Project 4: Real-time Domain Adaptation in Semantic Segmentation

Avantaggiato Vincenzo s323112
Cazzola Michele s323270
De Luca Marco s322912

Abstract

Segmentation models are often computationally complex, leading to high latency, which makes them unsuitable for real-time applications. On the other hand, real-time models often sacrifice performance. This project addresses the problem of real-time semantic segmentation by comparing the high-latency model DeepLabv2 with the real-time model PIDNet-S and evaluating techniques to improve domain adaptation on the latter (dataset augmentations, Adversarial Domain Adaptation, Image-to-Image Domain Adaptation). LoveDA, a dataset designed for semantic segmentation in geospatial domains, contains images from urban and rural scenes with different class distributions. Therefore, it is very good for evaluating the domain shift. Additionally, other segmentation models, BiSeNet and STDC, are assessed. Various segmentation loss functions were also explored, including cross-entropy loss (CE), weighted cross-entropy loss (WCE), Online Hard Example Mining Cross Entropy (OHEM CE) loss, and focal loss. Code can be found at https://github.com/MicheleCazzola/aml-2025-project4.

1. Introduction

Semantic segmentation is a computer vision task whose goal is to categorize each pixel in an image into a class or object. The goal is to produce a dense pixel-wise segmentation map of an image, where each pixel is assigned to a specific class or object. Many segmentation models are too complex, having as consequence a high latency that makes them unusable in real-time application like autonomous driving. Real-time models often come with the trade off of worse performance.

Gathering labeled data for semantic segmentation is a long and costly task, so it is a common habitude to use datasets off-the-shelf and then adapting them to the needed domain.

In this work we will address both problems, comparing DeepLab (a high-latency model) and PIDNet (a real time model) and evaluating tehcniques to improve domain shift on the latter. As an extension, other segmentation models (BiSeNet and STDCNet) are evaluated.

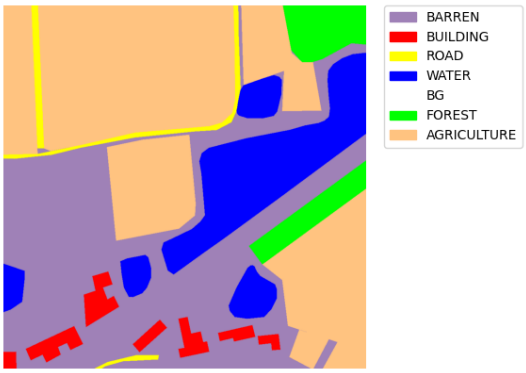

Figure 1. Example of a segmentation mask in LoveDA

2. Related work

2.1. Architectures

DeepLab [4] is a Deep Convolutional Neural Network architecture for semantic segmentation, proposed by [4] in 2017. DCNNs were originally designed for image classification [17, 23, 24] , with a repeated combination of max-pooling and downsampling layers resulting in a significant reduction in the spatial resolution. To overcome this problem, the last few downsampling layers are substituted by upsampling ones. This new convolution is called *atrous convolution* and is performed by using a "filter with holes" that takes inspiration from the *algorithme à trous* [15], already used in the context of DCNNs by [10, 19, 22].

DeepLab deals with the presence of objects at different scales by adopting a *atrous spatial pyramid pooling* (ASPP), which uses multiple parallel atrous convolutional layers with different sampling rates in order to recreate the successful pyramid spatial pooling of [11]. Finally, invariance to spacial transformations is achieved using a fully connected Conditional Random Field (CRF) by [18].

BiSeNet [29] introduced a new segmentation network composed of two branches: the *Spatial Path* (SP) and the *Context Path* (CP). SP (three layers, each composed of a convolution with stride = 2, batch normalization and ReLU)

extracts the output feature maps that is 1/8 of the original image, encoding rich spatial information due to the large spatial size of feature map. CP is designed to provide sufficient receptive field, using a lightweight model (like Xception [5]) and global average pooling.

In 2021, [7] improved the BiSeNet by removing structure redundancy and overcoming its two-stream design, creating the Short-Term Dense Concatenate network (**STDC** network). The context path is realized in an analog way, using 5 downsampling stages based on convolutions to perform feature extraction; as in BiSeNet, the features from the last two stages are refined using Attention Refine Modules. Moreover, they are fused with the features extracted from the backbone and those coming from the Detail Guidance, used as decoder part. Detail Guidance is performed by inserting the Detail Loss in the third stage of the backbone, to generate more detailed feature maps. The Detail ground-truth are binary feature maps generated from the segmentation ground-truth, encoding precise corner and boundary information.

