Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

Université de Carthage

Ecole Polytechnique de Tunisie

وزارة التعليم العالي و البحث العلمي

جامعة قرطاج

المدرسة التونسية للتقنيات

Engineering Internship Report

Semi-supervised semantic segmentation With High-and Low-Level Consistency

---

From 15th June 2021 to 15th august 2021.

Host Organization: Concordia University.

Elaborated by: Ines KOTTI

3rd Year Engineering Student - Option SISY

Supervised by: M. Vahid KHORASANI GHASSAB.

Academic Year: 2021-2022

ABSTRACT

One of the important aspects of machine learning is the ability to understand visual information from limited labeled data. Image-level classification has been extensively studied in semi-supervised settings, but high-density pixel-level classification with limited data has recently attracted attention.

In this project, we have implemented semi-supervised semantic segmentation using an adversarial network with semi-supervised classification. When training with few labels, the dual-branch technique eliminates both low-level and high-level errors.

Keywords: Computer vision, semi-supervised learning, semantic segmentation, generative adversarial networks

ACKNOWLEDGEMENT

I would like to express my deepest and most sincere appreciation and admiration to my Tutor and Internship Supervisor, whose efforts and guidance have helped me to dig deep into myself and push my limits through all the situations that I encountered.

CONTENTS

LIST OF FIGURES

GENERAL INTRODUCTION

Deep learning has been very successful in working with images as data, and it's now at the point where it can outperform humans in several use cases. the most critical challenges that individuals wished to solve with computer vision are image classification, object recognition and segmentation in an increasing order of difficulty in presenting an image. When it comes to object recognition, we take it a step further and try to use bounding boxes in conjunction with all the items in a picture to figure out where they are. By attempting to pinpoint the exact boundary of the items in the image, image segmentation takes it to a new level.

Semantic segmentation is one of the key computer vision tasks important in a variety of applications such as autonomous driving, medical imaging, and robotics. For example, it is critical in self-driving cars and robotics because the models must grasp the context of the environment in which they operate.

It is the technique of assigning a class label to each pixel in an image. These labels could include a person, car, flower, piece of furniture, etc., just to mention a few. The assignment of these semantic labels requires the precise definition of the outline of objects and therefore places much more stringent demands on the localization accuracy than other tasks for recognizing visual entities, such as image-level classification or bounding box-level detection.

The main problematic we have worked on during this internship is to create a semi-supervised semantic segmentation method that learns from a few pixel-wise labeled examples while also taking advantage of additional annotation-free images. The proposed approach relies on adversarial training with a feature matching loss to learn from unlabeled images. Chapter one is dedicated to the description of deep learning fundamentals. Chapter two is devoted to mathematical background. Later on, Chapter three focuses on semantic segmentation. After that, we focus on description of S4GAN and MLMT. Finally, some conclusions and future perspectives are drawn.

HOST ORGANIZATION PRESENTATION

Concordia University is a next-generation university that is continually rethinking higher education's future. Concordia University is a premier university in North America founded over the past 50 years and enrolling around 51,000 students annually through its innovative approach to experiential learning and cross-functional research. It is located in the vibrant and multicultural city of Montreal and was founded in 1974 following the merger of Loyola College and Sir George Williams University. Concordia University offers more than 100 undergraduate and graduate programs leading to degrees, diplomas, and certificates in four faculties, as well as a continuing education program, to students and faculty in an exciting academic and social atmosphere.

Figure 1 – Logo of Concordia University

During this internship I worked in the computer science research team headed by my supervisor Mr. Vahid Khourasani. As the center's name suggests, its main activities focus on the use of deep learning techniques to solve problems related to computer vision (specifically, understanding and using images, videos, digital documents, etc.).

CHAPTER 1

DEEP LEARNING FUNDAMENTALS

In this chapter, we will present the important deep learning fundamentals that will help us understand the work done during this report.

1.1 Deep Neural Network Concepts

Neural networks are a series of algorithms loosely modeled on the human brain that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering of raw inputs. The patterns that they recognize are digital, contained in vectors into which all real data, be it images, sounds, texts or time series, must be translated.

Neural networks help us to group and classify. You can think of it as a cluster and classification level in addition to the data that you store and manage. They help group untagged data based on similarities between the sample inputs, and they classify data when you have a tagged dataset to train with.

