Analysis of loss function impact on Real-time Domain Adaptation in Semantic Segmentation

Allan Brunstein
s294072
Politecnico di Torino
Turin, Italy
s294072@studenti.polito.it

Marco Fabio De Souza Coelho
s303366
Politecnico di Torino
Turin, Italy
s303366@studenti.polito.it

***Abstract*—This report aims at clarifying what is the impact of the loss function in the accuracy of a Real-time Domain Adaptation in Semantic Segmentation model, using adversarial networks, and with class imbalance. By comparing three different loss types and comparing the results, it became clear that using different functions to calculate the loss while training produces severely different analysis.**

***Index Terms*—semantic segmentation, domain adaptation, loss function**

I. Introduction

The definition of Semantic Segmentation is associated with the task of separating image parts that belong to the same object class. In this process, each pixel of the input image is classified according to a predefined category, which can be asphalt, trees, cars, traffic signs, houses, etc.

To perform this task, the most common techniques currently train Convolutional Neural Networks (CNN's) with labeled data. After the training process, unlabeled data (new data) is applied to the model and the evaluation of the predictor's performance is calculated.

One of the biggest problems nowadays with machine learning is data availability. Due to the lack of properly labeled data it became very common to train machine learning algorithms in a particular source dataset while having to apply it in a different target dataset, this process is known as Domain Shift. The main difference is not on the classes of the classified objects, but it is on the small environmental characteristics that compose each dataset. A good example could be training a martian rover on earth's terrain and giving it real martial terrain images as input, even though there are many similarities between both terrains the machine learning model could struggle in outputting correct predictions.

In the case of our model, given that the source dataset and the target dataset share the same feature space (in this case, the same objects for detection), it is possible to use the same model trained on the source dataset to perform a semantic segmentation on the target dataset. The process of training a ML model to properly learn how to deal with domain shift is very challenging, and the technique being used by our group to try to deal with this issue is based on the use of an adversarial network.

In this project, the source images are going to be imported from the GTA5 dataset and the target domain is going to be constituted of the images imported from the Cityscapes dataset. The choice of the GTA5 dataset comes from the fact that computer generated images, such as the ones from the game, are much easier to label than real images. As a baseline, a segmentation network was trained exclusively on the Cityscapes dataset (without domain adaptation) and the resulting MIoU is going to be considered the upper bound for the results section of the model with the domain adaptation applied.

The accuracy and the performance of the algorithm was measured by the Mean Intersection over Union (MIoU), the Precision per Pixel accuracy, the number of parameters of the network and the number of floating point operations.

One really big issue about the datasets being used is the class imbalance. The goal of this project is to compare how different loss functions impact the overall performance and accuracy of the real time semantic segmentation algorithm with domain adaptation, given the very imbalanced datasets. The different types of loss used were, Cross-Entropy loss, Weighted Cross Entropy loss and the Focal Loss. All relevant parameters are discussed further in the report.

This project, with all of its files, folders and contents, was uploaded to the following GitHub repository:
https://github.com/marcotommasini/AdversarialSegNet.git

II. Related Works

This project was made possible due to the early contribution of several datasets and machine-learning models. Some important concepts to the understanding of this project are explained in this section.

A. datasets

1) Cityscapes: CityScapes is a large-scale dataset created in order to train and test approaches for pixel-level and instance-level semantic labeling. Cityscapes is comprised of a large, diverse set of stereo video sequences recorded in the streets of 50 different cities. 5000 of these images have high quality pixel-level annotations and 20 000 additional images have coarse annotations to enable methods that leverage large

volumes of weakly-labeled data. All the data was acquired from the perspective of a moving vehicle during different periods of the year, covering spring, summer and fall in all cities. The total dataset contains images of 30 different classes, but for the purposes of our project only 19 different classes were used in total. The decision of these 19 classes was based on their frequency in the dataset.

2) GTA5: The GTA 5 dataset was created in order to more easily acquire labeled data, since all the images are computer generated it is much quicker to generate the semantic labels for those images. What makes this dataset useful is the fact that GTA5 is a realistic looking game with object displacements that imitate real life. The dataset is composed of around 25 thousand image samples with their respective segmented labels, and the same classes as the CityScapes dataset.

