentation serves as a strategy to increase the variability of the training dataset through various transformations, which is crucial in addressing domain shift. Therefore, the method applied was to test the usage of some augmentation techniques in order to make synthetic data more representative of real-world scenarios.

To evaluate the effectiveness of data augmentations, we trained the BiSeNet with a variety of augmented GTA5 data and tested it on the Cityscapes dataset. We applied each augmentation technique separately (with a probability of 50% of been applied in each image) and then tested the combination of the 2 best augmentations. The technical specifications of the training process, such as batch-size, loss and optimizer were the same used in the previous experiment III-C

The specific augmentations tested were:

- Random Invert: This technique inverts colors by assigning the complementary RGB values to reach 1 in each channel (once pixels were scaled to [0,1] interval by toTensor() transformation in pytorch)
- Color Jitter: This involves changing brightness, contrast, saturation, and hue. Brightness and contrast adjustments can make light areas lighter and dark areas darker, while saturation adjustments control the vividness of colors. These three adjustments are applied with a parameter of 0.5, meaning a uniform random change in the interval [-50%, +50%], and a hue shift of 0.1 radians (a small shift of colors on the color wheel).
- Gaussian Blur: This reduces image noise and details, adding a blurring effect by averaging pixel values with their neighbors, weighted by their distance from the center of a sliding kernel.
- Adjust Sharpness: This method changes the sharpness of images. Once we were already considering Gaussian Blur, it would make more sense to test an increase in sharpness by using a factor of 2, making edges more distinct and enhancing high-frequency components.
- Add Gaussian Noise: This adds random noise to the pixels, slightly altering their colors. It also adds variability and can act as regularization, preventing overfitting and encouraging robustness to small changes.

By exposing the model to a broader range of variations during training, data augmentations can improve the ability of the model to generalize to unseen real-world data. Because of this exposure, the model can learn robust features that are less sensitive to domain-specific characteristics, which can create a smoother transition between synthetic and real-world domains, facilitating domain alignment.

E. Domain adaption based on adversarial approaches

Although data augmentation can alleviate the impact of domain shift to some extent, this approach is rather naïve. To more effectively address the domain shift problem, more sophisticated methods are required. It should be kept in mind that our ultimate goal is to enable our segmentation model to achieve acceptable generalization performance in the target domain. In semantic segmentation tasks, the output spaces of different domains share a significant amount of similarities, such as spatial layout and local context (as illustrated in Fig. 8). Therefore, if we can make the target domain predictions closer to those in the source domain, we can mitigate the effects of domain shift to a considerable extent.

In this experiment, we employ adversarial training, inspired by Generative Adversarial Networks (GANs) [8], to achieve domain adaptation. This approach comprises two main components: 1) a segmentation network G that predicts output results, and 2) a discriminator D that distinguishes whether the input is from the source or target segmentation output. Upon completion of the training process, we aim for both the segmentation model and the discriminator to achieve robust performance. This alignment in the output space between the two domains facilitates effective domain adaptation.

Fig. 7 [5] illustrates in detail how adversarial training is utilized for domain adaptation. Here, H represents the height of the input image, W represents the width, and C denotes the number of different categories.

Firstly, the source image I_s (with annotations) and the target image I_t (without annotations) are forwarded through the segmentation network to obtain their respective feature maps. These feature maps are then upsampled to their original sizes. Finally, after passing through a softmax layer, we obtain the source output P_s and the target output P_t, respectively. Each $P^{(h,w)}$ is a vector of length C, where each value in the vector represents the probability of the corresponding category being predicted.

Since the source images have ground truth labels Y_s (where each $Y^{(h,w)}$ is a one-hot vector of length C indicating the true category label of the corresponding pixel), we can optimize G using the cross-entropy loss L_{seg}. The segmentation loss L_{seg} can be defined as:

$$\mathcal{L}_{seg}(I_s) = -\sum_{h,w}\sum_{c\epsilon C} Y_s^{(h,w,c)} \log(P_s^{(h,w,c)}) \tag{1}$$

On the other hand, for the training of the discriminator, we forward the segmentation softmax output P into a fully-convolutional discriminator D. This allows us to obtain the probability of each pixel belonging to either the source domain or the target domain. Similarly, we can optimize D using the cross-entropy loss L_d. The loss function can be written as:

$$\mathcal{L}_d(P) = -\sum_{h,w}(1-z)\log(\mathbf{D}(P)^{(h,w,0)}) + z\log(\mathbf{D}(P)^{(h,w,1)}) \tag{2}$$

where $z = 0$ if the sample is drawn from the target domain, and $z = 1$ for the sample from the source domain.