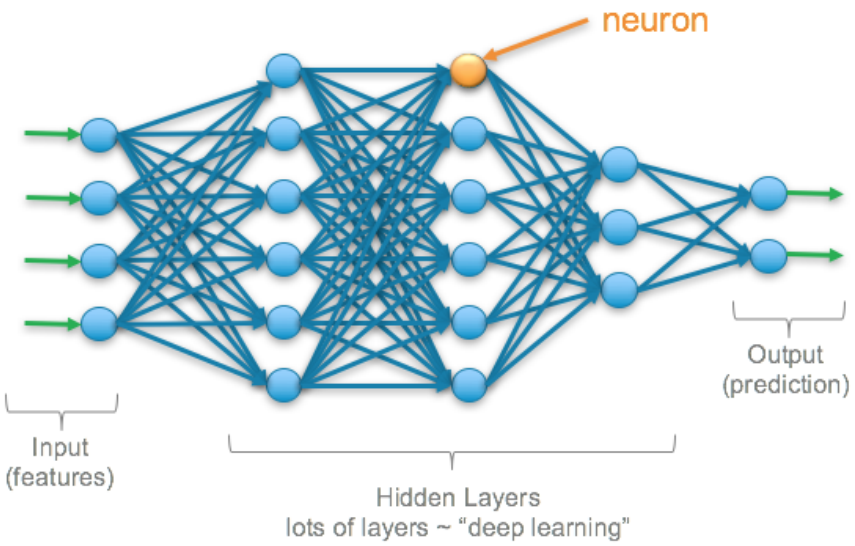


Figure 1.1 – Deep Learning Concepts

With Deep Learning, we are talking about algorithms capable of mimicking the actions of the human brain using artificial neural networks. Networks are made up of tens or even hundreds of "layers" of neurons, each receiving and interpreting information from the previous layer. So, each neuron, represented in the previous image by a circle, can be seen as a linear model. By interconnecting neurons as a layer, we turn our neural network into a very complex nonlinear model.

To illustrate the concept, let's take a dog-cat classification problem from an image. During training, the algorithm will adjust the weights of the neurons in order to reduce the difference between the results obtained and the expected results. The model will be able to learn to detect triangles in an image, since cats have much more triangular ears than dogs.

1.2 Convolutional Neural Networks

A convolutional neural network (CNN) which is a feedforward neural network, excels in image processing. Its artificial neurons may respond to surrounding units within the coverage range. It includes a convolutional layer, a pooling layer, a fully connected layer, batch normalization layer and dropout layer.

1.2.1 Convolutional layer

The convolutional neural networks composed of several convolutional units. The parameters of each convolutional unit are obtained by optimizing the back propagation algorithm. His purpose is to extract different input features. In fact, the first convolutional layer may extract only some low-level features such as edges, lines, and angles. However, a multi-layer network can extract more complex features based on the low-level features.

For simplification matters, we describe a 2D single-channel convolution with one filter. The filter is passed through the gray scale image using certain strides, and an element wise product followed by a sum of the products occurs between the bloc of the original image and the filter weights.

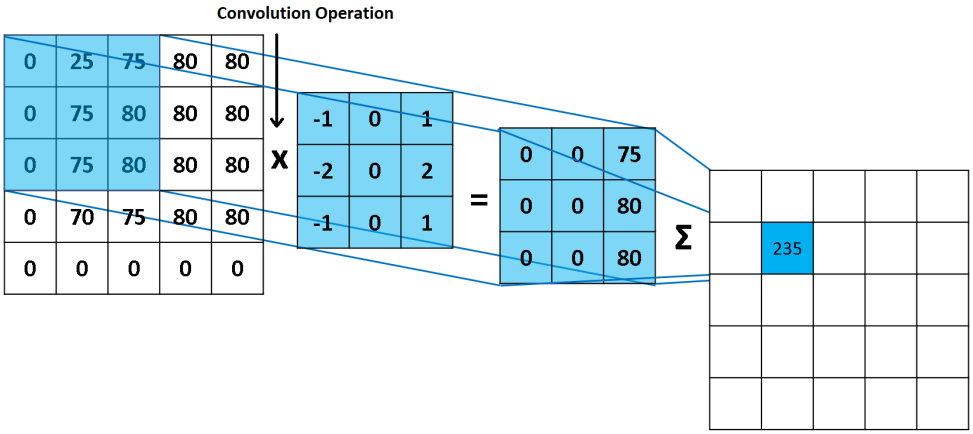


Figure 1.2 – Convolution procedure

1.2.2 Pooling layer

Pooling is a discretization method based on samples. Her purpose is to reduce the size of the input on the next layers. It has two types: max pooling and average pooling. When max pooling is used, the maximum value in a small square area is selected as the representative of this area, while the mean value is selected as the representative when average pooling is used. In fact, the side of this small area is the pool window size. The following figure shows the max pooling and average pooling operation whose pooling window size is 2.