PIDNet [28] takes inspiration from the PID controller in order to create a three-branch network architecture opposed to the Two-Branch Networks [7, 16, 20, 21, 29] that inherently suffer from overshoot issues (they are considered as PI controllers), attaching to them an auxiliary derivative branch (ADB). The three branches have complementary responsibilities: the proportional (**P**) parses and preserves detailed information in high-resolution feature maps, the integral (**I**) aggregates context information both locally and globally to parse long-range dependencies, the derivative (**D**) extracts high-frequency features to predict boundary regions.

A Pixel-attention-guided fusion module (Pag) is used by the P branch to selectively learn the useful semantic features from I branch without being overwhelmed. PIDNet-M and PIDNet-S use a parallelized version of DAPPM [16], called PAPPM, to perform pyramid pooling, while PIDNet-L uses the original DAPPM. Boundary-attention-guided fusion module (Bag) is employed to guide the fusion of detailed (P) and context (I) representations, given the boundary features extracted by ADB.

2.2. Domain Adaptation

Adversarial Domain Adaptation. Domain Adaptation methods address the domain-shift problem between the source and the target domains by aligning their feature distribution. [8, 9] introduced DomainAdversarial Neural Network (DANN) to transfer the feature distribution. PixelDA [2] method addresses domain adaptation for image classification by transferring the source images to target domain, thereby obtaining a simulated training set for target images. Domain adaptation for pixel-level prediction tasks was explored by [14] (using a fully-convolutional approach) and by CyCADA [13] (using a CycleGAN [30]).

In 2018, [26] introduced a new algorithm for adversarial domain adaptation, starting from pixel-level predictions. It consists of two different networks $\mathbf{G}$ (segmentation network) and $\mathbf{D}$ (discriminator), which are trained using a labeled source dataset and an unlabeled target dataset. During training, $\mathbf{G}$ is optimized using source images and the segmentation softmax output predicted on the target image. Predictions on source and target image are then given as input to $\mathbf{D}$. The adversarial approach consists in training $\mathbf{G}$ and fooling $\mathbf{D}$ by propagating gradients from $\mathbf{D}$ to $\mathbf{G}$ and thus maximizing the probability of the target prediction being considered as the source prediction.

The segmentation network is DeepLab-v2 [4] with ResNet-101 [12] pre-trained on ImageNet [6]. The discriminator consists of 5 convolutional layers with kernel 4×4, stride of 2 and channel number of $\{64, 128, 256, 512, 1\}$, respectively, and interleaved by leaky ReLU parameterized by 0.2. At the end, an up-sampling layer is added.

The objective function is the weighted sum of the segmentation loss (computed on $\mathbf{G}$) and the adversarial loss (computed on $\mathbf{D}$).

Image-to-Image Domain Adaptation. Domain Adaptation via Cross-domain Mixed Sampling (**DACS**) [25] performs domain adaptation using augmented samples by mixing pairs of images from different domains. In particular, the ground-truth semantic map are used to extract a few classes from a source domain image and paste them onto an unlabeled target domain image.

To construct pseudo-labels for the new image, source domain labels are mixed with pseudo-labels created from the target domain image. Mixing across domains this way leads to parts of the pseudo-labels being replaced by parts of the ground-truth semantic maps, ensuring that over the course of training all classes will border pixels from the other domain.

2.3. Segmentation losses

In the following, some relevant segmentation losses are analyzed, highlighting their differences.

Cross Entropy loss. [1] It is a pixel-level loss and approaches 0 as the predicted probability for the target class approaches 1. It is based on the concept of cross-entropy, so it measures the difference between the probability distribution of predicted labels and the one of target class. It is defined as follows:

$$L_{CE}(y, t) = -\sum_{n=1}^{N} \log(t_n \cdot y_n) \tag{1}$$

where N is the number of pixels, t_n, y_n are respectively target class and predicted label for pixel n. Cross Entropy Loss comes also in a weighted variant, to take into account the

more and less represented classes in an unbalanced dataset:

$$L_{WCE}(y,t,w) = -\sum_{n=1}^{N} t_n \cdot w \log(t_n \cdot y_n) \qquad (2)$$

where w are the weights for all classes.

Online Hard Example Mining Cross Entropy (OHEM CE) loss. It is a Cross-Entropy Loss which considers only the hardest samples in each batch of the dataset, as done by [28]. After computing the pixel-wise model scores, it retains only those with confidence under a given threshold t. Moreover, it imposes the constraint to consider at least a given number of pixels m. Both the threshold and the minimum number of pixels are hyperparameters to choose with cross-validation.

Focal loss. [1] It is a modified version of cross-entropy loss that assigns weights differently to hard samples (samples with a high probability of being misclassified) and easy samples (with a high probability of being correctly classified). That is what is called prioritized sampling. It is defined as follows:

$$L_{\text{focal}}(y,t,\gamma) = -\sum_{n=1}^{N} (1 - t_n \cdot y_n)^{\gamma} \log(t_n \cdot y_n) \qquad (3)$$

where γ is a non-negative tunable hyperparameter. If γ is 0 for all samples, it is a plain cross-entropy loss.