To facilitate adversarial training, we include an adversarial loss L_{adv} alongside the segmentation loss to jointly optimize G. The adversarial loss can be formulated as:

$$\mathcal{L}_{adv}(I_t) = -\sum_{h,w}\log(\mathbf{D}(P_t)^{(h,w,1)}) \tag{3}$$

This loss function incentivizes the discriminator to classify the target output P_t as originating from the source domain, thereby adjusting the weights of the segmentation model G. This alignment aims to make the distribution of the target output P_t similar to that of the source output P_s. Therefore, the final optimization objective for the segmentation model can be defined as:

$$\mathcal{L}(I_s, I_t) = \mathcal{L}_{seg}(I_s) + \lambda_{adv}\mathcal{L}_{adv}(I_t) \tag{4}$$

where λ_{adv} is a weighting factor for the adversarial term L_{adv}.

For this experiment, the configuration of the models is as follows:

The segmentation model remains BiseNet, consistent with the previous experiment. The discriminator architecture mirrors that used in [5]. The output predictions of the segmentation model pass through 4 convolutional layers, each followed by a leaky ReLU activation with a fixed negative slope of 0.2. A classifier layer at the end produces classification outputs, distinguishing whether inputs belong to the source or target domain.

For the segmentation model, we maintain the same parameter settings as in the previous experiment. For the discriminator, we employ the Adam optimizer with a learning rate of 1e-4, applying the same polynomial decay as used in the segmentation network. The momentum values are set to 0.9 and 0.99. For adversarial training, in order to balance the training of the Segmentation model and the discriminator, we set the parameter λ_{adv} to 0.001.

In terms of the dataset, this experiment differs from the previous ones. For training samples, we not only utilize the Gta5 dataset to train the segmentation model but also employ the Cityscapes training dataset for adversarial training of the segmentation model. Both the Gta5 and Cityscapes datasets are used jointly to train the discriminator. The Cityscapes validation dataset is subsequently used to evaluate the model's final performance. Similar to the previous experiments, the Gta5 dataset is downsampled to 1280×720 using nearest-neighbor interpolation, while the Cityscapes training and validation datasets are downsampled to 1024×512.

Additionally, prior to training the model, we will employ the best-performing data augmentation strategy from the previous experiment. This approach will allow us to investigate the extent to which adversarial training can enhance performance.

IV. Results

A. Classic v.s. Real-time semantic segmentation network

As illustrated in Table I, the results from the first two experiments demonstrate that training DeepLabV2 on the Cityscapes training set yields a mean Intersection over Union (mIoU) of 55.3% on the Cityscapes validation set. Similarly, BiseNet achieves a comparable mIoU of 52.2%, establishing an upper bound for performance in subsequent experiments. The observation that DeepLabV2 achieves a higher mIoU on the validation set compared to BiSeNet aligns with our expectations; however, the margin of superiority is minimal. One possible explanation for this small difference is the constraint imposed by computational limitations, which necessitated setting the batch size to 2 during our experiments with DeepLabV2. In contrast, the authors of the original DeepLabV2 paper [4] used a batch size of 20, indicating a substantial discrepancy. A smaller batch size can lead to higher variance in gradient estimates, potentially affecting the final optimization of parameters. For BiSeNet, although we used a batch size of 8 compared to the batch size of 16 used in the original BiSeNet paper [3], this difference is relatively minor and results in an acceptable impact on the variance of gradient estimates.

Additionally, a comparative analysis of both models in terms of latency and frames per second (FPS) reveals a significant advantage of BiseNet over DeepLabV2 in real-time performance. Specifically, BiseNet exhibits an approximately 18-fold increase in speed. The evaluation outcomes of Experiments 1 and 2 substantiate that BiseNet, as a lightweight model, is highly suitable for deployment in real-time applications.

B. Evaluating the domain shift problem in Semantic Segmentation

The chart 4 shows the performance of the model trained on GTA5 and directly evaluated on Cityscapes. By comparing this result to the one obtained on IV-A (Chart 3), it is clear the impact of the domain shift. Here, it is important to highlight the huge difference between validation mIoU on both, which quantifies the observable difference between the datasets.