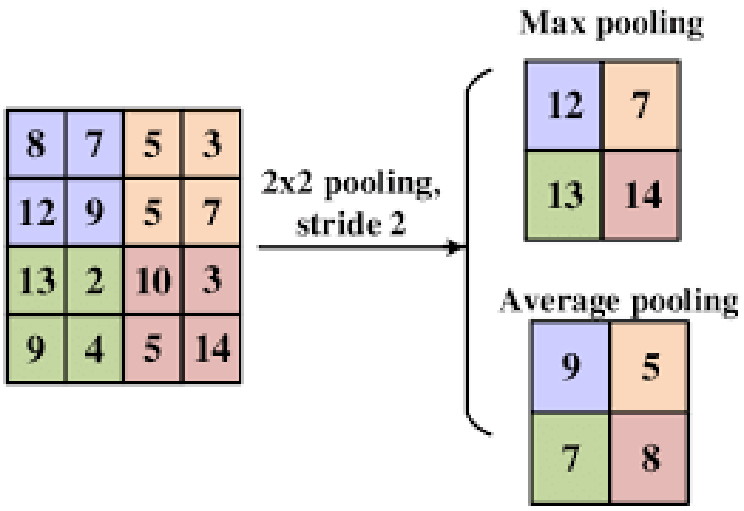


Figure 1.3 – Pooling operation

1.2.3 Fully Connected layers

These layers are placed at the end of the CNN architecture and are fully connected to all output neurons. After receiving an input vector, the FC layer successively applies a linear combination and an activation function in order to classify the input image. It finally returns a vector of size d corresponding to the number of classes in which each component represents the probability for the input image to belong to a class.

1.2.4 Batch normalization layer

Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. It is a process to make neural networks faster and more stable through adding extra layers in a deep neural network. The following figure shows a detailed explanation of batch normalization.

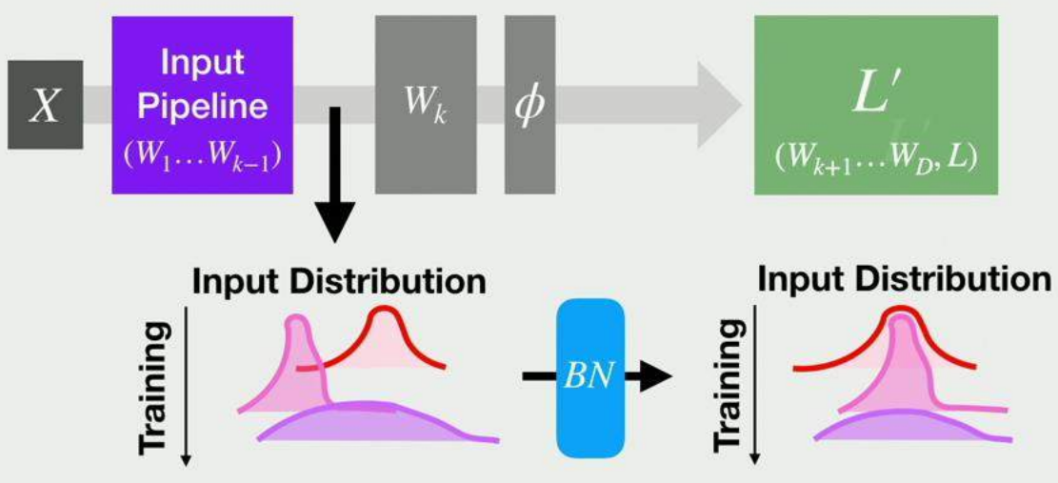
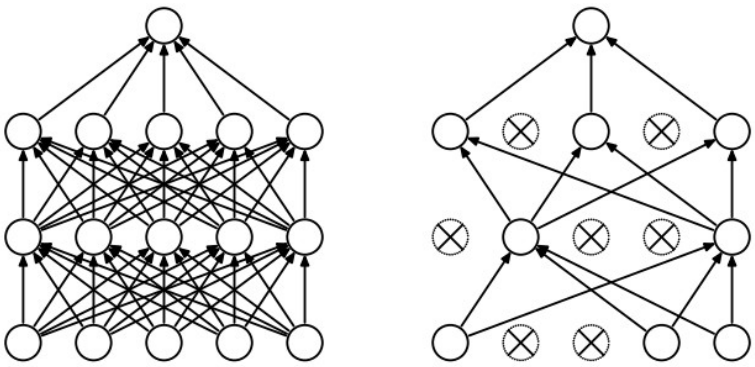