B. Models

1) BiseNet: BiseNet is a semantic segmentation algorithm that focuses on producing accurately segmented images in real time, the creation of this model was inspired by the incapability of modern algorithms to produce accurate results in real time. The model consists basically of two convolutional paths, the Spatial path and the Context path. The objective of this separation is to deal with the loss of spatial information and the shrinkage of the receptive field. For the architecture of the Spatial path it was used three convolutional layers in order to obtain feature maps with 1/8 of the original size, the Context path on the other hand consists of a pooling layer appended to the end of the pretrained network. The pretrained networks chosen for our experiments were the Resnet101, and the Resnet18. The main object of these pretrained networks is to downsample the feature in order to obtain a large receptive field, which will encode high level information about the input images. In order to properly connect the results of these networks the researchers also created Feature Fusion Module, and the Attention Refinement Module. The pretrained networks were trained on the ImageNet dataset.

2) Adversarial network: The main objective of the adversarial network is to be able to identify from the labeled data, if the data came from the source or the target domain. In order to accomplish this goal the network was built with 5 convolutional layers in sequence and a "Leaky Relu" activation function on the last layer. The results of the network are categorical with 0 being "Source" and 1 being "Target".

3) ResNet: In the context of our project, the Resnets are only being used as a backbone for the more robust BiseNet model. There are many different types of Resnet, the ones that are being used are the Resnet18 and the Resnet101. The Resnets are a set of image classification models based on deep convolutional neural networks. The main idea behind the creation of the resnet was to generate a model that could, with many less convolutional layers, outperform current deep learning models. The proposed model aims to fix the accuracy degradation in deep convolutional networks, and prevent the problem of the vanishing gradient. The model is based on a series of convolutional blocks, pooling blocks, and "shortcut" connections that link directly the input of a block to the output of another block.

C. Convolution

1) Depth-Wise Separable Convolution: An image convolution is the process of adding each element of the image to its local neighbors by the means of the sum of multiplication of the kernel by the original values of the image matrix. This operation is the main concept behind any convolutional block of a neural network. The problem involved with this method is that, when very deep neural networks are being used, the number of operations becomes too high. A solution to this problem is to use a different convolution technique called Depth-Wise separable convolution. This operation works by splitting the convolution process into multiple smaller convolutions. The output of this type of convolution is the same as the original technique, and the only difference is that the number of operations performed is much smaller, saving time and computational power.

III. Methodology

A. Image Preparation

Before the image could be used in the network, some preparation was needed to adapt it. The data cleaning and preparation part was based on resizing the image into 1024x512 instead of the original 2048x1024, and then converting it into a float Tensor, which was the default data type for this project. In order to acquire the images from the image source and resize them, a dataset class was used. This dataset object was then used to create the Data Loader object, which will store all the information separated in batches and would define other parameters like number of workers. The Data loader object is iterable, and was used as input to train the network.

B. Training Phase

The training process of this model was pretty similar to the training of a non-adversarial model, data was used as input, and labels were used in order calculate the training loss.

To help better understand the training process it was decided to divide it mentally into two parts, segmentation network training and adversarial network training.

The first step of the segmentation training process was the forward propagation of the batch from the source dataset into the the segmentation network. A segmentation loss was then calculated based on these results and on the labeled data, this loss was later used to train specifically the segmentation part of the network.

The next step of the segmentation network training was the forward propagation of the unlabeled data from the target dataset. The results of this segmentation were then fed forward to the adversarial network, which produced a binary label describing the origin of the segmented image, 0 for source and 1 for target. The loss function used was the Binary Cross Entropy loss.

In order to properly train the full model into segmenting the CityScapes images as if they were from the GTA5 dataset,

it was needed to use a training technique that "fools" the discriminator into thinking that the images from the Target dataset were actually from the Source dataset. In order to produce these results, the adversarial loss was generated using the label belonging to the Source dataset, even though the actual image came from the Target dataset. This adversarial loss will then be summed to the segmentation loss via the function :

$$\mathcal{L}_{seg} = \mathcal{L}_{source} + \lambda_{adv} * \mathcal{L}_{target}$$

The "Lambda" parameter was used as a weight for the adversarial loss generated by the target batch in order to finely control how this loss would affect the final segmentation loss result.

The $\mathcal{L}_{seg}$ was backpropagated only to the segmentation part of the model, this was possible because the gradient calculations for the adversarial part were deactivated.

The second big step was to calculate the loss that would be back propagated only to the adversarial part of the model. In order to compute the $\mathcal{L}_{adv}$ loss, some procedures were also followed. The first thing that had to be done was to generate the labels for both source and target dataset batches using the segmentation network. These labels were then fed forward to the adversarial network, and the respective losses were calculated. These losses were calculated using the correct labels, source label for the segmented source images and target label for the segmented target images. The function used to calculate these losses was the Binary Cross-Entropy with Logits. The total loss that would be backpropagated was calculated using the arithmetic average between the loss of the source batch and the target batch.

$$\mathcal{L}_{D,adv} = \frac{\mathcal{L}_{D,source} + \mathcal{L}_{D,target}}{2}$$

After both losses were back propagated, the last mains thing to do was to calculate the step for the optimizer. For the optimizer it was decided to use the Stochastic Gradient Descent for the segmentation network, and the ADAM optimizer for the adversarial network.