3. Methodologies

This section summarizes the steps we followed during the project, the choices we made and the reason behind them.

3.1. Dataset analysis

LoveDA [27]: Land-cOVEr Domain Adaptive Semantic Segmentation is a dataset for a semantic segmentation task, released in 2021 by the Wuhan University, China. It is used in the geospatial domain. The dataset consists of 5987 images with 20658 labeled objects belonging to 7 different classes including background, road, building, and other: forest, water, agriculture, and barren. Images in the LoveDA dataset have pixel-level semantic segmentation annotations. There are 1796 (30% of the total) unlabeled images (i.e. without annotations). There are 3 splits in the dataset: train (2522 images), test (1796 images), and val (1669 images).

Each split is divided in two domains: urban and rural, making it a perfect dataset for our experiments on domain adaptation.

The major characteristics of this dataset are summarized as follows:

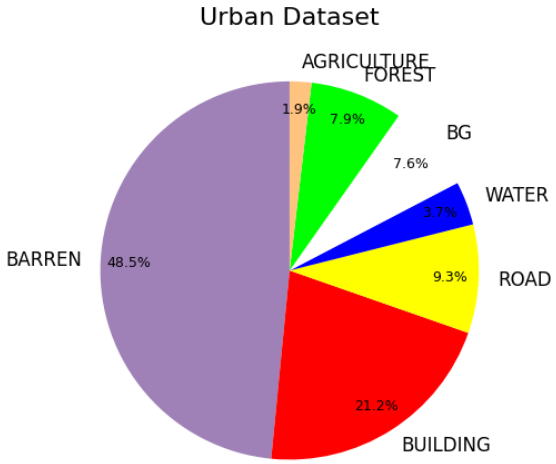

Figure 2. Relative distribution of classes in the urban training dataset of LoveDA

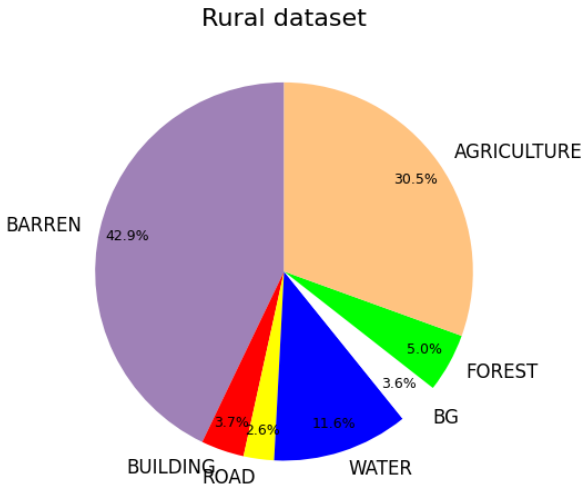

Figure 3. Relative distribution of classes in the rural validation dataset of LoveDA

1. Multi-scale objects. The HSR images were collected from 18 complex urban and rural scenes, covering three different cities in China. The objects in the same category are in completely different geographical landscapes in the different scenes, which increases the scale variation.

2. Complex background samples. The remote sensing semantic segmentation task is always faced with the complex background samples.

3. Inconsistent class distributions. The urban and rural scenes have different class distributions. The urban

scenes with high population densities contain lots of artificial objects such as buildings and roads. In contrast, the rural scenes include more natural elements, such as forest and water. More details on the distribution of classes in the two domains can be seen in Figures 2 and 3.

3.2. Segmentation and domain adaptation

In this project, three main tasks were performed:

1. Training and evaluating on the same domain (Urban).
2. Testing the domain-shift capabilities of models, by training on Urban and evaluating on Rural domain.
3. Applying domain adaptation techniques to reduce domain shift.

After a first comparison between *DeepLabv2* and *PIDNet-S* both on tasks 1 and 2, we then focused on *PIDNet-S*, choosing the best augmentations and completing task 3.

3.3. Proposed extensions

As an extension to the previous work, we:

- first tried different techniques in order to improve the performance of *PIDNet-S*. Specifically, we weighted the CrossEntropyLoss using the class distribution in order to pay more attention to underrepresented classes and we also tried using other loss functions (OHEM Loss, Focal Loss).
- then, we decided evaluate on tasks 1, 2 other architectures that were not previously tested on this dataset (*BiSeNet*, *STDC*).