Meanwhile the previous section provides an upper bound to the final performance of our model (Fig. 3), this experiment defines a lower bound for it (Fig. 4).

In the table IIa, it is possible to see the overall performance of the model as well as the per class mIoU. It is evident that the capability of segmenting certain classes such as road, building, vegetation and sky are much higher than other such as motorcycle or bicycle. This may be attributed to the fact that the former group of classes are more prevalent, covering larger areas of the images (class imbalance) and also have more consistent shapes and appearances.

In general, results do not present a good mIoU. Specifically, we can see that the mIoU for the car class is less than 40%, which would represent a clear risk for cases in which real time segmentation is useful, such as for self-driving cars. Given that, it is important to further analyze the results with the

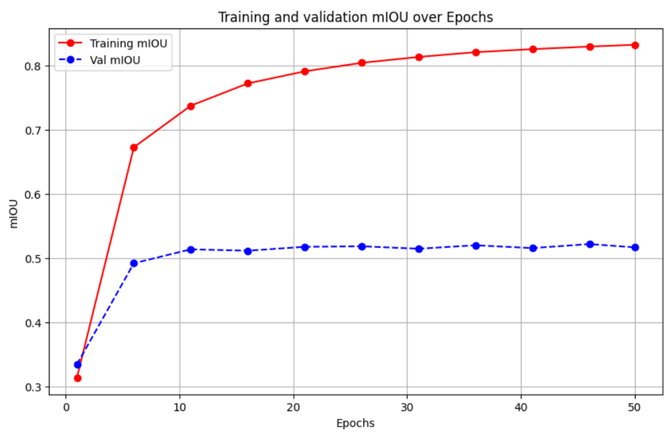

Fig. 3: mIoU - BiSeNet

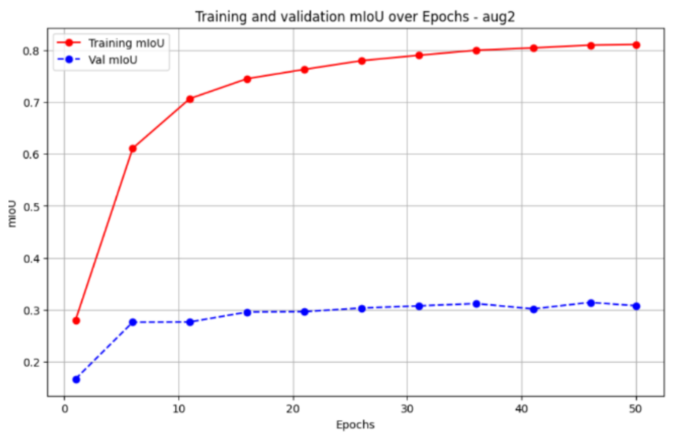

Fig. 5: mIoU - augmentation 2

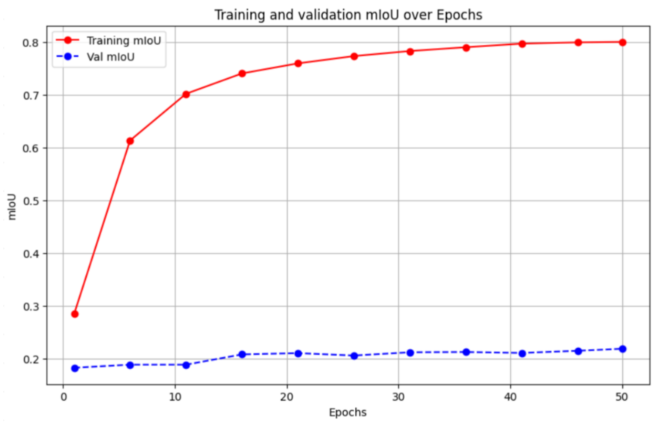

Fig. 4: mIoU - baseline

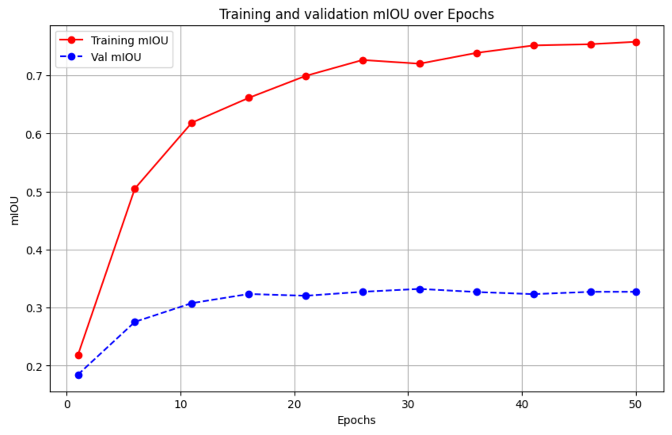

Fig. 6: mIoU - adversarial approach

inclusion of augmentations and how closer they can help to reach a desired result.

C. Data augmentations to reduce the domain shift

Noticeably, as seen in table IIa, augmentations techniques really have the capability of improving results. In particular, augmentation 2 (color jitter) improved mIoU the most (with the learning curve shown in Fig. 5). Furthermore, augmentation 5 (random noise) also increased substantially upon the baseline. Probably, both of them improve due to the difference observed in colors and lighting conditions between synthetic and real data, as discussed in III-C.

Therefore, the combination of augmentations tested was Aug. 2 + Aug. 5. Once both really improved upon the baseline, we would expect a combination of them to improve further more. However, as seen in Table IIb it did not. The mIoU obtained by it was actually worse than Aug. 2 alone.

Here, we consolidate some hypothesis for it:

- Redundancy: color jitter already introduces variability in color and texture, which might be sufficient for improving model performance. Adding random noise (which slightly changes colors) on top of this might not provide additional benefits and could instead introduce unnecessary complexity or confusion
- Interaction: when combined, the noise might disrupt the beneficial effects of color jitter by introducing unnecessary variability that interferes with the segmentation task

Overall, applying color jittering into the images improves a lot, increasing the mIoU by more than 40% relative to the baseline. This demonstrates the effectiveness of adding augmentations to synthetic images in order to improve the model performance on real ones.

To visualize this improvement, we can take a look at figures 9, 10, 11. It is evident that the segmentation quality improved a lot, specially regarding cars, which can be justified by the relevant increase in the car class' mIoU.

D. Domain adaption based on adversarial approaches

In the previous experiment, we identified the optimal data augmentation strategy (color jitter) for the GTA5 dataset, achieving a mean Intersection over Union (mIoU) of 31.4% on the Cityscapes validation set. As shown in Table IIb and Fig. 6, further applying adversarial training on this foundation results in an additional improvement of approximately 5.7% in mIoU on the Cityscapes validation set. This outcome suggests

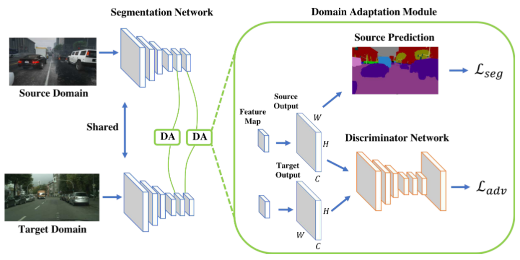

Fig. 7: Algorithmic overview
Source: [5]

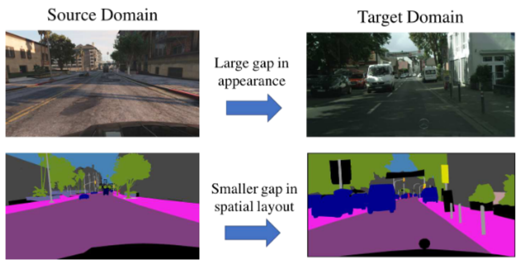

Fig. 8: Motivation of learning adaptation in the output space
Source: [5]

that combining adversarial adaptation with data augmentation effectively further mitigate domain shift issues encountered in real-time semantic segmentation tasks. Fig. 12 provides a visual representation of this enhancement.

We can observe that the application of adversarial training indeed generates smoother and more accurate segmentation maps. For instance, by comparing the bottom portions of Fig. 11 and Fig. 12, we notice that adversarial training significantly improves the recognition of roads, reducing the noise seen in Fig. 11, where large portions of the road are misclassified as terrain. This observation aligns with the results shown in Table II, where the model utilizing adversarial training achieves an mIoU of 89% for road segmentation, outperforming other experiments.