This set of operations was done for each iteration of the training process. These steps were repeated until the maximum number of epochs was achieved, which could be 50 or 100 depending on the type of testing that was being done.

The following parameters were using for all experiments in the training phase:

- Number of classes: 19
- Initial value for the learning rates: $2.5 * 10^{-4}$
- Number of workers: 4
- Batch size: 8 for ResNet18, 4 for ResNet101
- $\lambda_{adv} = 10^{-3}$
- Optimizer: SGD (segmentation), Adam (adversarial)

C. Loss functions

The end goal of this project was to understand how different types of loss functions would affect the performance of the model when in the presence of a highly unbalanced dataset. In order to do the testings, three different types of loss function were used for the segmentation network. Those losses were, Cross Entropy, Weighted Cross Entropy, and Focal Loss. The losses will be explained, and results will be compared and analysed in the next section.

1) Cross entropy: For a specific number of classes n, the cross-entropy loss is defined as

$$L_{CE} = -\sum_{i=1}^{n} t_i * log_2(p_i)$$

where t_i is the truth label and p_i is the Softmax probability for the i-th class.

2) Weighted Cross Entropy: The Weighted Cross Entropy is a variation of the Cross Entropy loss that aims at solving class imbalance in the dataset issues. The bigger the relative frequency of a class in the training dataset is, the lower it's weight is going to be, thus making the machine-learning algorithm focus on the hardest classes.

$$L_{WCE} = -\sum_{i=1}^{n} w_i * t_i * log_2(p_i)$$

$$w_i = \frac{1}{log(1.02 + \frac{f_i}{\sum_{k=1}^{n} f_k})}$$

where w_i is the attributed weight for the i-th class and f_i is the count of pixels belonging to the i-th class in the training dataset.

3) Focal loss: The focal loss has the main goal of addressing the class imbalance issue, and this is done by applying a modulating term to the original cross-entropy loss.

$$FL = -\sum_{i=1}^{n} (1 - p_i)^{\gamma} * t_i * log_2(p_t)$$

The presence of this new term $(1 - p_i)^{\gamma}$ will decrease the load from well-classified classes substantially, depending on the chosen parameter $\gamma \geq 0$. In other words, this will make the model focus the training in the hardest classes. Therefore, in datasets where background classes are present in vast quantities, the focal loss will detect objects with much higher precision than the normal cross-entropy loss. In this project, the parameter γ was set to 1.

D. Baseline performance

The main goal of the use of the adversarial network in this project was to make it possible for the neural network to train the segmentation network into learning how to properly segment images of the real world without actually having real world labels. To properly understand how well the segmentation model was working when attached to the adversarial

network, a baseline result was needed for comparison. In this case the baseline results were the validation results from the BiseNet model trained on labeled real world data, the idea was to understand what would be the upper bound for the segmentation model performance.

The results of these experiments would be later compared in this paper as a way to understand how well our adversarial technique worked.

IV. Experiments

Legend: RN18: ResNet18, RN101: ResNet101, CE: Cross-Entropy, WCE: Weighted Cross-Entropy, FL: Focal Loss, NSEP: non-separable convolution, SEP: depth-wise separable convolution

A. Precision per pixel

ResNet	Epochs	Convolution	CE	WCE	FL
RN18	50	NSEP	0.567	0.502	0.574
RN18	100	NSEP	0.562	0.553	0.594
RN101	50	NSEP	0.628	0.543	0.608
RN101	100	NSEP	0.617	0.581	0.574
RN101	50	SEP	-	0.590	-
RN101	100	SEP	-	0.580	-

B. MIoU

ResNet	Epochs	Convolution	CE	WCE	FL
RN18	50	NSEP	0.164	0.190	0.210
RN18	100	NSEP	0.177	0.205	0.217
RN101	50	NSEP	0.222	0.234	0.259
RN101	100	NSEP	0.232	0.246	0.245
RN101	50	SEP	-	0.243	-
RN101	100	SEP	-	0.242	-

C. Results and Discussions

By observing the results from the experiments above, it is clear that the Focal Loss outperforms both the Cross-Entropy Loss and the Weighted Cross-Entropy Loss in most situations for this specific dataset. Noticeably, the class imbalance issue is a huge problem since there are only 19 different classes of detectable objects, and since the images are mostly urban landscapes, obviously roads, cars and other urban objects are going to be more frequent. Analyzing the accuracy measures, the Focal Loss is the better option to deal with this class imbalance.