3.4. The process

Our process was structured into a series of stages. We began by understanding the data representation used in LoveDA and defining the structure for the data loader. Additionally, we established a common framework for representing model performance and plotting results. To ensure consistency in model execution, we defined standardized patterns, including a common evaluation function, enforcing reproducibility, and saving checkpoints at every epoch.

Once these preliminary steps were completed, we implemented DeepLabv2 based on the work of [4]. We then made key preprocessing decisions, such as selecting an appropriate normalization scheme, resizing images to accommodate computational resource limitations, and tuning hyperparameters. Afterward, we trained and validated the model on the urban split.

Next, we implemented PIDNet-S following [28]. As with DeepLabv2, we performed parameter tuning and trained and validated the model on the urban split. We then conducted a comparison between DeepLabv2 and PIDNet-S to assess their relative performance. Furthermore, we analyzed the domain shift effect by training PIDNet-S on the urban split and validating it on the rural split without modifying any parameters. We also tried the same thing on DeepLab to see the difference in the impact.

To enhance model robustness, we selected a set of augmentations based on meaningful properties inherent to the dataset. We then evaluated the model using the best-performing augmentations. Additionally, we implemented an unsupervised domain adaptation approach based on [26], leveraging PIDNet-S and the selected augmentations. We also implemented an image-to-image translation method following [25], again using PIDNet-S and the chosen augmentations.

Finally, we analyzed the performance of these domain adaptation approaches in comparison to the standard model. We then explored broader considerations and potential extensions of our analysis, including alternative models and objective functions, and evaluated the impact of these proposed extensions on overall performance.

4. Experiments

In this section we will present the experiments conducted, each evaluated using mIoU (mean Intersection over Union) as the only metric. To keep a fair comparison, all experiments are run with a batch size of 6 and downsampling the input images and masks to $(512, 512)$ for 20 epochs. More detailed results can be found on the GitHub repository.

4.1. Model comparison

DeepLabv2. As a first experiment we evaluated the performance of DeepLabv2 in semantic segmentation on a single domain (urban). We use the cross-entropy loss as criterion for model predictions. Training is performed using Adam optimizer (with a learning rate of 0.001 and a weight decay of 0.001) and a step scheduler (with a step size of 10 and $\gamma = 0.1$). Best result of 36.69% is obtained at epoch 15. Due to its high latency, training and evaluating this model was non-trivial, forcing us to use checkpoints to stop and resume training.

PIDNet-S. PIDNet-S model is trained on the same task using a SGD optimizer (with a learning rate of 0.001, momentum of 0.9 and a weight decay of 0.01, which is quite high but proved to perform better) and a step scheduler (with a step size of 10 and $\gamma = 0.1$). While DeepLabv2 computes just one loss function, PIDNet-S must account for its multi-branch structure. Therefore, here are all the employed losses:

- semantic loss involving both P and I branches consists

Model	Latency	FLOPs	Params
DeepLabv2	484.40 ms	0.729T	43.016M
PIDNet-S	22.82 ms	25.331G	7.718M
BiSeNet	119.21 ms	0.178T	50.404M
STDC	28.97 ms	35.396T	12.044M

Table 1. Model comparison on Latency, FLOPs and parameters. More details on Figure 5.

of a cross-entropy between the weighted sum of the outputs of the two branches and the ground truth masks (this is called $loss_s$)

- semantic loss involving branch I is again a cross-entropy between the branch output and boundary pseudo-labels computed starting from outputs of branch D (this is $loss_{sb}$)

- boundary loss involving branch D is instead a weighted binary cross-entropy between the branch output and the ground truth boundaries (this is $loss_b$)

Best result of 33.15% is reached at epoch 6.

Due to its size (see Table 1), DeepLabv2 allows reaching a higher mIoU, but comes with the burden of high latency, making it useless for Real-Time Applications. The 3.5% reduction of mIoU in PIDNet-S is an acceptable trade-off for a network that is 20x faster in inference.

4.2. Domain shift and mitigations

Domain shift. After evaluating the two different models on the urban split, we validated PIDNet-S on the rural split. The result, as expected, was worse: the best mIoU measured on PIDNet-S for the rural split is 23.32% at epoch 14, with the same parameters. We tried the same thing on DeepLabv2, and the best mIoU was 21.05% at epoch 19.

Augmentations. Considering the characteristics of the dataset (satellite images with a lot of small objects), we tried to figure out possible augmentations in order to improve PIDNet-S performance. We considered an online augmentation approach, i.e. random dataset images are transformed at load time. We first considered single augmentations and then some combinations (randomly apply one augmentation or the other one, mix both augmentations). An overview of all experimented augmentations can be found in Figure 4. The name and the implementation of each augmentation are defined by the Python library Albumentations [3].