Figure 1.4 – Detailed explanation of batch normalization

1.2.5 Dropout layer

The dropout layer is a regularization technique used to reduce overfitting in artificial neural networks. Its use also worsens the training loss error.

During training, its functionality is to cause half of the neurons in a particular layer to be deactivated. This improves generalization because it forces your layer to learn the same "concept" with different neurons. And during the prediction phase, the dropout is deactivated.



(a) Standard Neural Net (b) After applying dropout.

Figure 1.5 – Illustration of the dropout technique

CHAPTER 2

MATHEMATICAL BACKGROUND

In this chapter, we will focus on the mathematical context. We will first describe semantic segmentation. Then, the Generative Adversarial Network and finally the Mean Teacher model.

2.1 Semantic Segmentation

We know that an image is nothing more than a collection of pixels.Image segmentation is the process of classifying each pixel of an image belonging to a certain class, and therefore can be viewed as image classification at a pixel level. For example, in an image that has many persons, segmentation will label all the objects as person objects.

Figure 2.1 – semantic segmentation operation

However, a separate class of models known as instance segmentation is capable of labeling separate instances where an object appears in an image. This type of segmentation can be very useful in applications used to understand objects, such as self-driving cars, which require a full understanding of their surroundings at a pixel perfect level. Hence, image segmentation is used to identify lanes and other necessary information.

2.2 Generative Adversarial Network

Generative adversarial networks are a model of data generation that can create a generative model of a base data set by using an adversarial game between two players. The two players correspond to a generator and a discriminator.