More than that, the ResNet101 seemed to outperform the ResNet18 in terms of MIoU, however, in general the simulations were about 30% slower. Also, despite the fact that there wasn't great performance gain while using the depth-wise separable convolution in the adversarial network, the algorithm ran about 25% faster than normally, probably because the number of floating points operations and number of parameters was severely reduced. Although it was not seen such a harsh performance difference, the greatest benefit comes from the fact that the adversarial network became much lighter when the separable convolution was used instead of the standard convolution.

TABLE I
Number of parameters and FLOPs

Conv-Type	Network	FLOPs	PARAM
SEPARABLE	ADV	2.135.425.024	191.364
SEPARABLE	BiseNet	—-	12.581.672
STANDARD	ADV	30.878.466.048	2.781.121
STANDARD	BiseNet	—-	12.581.672

As it can be seen from the results, the number of floating point operations when using separable convolutions for the adversarial network is around 93.1% smaller than the number of operations done with the the standard convolution. This means that not only the network got lighter, it also executed almost 15 times less operations. The number of parameters also decreased significantly for the adversarial network.

Another important aspect of the project was to understand how the model trained using the adversarial technique compared to the model using only labeled data from the CityScapes dataset. The MIoU achieved for the model without the adversarial technique was .391 for the ResNet18, with .780 precision per pixel accuracy. For the ResNet101, the MIoU was .451, with .792 precision per pixel accuracy. All of these simulations were done using Cross Entropy loss. This results are important, because they can be considered the upper bound for this project, since the introduction of Domain Adaptation model can't perform better than the same model trained on the fully labeled dataset.

The highest MIoU value achieved in the simulations with the Domain Adaptation was .259, from the model with ResNet101, Focal Loss, Standard Convolution and number of Epochs equal to 50. It was expected that the difference between the source and target domains would naturally decrease the performance of the algorithm, and the group found that .259 is a solid performance given all the process and that the dataset consists of only 500 images, which is very limited. Had the dataset had more images the performance could possibly rise, but we cannot be sure due to the physical limitations imposed by the platform that was being used to run the algorithms.

V. Conclusions

After finishing the project the group concluded that one of the main issues while performing the Domain Adaptation task is the class imbalance, which can be addressed by changing several parameters of the algorithms. In the tested scenario it was observed that the loss function is a main parameter that can drastically increase both performance, accuracy and speed of the algorithm when correctly applied. Therefore, it became clear that for the real time domain adaptation semantic segmentation task, the Focal Loss produce generally better results than the Weighted Cross Entropy and the Cross Entropy functions, addressing the class imbalance task in a great way for a small dataset.

VI. Appendix

The following tables compare the MIoU metric per each class, in each simulation.

Legend: RN18: ResNet18, RN101: ResNet101, CE: Cross-Entropy, WCE: Weighted Cross-Entropy, FL: Focal Loss, NSEP: non-separable convolution, SEP: depth-wise separable convolution

TABLE II
RESNET18, 50 EPOCHS, NON-SEPARABLE CONVOLUTION

	CE	WCE	FL
Precision per pixel	56.66%	50.25%	57.42%
Average mIoU	16.41%	19.04%	20.99%
road	55.60%	38.14%	57.28%
sidewalk	19.27%	13.65%	19.57%
building	54.05%	61.13%	46.90%
wall	3.08%	12.50%	13.84%
fence	0.31%	10.04%	12.21%
pole	0.39%	7.60%	7.47%
light	0.00%	0.00%	6.30%
sign	0.02%	0.00%	0.53%
vegetation	67.48%	66.14%	71.23%
terrain	6.70%	9.36%	16.00%
sky	54.59%	53.30%	59.14%
person	0.82%	19.62%	19.67%
rider	0.05%	0.00%	0.04%
car	42.59%	54.10%	55.97%
truck	6.53%	8.18%	8.57%
bus	0.00%	7.25%	4.11%
train	0.00%	0.69%	0.00%
motorcycle	0.00%	0.01%	0.00%
bicycle	0.39%	0.00%	0.00%