The best performing augmentations (in terms of mIoU) are:

- Random Crop (28.16% at epoch 19)

- Random sized Crop (24.97% at epoch 18)

- Horizontal Flip (23.82% at epoch 10)

Random Crop and Random sized Crop select one portion of the image, respectively given a fixed crop size (512) and a size interval (64 to 512). They guide the model in focusing on specific parts of the image, and thus improve segmentation for objects which are small in the original images. Horizontal Flip flips the image horizontally allowing the model to generalize better over objects with different structures, which can or can not contain some kinds of symmetry.

Since combinations of different augmentations were not good enough, we chose to only apply Random Crop in the following steps because it significantly improves the model performance. Augmentation has been applied only on source dataset.

4.3. Domain adaptation

Adversarial approach. We then considered an adversarial domain adaptation technique to understand if there is an improvement in the model generalization ability across different domains. Following [26], we used PIDNet as segmentation network and a fully connected discriminator. The loss of the segmentation network is the combination of the three losses computed by PIDNet, whereas the loss of the discriminator is a binary cross entropy.

Since these are two distinct networks, their training process is defined differently. We used SGD as optimizer for the segmentation network and Adam for the domain discriminator. For the segmentation network, the used parameters are the same as in the previous analyses (learning rate of 0.001, momentum of 0.9, weight decay of 0.01), whereas for the discriminator, the learning rate is 10^{-5} no weight decay is used. The step scheduler is common to both networks, with step size = 10 and gamma = 0.1. Again, the values are justified by performance improvements.

This configuration provided a mIoU of 25.50% at epoch 12. As we can see, this result, obtained combining adversarial approach and the best augmentation, is worse than using just augmentations. Moreover, we noticed that the loss of the discriminator is basically constant over the various epochs. This could be due to the characteristics of features provided to the discriminator, thus leading to the conclusion that adversarial approach is not an optimal solution to the domain shift issue for our dataset.

Image to image. As an alternative approach, we implemented an image to image translation pattern following [25]. Like in the previous case, training is performed using images from both labeled source domain and unlabeled target domain, and the same parameters for optimizer and scheduler are used.

However, while the adversarial approach uses two separate networks which are totally different, the original implementation of image to image approach uses the segmentation model and an auxiliary model (EMA model). The

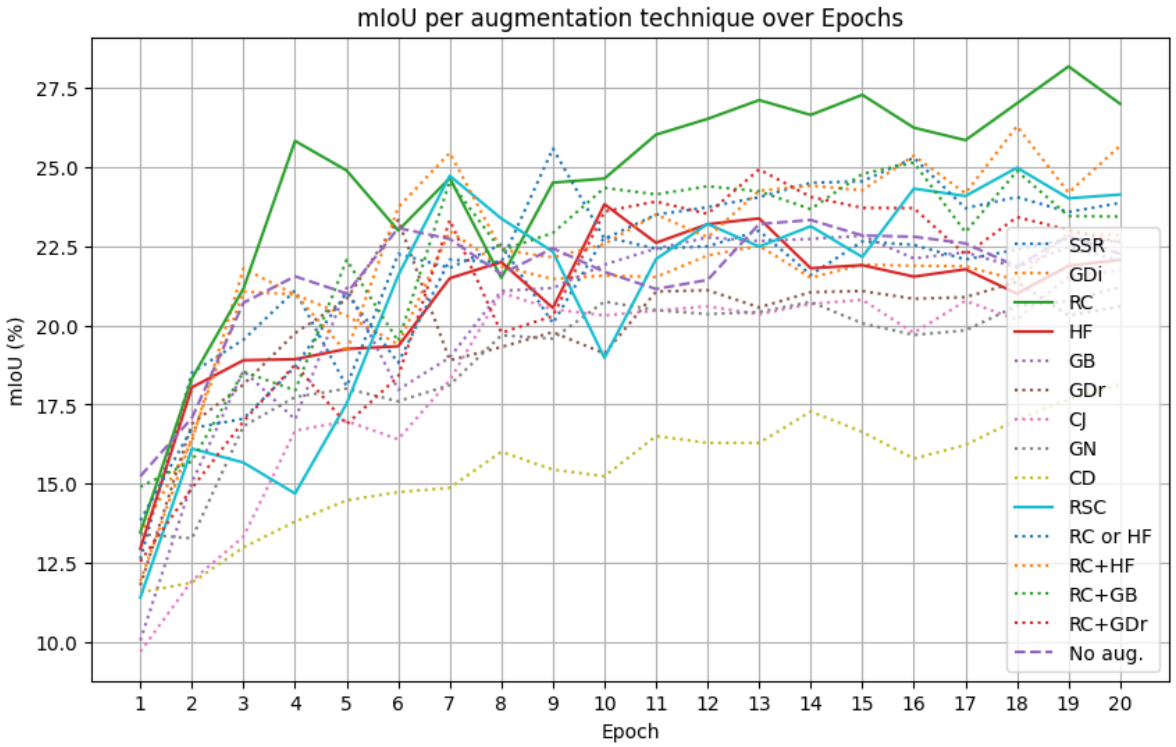

Figure 4. mIoU of different augmentations over 20 epochs. (SSR = Shift Scale Rotate, GDi = Grid Distorsion, RC = Random Crop, HF = Horizontal Flip, GB = Gaussian Blur, GDr = Grid Dropout, CJ = Color Jitter, GN = Gaussian Noise, CD = Channel Dropout, RSC = Random Scaled Crop)