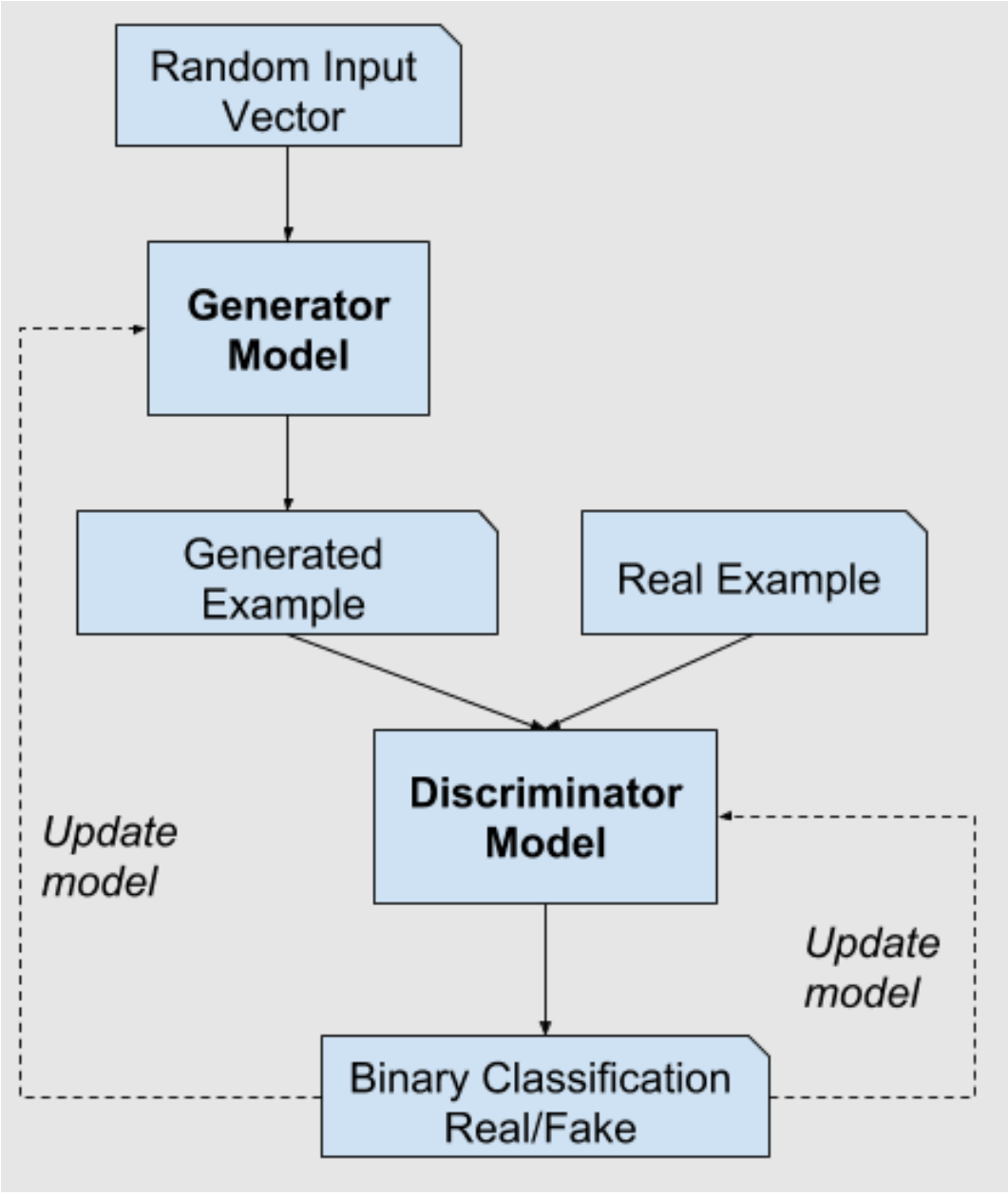


Figure 2.2 – Example of the Generative Adversarial Network Model Architecture

The generator takes the Gaussian noise as input and produces an output, which is a sample generated as the source data. The discriminator is usually a probabilistic classifier like logistic regression whose job is to distinguish the actual samples from the base data set and the generated sample. The generator tries to create samples that are as realistic as possible, its job is to trick the discriminator, while the discriminator's job is to identify false samples regardless of how the generator is trying to trick it. The problem can be understood as a contradictory game between generator and discriminator, and the formal optimization model is a minimax learning

problem. The equilibrium of this minimax game provides the final shaped model. Typically, this equilibrium point is the point at which the discriminator is unable to distinguish real samples from false ones.

2.3 Mean Teacher model

In the Mean Teacher model [1], we have two identical models that are trained with two different strategies, called the student and teacher model. In which, only the student model is trained. And, a very minimal amount of student model weight is assigned to the teacher model at each stage called exponential average weights, which is why we call it Mean Teacher. The ability to use abundant unlabeled data called semi-supervised learning is one of the main advantages of the average teacher.

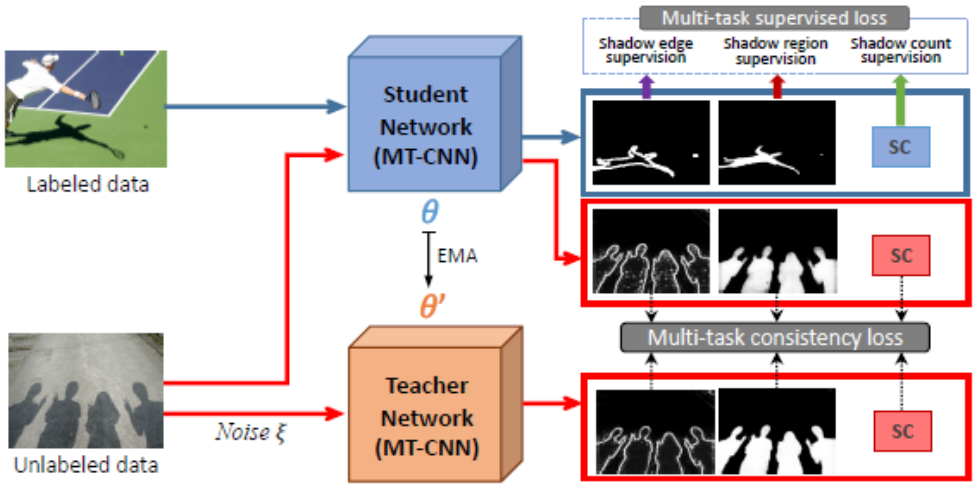


Figure 2.3 – Example of Mean Teacher Model

CHAPTER 3

SEMANTIC SEGMENTATION

In this third chapter, we will detail the architecture proposed by Sudhanshu Mittal, Maxim Tatarchenko and Thomas Brox to classify each pixel of an image belonging to a certain class and the employed architecture. Next, we will discuss the obtained results.