Furthermore, focusing on the right side of the images, we find that adversarial training considerably enhances the accuracy of sidewalk recognition. This is corroborated by Table II, where the model with adversarial training attains an mIoU of 36% for sidewalks, an 11% improvement over the model that only employs color jitter augmentation.

Finally, based on both figures and Table II, we can also observe improvements in car prediction accuracy with adversarial training. For example, in the central part of the images, our model does not misclassify roads as cars, and at the bottom of the images, it accurately delineates the edges of cars to a greater extent.

V. Conclusion

The results show that real-time semantic segmentation models can lead to strong improvement in speed when trained on the Cityscapes dataset, with BiSeNet increasing FPS from 3.70

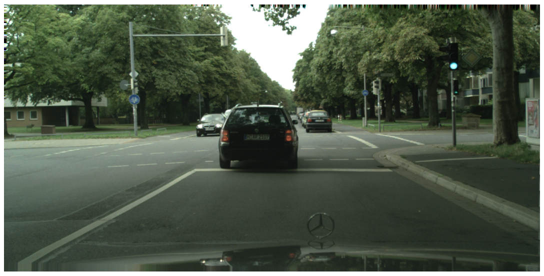

Fig. 9: real image

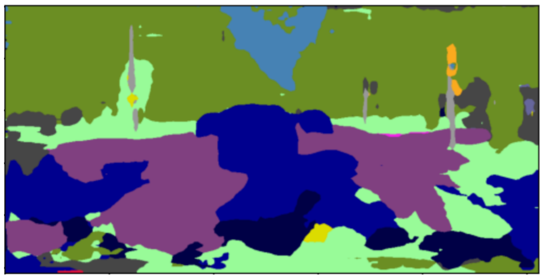

Fig. 10: segmentation - without augmentation

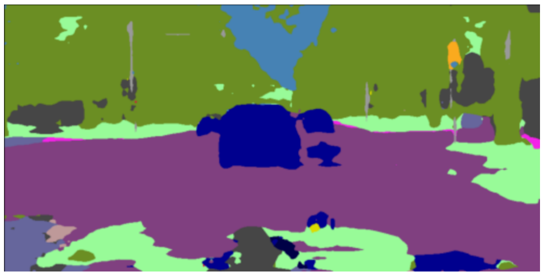

Fig. 11: segmentation - with augmentation

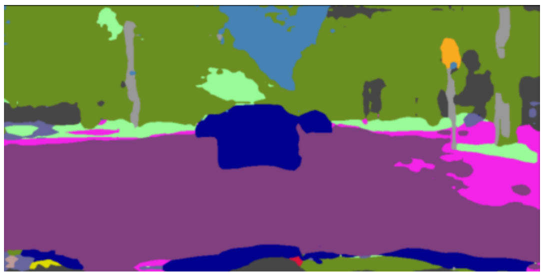

Fig. 12: segmentation - adversarial with augmentation

TABLE I: Performance Comparison of DeepLabV2 and BiSeNet Models on the Cityscapes Dataset

	mIoU	*FPS*	*Latency*	*FLOPs*	*Params*
DeepLabV2	55.3%	3.697	271.90ms	2.911G	43.901M
BiSeNet	52.2%	65.677	15.47ms	0.302G	12.582M

TABLE II: Domain Shift GTA5 → Cityscapes

(a) Intersection over unit (IoU) performances (%) for BiSeNet - simple augmentations

Classes	***Base***	***Aug1***	***Aug2***	***Aug3***	***Aug4***	***Aug5***
Road	0.59	0.65	0.84	0.44	0.50	0.70
Sidewalk	0.09	0.16	0.25	0.12	0.13	0.05
Building	0.63	0.66	0.74	0.70	0.66	0.70
Wall	0.09	0.12	0.22	0.11	0.09	0.09
Fence	0.03	0.09	0.11	0.14	0.10	0.08
Pole	0.14	0.17	0.22	0.21	0.16	0.18
Light	0.13	0.09	0.16	0.16	0.11	0.18
Sign	0.09	0.06	0.07	0.07	0.05	0.08
Vegetation	0.76	0.78	0.78	0.77	0.76	0.74
Terrain	0.11	0.11	0.16	0.06	0.05	0.06
Sky	0.56	0.57	0.75	0.70	0.68	0.71
Person	0.39	0.37	0.41	0.38	0.34	0.35
Rider	0.01	0.00	0.08	0.07	0.01	0.03
Car	0.36	0.52	0.73	0.35	0.47	0.68
Truck	0.07	0.06	0.21	0.03	0.11	0.09
Bus	0.05	0.03	0.13	0.02	0.03	0.02
Train	0.04	0.02	0.00	0.00	0.00	0.00
Motorcycle	0.02	0.01	0.07	0.07	0.05	0.03
Bicycle	0.00	0.00	0.04	0.01	0.01	0.00
mIoU	**0.219**	**0.236**	**0.314**	**0.232**	**0.227**	**0.250**