TABLE III
RESNET18, 100 EPOCHS, NON-SEPARABLE CONVOLUTION

	CE	WCE	FL
Precision per pixel	56.21%	55.32%	59.39%
Average mIoU	17.66%	20.48%	21.66%
road	52.62%	51.68%	62.56%
sidewalk	20.35%	16.23%	23.44%
building	52.87%	61.24%	51.40%
wall	10.84%	12.19%	10.42%
fence	2.35%	11.11%	12.12%
pole	0.77%	7.87%	8.83%
light	0.00%	0.00%	8.87%
sign	0.06%	0.01%	1.72%
vegetation	68.79%	68.27%	70.82%
terrain	7.86%	10.05%	11.54%
sky	54.04%	57.44%	57.76%
person	12.91%	20.23%	19.87%
rider	0.30%	0.00%	0.01%
car	43.09%	57.20%	56.14%
truck	8.03%	8.87%	8.71%
bus	0.00%	6.72%	5.77%
train	0.00%	0.03%	0.00%
motorcycle	0.00%	0.00%	1.53%
bicycle	0.58%	0.01%	0.00%

TABLE IV
RESNET101, 50 EPOCHS

	WCE	(SEP) WCE	FL
Precision per pixel	54.32%	58.96%	60.78%
Average mIoU	23.41%	24.25%	25.89%
road	43.48%	53.80%	59.08%
sidewalk	19.92%	21.32%	27.04%
building	46.39%	55.00%	52.49%
wall	13.36%	17.52%	18.01%
fence	18.24%	16.97%	12.93%
pole	10.43%	11.61%	14.46%
light	10.09%	8.55%	14.25%
sign	5.01%	0.12%	5.81%
vegetation	74.43%	78.13%	76.69%
terrain	15.74%	26.57%	29.28%
sky	64.50%	63.68%	63.05%
person	29.45%	28.58%	28.41%
rider	1.56%	0.11%	1.53%
car	62.76%	60.84%	65.27%
truck	10.89%	10.35%	10.17%
bus	10.26%	7.49%	6.22%
train	3.30%	0.00%	1.62%
motorcycle	5.05%	0.05%	5.66%
bicycle	0.00%	0.00%	0.01%

TABLE V
RESNET101, 100 EPOCHS

	WCE	(SEP) WCE	FL
Precision per pixel	58.13%	57.98%	57.44%
Average mIoU	24.62%	24.24%	24.46%
road	51.79%	51.82%	48.31%
sidewalk	21.77%	23.03%	24.53%
building	49.50%	50.18%	50.05%
wall	11.49%	12.97%	16.29%
fence	15.49%	11.83%	11.53%
pole	10.77%	12.09%	13.70%
light	10.53%	12.60%	12.27%
sign	5.57%	1.78%	3.98%
vegetation	77.94%	79.02%	77.29%
terrain	23.08%	30.23%	26.34%
sky	66.78%	63.06%	66.28%
person	28.91%	28.18%	27.14%
rider	1.66%	0.53%	0.78%
car	64.36%	61.47%	63.69%
truck	12.04%	11.55%	10.68%
bus	9.64%	7.61%	4.35%
train	1.79%	0.00%	1.12%
motorcycle	4.61%	2.58%	6.32%
bicycle	0.00%	0.00%	0.02%

References

[1] "Semantic segmentation." https://paperswithcode.com/task/semantic-segmentation.

[2] Wikipedia contributors, "Domain adaptation — Wikipedia, the free encyclopedia," 2022. [Online; accessed 13-July-2022].

[3] "Real-time semantic segmentation." https://paperswithcode.com/task/real-time-semantic-segmentation.

[4] K. E. Koech, "Cross-entropy loss function." https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e, Nov 2021.

[5] "Papers with code - focal loss explained." https://paperswithcode.com/method/focal-loss.

[6] Z. Zhou, H. Huang, and B. Fang, "Application of weighted cross-entropy loss function in intrusion detection." https://www.scirp.org/journal/paperinformation.aspx?paperid=113008, Nov 2021.

[7] S. Hao, Y. Zhou, and Y. Guo, "A brief survey on semantic segmentation with deep learning," *Neurocomputing*, vol. 406, p. 302–321, 2020.

[8] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," *Computer Vision – ECCV 2018*, p. 334–349, 2018.

[9] S. Zhao, X. Yue, S. Zhang, B. Li, H. Zhao, B. Wu, R. Krishna, J. E. Gonzalez, A. L. Sangiovanni-Vincentelli, S. A. Seshia, and et al., "A review of single-source deep unsupervised visual domain adaptation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, p. 473–493, 2022.

[10] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," *Computer Vision – ECCV 2016*, p. 102–118, 2016.

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

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.