latter is actually a copy of the former, but its gradients are not back-propagated and instead they are manually updated at the end of each epoch.

The EMA model is used to compute pseudo-labels that are going to be mixed with labels belonging to randomly selected classes from source domain. Instead, the main model computes predictions on source images, thus optimizing the "labeled loss", but it also predicts over a mix between source and target images, optimizing the "unlabeled loss".

However, the best results are obtained without using the EMA model (mIoU 25.36% at epoch 18), but predicting the pseudo-labels with the same PIDNet model we are training. Still, these results are not as good as the previously obtained one, meaning that the DACS approach is not suitable for LoveDA + PIDNet combination.

We can impute this degradation to the nature of LoveDA dataset that, containing satellite images, is not suitable for class mixing as some strange combinations (like buildings on water) can occur. Moreover, in this dataset few classes cover most of the image. Therefore, if selected, they cause the whole source to be transferred to the target, thus making mixing useless.

4.4. Extensions

4.4.1 Improving PIDNet-S

Class weights. Considering the unbalanced nature of classes in LoveDA, we thought to introduce a penalty in the loss to account for this disparity. We first calculated the classes distribution on the urban split, because the rural split must be considered as unlabeled data and therefore excluded from estimations like this, otherwise the model would result biased. Then, using the following formulation:

$$w_i = \frac{1}{p_i^\alpha} \tag{4}$$

$$w_i' = \frac{w_i}{\frac{1}{n}\sum_{j=1}^{n} w_j} \tag{5}$$

$$p_i = \frac{c_i}{\sum_{j=1}^{n} c_j} \tag{6}$$

where c_i is the count of pixels belonging to class i, n is the number of classes, $\alpha = 0.5$ is a hyperparameter. w_i represents the weight for class i and w_i' is its normalization.

Weights are then provided as input to the weighted cross-entropy. This approach slightly degraded results in Urban to

Experiment	Barren	Building	Road	Water	Background	Forest	Agriculture	Total
Simple domain shift	49.34	26.59	20.89	26.63	7.17	4.28	24.92	23.32
Random Crop	49.19	34.64	24.68	28.30	6.86	6.82	38.39	28.16
Random Sized Crop	42.80	15.62	23.83	37.50	4.58	8.88	34.82	24.97
Horizontal Flip	49.02	32.80	16.90	20.85	3.61	3.18	31.84	23.82
Random Crop + Adversarial	49.46	28.29	24.64	25.11	9.02	5.61	33.19	25.50
Random Crop + DACS	41.67	29.77	25.10	24.10	3.89	14.38	36.55	26.17
Weighted CE	48.33	29.59	18.19	25.46	14.61	6.63	24.94	25.45
Weighted CE + Random Crop	47.91	29.16	26.70	27.89	8.56	9.66	38.83	27.82

Table 2. mIoUs (%) per class in domain shift for different PIDNet-S experiments

Urban approach but lead to a 2.3% improvement in domain shift.

OHEM Cross Entropy. This experiment was carried out using the same parameters as before together with the best augmentation, but using OHEM Cross Entropy instead of standard CE. Chosen OHEM parameters ($threshold = 0.9$ and $keep = 150000$) did not lead to any improvement, reaching 19.47% in domain shift and 27.75% on urban domain.

Focal Loss. Same procedure was repeated with focal loss (with $\gamma = 2$) but still no improvement was found. Best results are 28.50% on urban domain and 21.20% in domain shift.

4.4.2 Testing new architectures

As a final step for this project, we thought of two alternative networks to highlight their difference in terms of performance to PIDNet-S, considering the complexity of LoveDA. The models were analyzed in their inference capabilities on the same domain and in case of domain shift, without using any augmentation.

BiSeNet. We tried BiSeNet using as context path ResNet101. Training and validating on urban split results in a mIoU of 32.85% at epoch 13. This was reached using a learning rate of 0.001, step size 10, gamma 0.1, weight decay 0.0001. This model shows some big peaks in validation loss, corresponding to drops in mIoU, but more or less the trend is acceptable.