(b) Intersection over unit (IoU) performances (%) for BiSeNet - combined augmentations and adversarial approach

Classes	***Aug2+Aug5***	***UDA on Aug2***
Road	0.84	0.89
Sidewalk	0.21	0.36
Building	0.74	0.77
Wall	0.21	0.21
Fence	0.09	0.09
Pole	0.19	0.22
Light	0.14	0.14
Sign	0.06	0.07
Vegetation	0.78	0.77
Terrain	0.22	0.22
Sky	0.84	0.78
Person	0.42	0.37
Rider	0.07	0.09
Car	0.67	0.75
Truck	0.12	0.22
Bus	0.11	0.20
Train	0.00	0.05
Motorcycle	0.07	0.04
Bicycle	0.05	0.02
mIoU	**0.308**	**0.332**

to 65.68 and having a much lower latency when compared to DeepLabV2.

On the other hand, it is important to highlight that, by choosing BiSeNet instead of DeepLabV2, we are trading performance for speed. Yet, once we were mainly interested in real-time segmentation, the fact that BiSeNet is a much lighter and faster model makes it much more suitable for autonomous driving applications, in which the time response must be almost instantaneously.

Results also make clear the drop in performance when domain shift is applied between training and validation, a problem effectively addressed by the introduction of data augmentation. This technique presented a good performance on reducing the gap between the domains, which highly increased the mIoU performance (specially for some important classes, such as car and road).

On the last step, we concluded that, by approximating the distributions of segmentation maps on source and target domains, adversarial learning improved even more the model performance when combined with data augmentation techniques. The learning curve of the adversarial training part (Fig. 6) clears the improvement upon the baseline (Fig. 4), getting closer to the upperbound (Fig. 3).

Even with it, there is still room for improvement. Here, we point a few topics that could be addressed:

- performance differences among classes
- computational limitations
- single-layer adversarial approach

Regarding the first point, it is visible that the performance on some classes were better than on others. This effect can be explained by the class imbalance naturally present on city images. There are more pixels classified for these prevailing classes, which makes it easier for the model to correctly learn to segment them. For that reason, dealing with this issue could lead to even better results.

About the computational limitations, we had to make some adaptations to guarantee the possibility of successfully completing the training process. As mentioned, this included training BiSeNet with a batch size of 8 instead of 16 and using only a subset of GTA5 and Cityscapes, rather than the whole dataset, to train and validate the model. We believe that with more computational resources a better performance could be achieved, once it would enable the usage of the whole dataset and a larger batch size.

Lastly, it is important to point that we used a single-layer adversarial approach, which means that there is still margin to improvement if a multi-layer adversarial learning is applied.

References

[1] Stephan R. Richter, Vibhav Vineet, Stefan Roth, Vladlen Koltun, "Playing for Data: Ground Truth from Computer Games", ECCV, 2016

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, Bernt Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding", CVPR, 2016

[3] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, Nong Sang, "BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation", ECCV 2018

[4] Liang-Chieh Chen, George Papandreou, Senior Member, IEEE, Iasonas Kokkinos, Member, IEEE, Kevin Murphy, and Alan L. Yuille, Fellow, IEEE, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs", IEEE transactions on pattern analysis and machine intelligence, 2017

[5] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, Manmohan Chandraker, "Learning to Adapt Structured Output Space for Semantic Segmentation", CVPR, 2018

[6] S Hao, Y Zhou, Y Guo, "A brief survey on semantic segmentation with deep learning", Neurocomputing, 2020

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", CVPR, 2016

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative adversarial networks", Communications of the ACM, Volume 63, Issue 11