Training on urban split and validating on rural split results, of course, in worse performance, reaching a mIoU of 21.45%, with a trend for the validation loss which is not much different from model inference on the same domain.

STDC. We wanted also to try STDC, using as backbone STDCNet813 and just focusing on the version without detailed guidance. Performing both training and validation on urban split results in a mIoU of 33.09% at epoch 8. This was obtained using the same parameters as for BiSeNet, except for a learning rate of 0.0001. The trend presented by the model is unstable in terms of mIoU, which has many ups and downs. However, it still reaches performance comparable to PIDNet.

Training and validating on different splits results in a trend which is more stable and reaches 25.67%, so it performs better than BiSeNet.

Figure 5 shows a comparison between all the models we tested. When it comes to urban to urban training/evaluation, PIDNet-S and STDC perform similar in terms of mIoU, whereas in urban vs rural training/evaluation STDC overcomes PIDNet-S. However, latency in STDC is higher, so PIDNet-S is the best one in terms of Latency/mIoU trade-off.

5. Conclusions

Our experiments confirm that PIDNet is a valid architecture, that can lead to really good results if correctly tuned expecially considering that we used the small version of the model. Further research can be conducted in the field of domain adaptation, since the explored approaches do not improve the final results. We took this work as an occasion to improve our deep learning related skills, understand the under-the-hood implementation of model architectures and how to engineer models to perform different tasks.

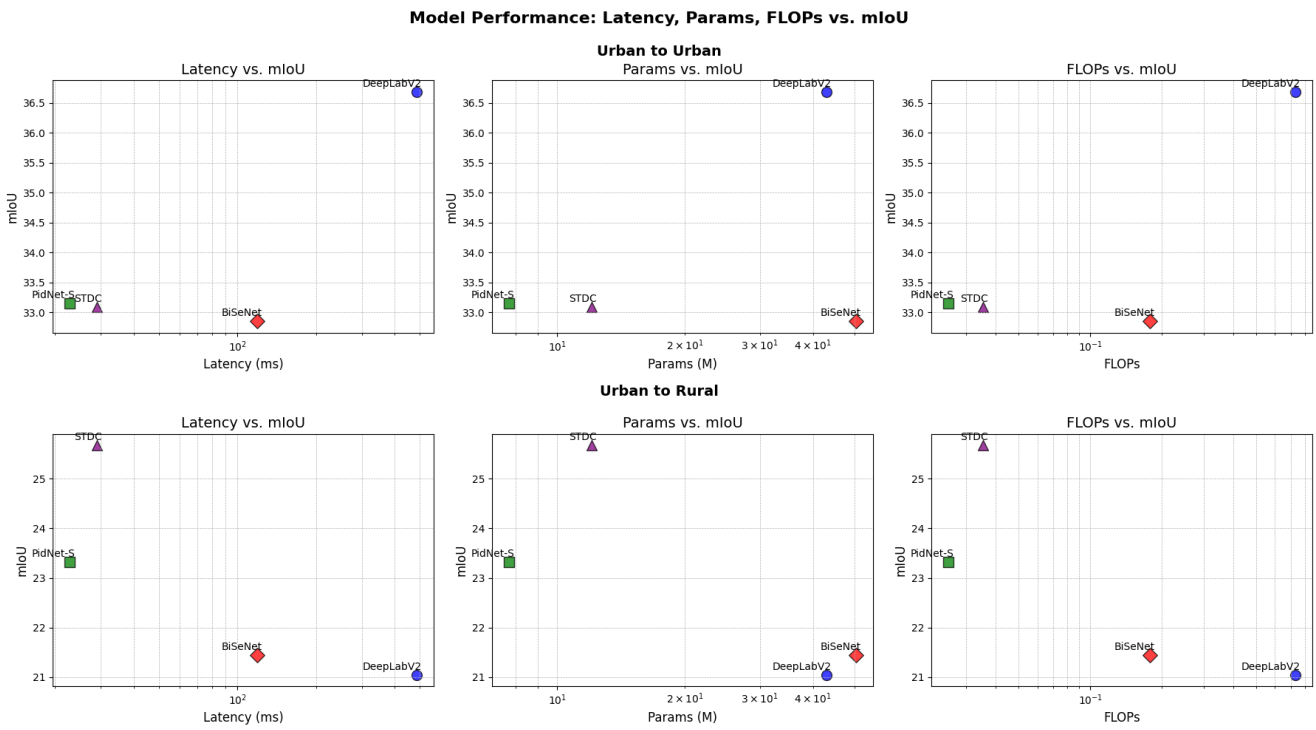

Figure 5. Model comparison, first row displays results on the same domain (Urban), second row displays results in domain shift

Model	Source	Target	Augmentations	Domain Adaptation	Criterion	Best mIoU (%)
DeepLabv2	Urban	Urban	-	-	Cross Entropy	36.68
DeepLabv2	Urban	Rural	-	-	Cross Entropy	21.05
PIDNet-S	Urban	Urban	-	-	Cross Entropy + Boundary	33.15
PIDNet-S	Urban	Urban	-	-	Weighted CE + Boundary	32.78
PIDNet-S	Urban	Urban	-	-	OHEM + Boundary	27.75
PIDNet-S	Urban	Urban	-	-	Focal + Boundary	28.50
PIDNet-S	Urban	Rural	-	-	Cross Entropy + Boundary	23.32
PIDNet-S	Urban	Rural	-	-	Weighted CE + Boundary	25.45
PIDNet-S	Urban	Rural	-	-	OHEM + Boundary	19.47
PIDNet-S	Urban	Rural	-	-	Focal + Boundary	21.20
PIDNet-S	Urban	Rural	Random Crop	-	Cross Entropy + Boundary	28.16
PIDNet-S	Urban	Rural	Random Crop	-	Weighted CE + Boundary	27.82
PIDNet-S	Urban	Rural	Random Crop	Adversarial	Cross Entropy + Boundary	25.50
PIDNet-S	Urban	Rural	Random Crop	Image-to-Image	Cross Entropy + Boundary	26.17
BiSeNet	Urban	Urban	-	-	Cross Entropy	32.85
BiSeNet	Urban	Rural	-	-	Cross Entropy	21.45
STDCNet813	Urban	Urban	-	-	Cross Entropy	33.09
STDCNet813	Urban	Rural	-	-	Cross Entropy	25.67

Table 3. Results of different experiments

References

[1] Reza Azad, Moein Heidary, Kadir Yilmaz, Michael Hüttemann, Sanaz Karimijafarbigloo, Yuli Wu, Anke Schmeink, and Dorit Merhof. Loss functions in the era of semantic segmentation: A survey and outlook, 2023. 2, 3

[2] Konstantinos Bousmalis, Nathan Silberman, David Dohan,

Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks, 2017. 2

[3] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2):125, Feb. 2020. 5

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, 2017. 1, 2, 4

[5] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017. 2

[6] Jia Deng, Wei Dong, Richard Socher, Li Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009*, 2009 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pages 248–255, United States, 2009. IEEE Computer Society. Publisher Copyright: © 2009 IEEE.; 2009 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009 ; Conference date: 20-06-2009 Through 25-06-2009. 2

[7] Mingyuan Fan, Shenqi Lai, Junshi Huang, Xiaoming Wei, Zhenhua Chai, Junfeng Luo, and Xiaolin Wei. Rethinking bisenet for real-time semantic segmentation, 2021. 2

[8] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation, 2015. 2

[9] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks, 2016. 2

[10] Alessandro Giusti, Dan C. Cireşan, Jonathan Masci, Luca M. Gambardella, and Jürgen Schmidhuber. Fast image scanning with deep max-pooling convolutional neural networks, 2013. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition*, page 346–361. Springer International Publishing, 2014. 1

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 2

[13] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation, 2017. 2

[14] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation, 2016. 2

[15] M. Holschneider, R. Kronland-Martinet, J. Morlet, and Ph. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In Jean-Michel Combes, Alexander Grossmann, and Philippe Tchamitchian, editors, *Wavelets*, pages 286–297, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg. 1

[16] Yuanduo Hong, Huihui Pan, Weichao Sun, and Yisong Jia. Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes, 2021. 2

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. 1

[18] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials, 2012. 1

[19] George Papandreou, Iasonas Kokkinos, and Pierre-André Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 390–399, 2015. 1

[20] Rudra P K Poudel, Ujwal Bonde, Stephan Liwicki, and Christopher Zach. Contextnet: Exploring context and detail for semantic segmentation in real-time, 2018. 2

[21] Rudra P K Poudel, Stephan Liwicki, and Roberto Cipolla. Fast-scnn: Fast semantic segmentation network, 2019. 2

[22] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks, 2014. 1

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 1

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014. 1

[25] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. Dacs: Domain adaptation via cross-domain mixed sampling, 2020. 2, 4, 5

[26] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation, 2020. 2, 4, 5

[27] Junjue Wang, Zhuo Zheng, Ailong Ma, Xiaoyan Lu, and Yanfei Zhong. Loveda: A remote sensing land-cover dataset for domain adaptive semantic segmentation, 2022. 3

[28] Jiacong Xu, Zixiang Xiong, and Shankar P. Bhattacharyya. Pidnet: A real-time semantic segmentation network inspired by pid controllers, 2023. 2, 3, 4

[29] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation, 2018. 1, 2

[30] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020. 2