3.1 Work of Sudhanshu Mittal, Maxim Tatarchenko, and Thomas Brox

3.1.1 Overview of the architecture

In [2], Sudhanshu Mittal, Maxim Tatarchenko and Thomas Brox propose DeepLabv2 as the main of segmentation network which takes as input RGB images of size 256 * 512 pixels (see figure 3.1) and predicts C segmentation maps, one for each class. Figure 3.2 gives an overview of the architecture.

Figure 3.1 – Example of cityscape dataset

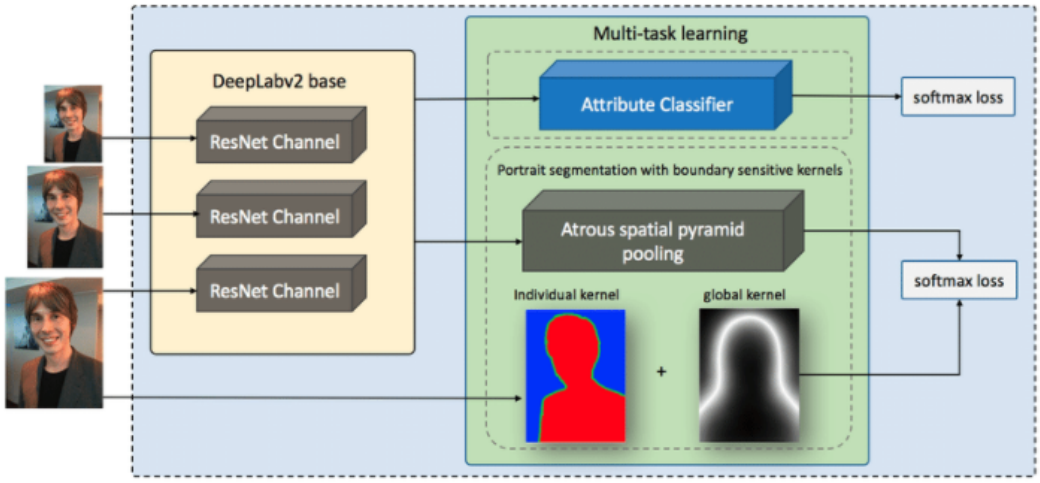


Figure 3.2 – Used architecture by Sudhanshu Mittal, Maxim Tatarchenko and Thomas Brox

3.1.2 Network description

The DeepLab-ResNet [3] is built on a fully convolutional variant of ResNet-101 with atrous (dilated) convolutions, atrous spatial pyramid pooling, and multiscale inputs. In fact, DeepLabv2 makes use of dilated convolutions to enlarge the receptive field size and incorporate larger context, and introduces atrous spatial pyramidal pooling to capture image context at multiple levels. So, the model is trained on a mini-batch of images and the corresponding ground truth masks, with the softmax classifier at the top. And during training, the masks are downsampled to match the size of the network output; during inference, to acquire the output of the same size as the input, bilinear oversampling is applied. The final segmentation mask is calculated using argmax on the logits.

3.1.3 Implementation details

They used the poly-learning policy for semantic segmentation model, where the base learning rate was multiplied by a factor of $\left(\left(1 - \frac{iter}{max_iter}\right)^{pow}\right)$ in every iteration. In their setup, pow = 0.9. For the optimizer, a stochastic gradient descent 'SGD' optimizer (see Appendix A) was used with a base learning rate of 2.5e-4,momentum 0.9 and a weight decay of 5e-4.

3.1.4 Training

The authors used a driving scene dataset, consisting of 2,975 densely annotated images and containing 19 classes : "road", "sidewalk", "building", "wall", "fence","pole", "traffic light", "traffic sign", "vegetation", "terrain", "sky", "person","rider", "car","truck","bus","train", "motorcycle" and "bicycle".

3.1.5 Performance

The proposed model achieved a 66% mIoU for full labeled data, 60.2% for 1/4 labeled data and 56.2% for 1/8 labeled data.

3.2 Discussion

The DeepLabv2 model produces false positives, predicting incorrect class labels even with full data annotation. therefore, due to memory constraints, they take turns to the discriminator to identify the best predictions.

3.3 Employed Architecture

3.3.1 Overview of the architecture

To create our model, we used PyTorch to implement U-NET model for semantic segmentation. In fact, U-NET is a symmetric architecture, as depicted below.

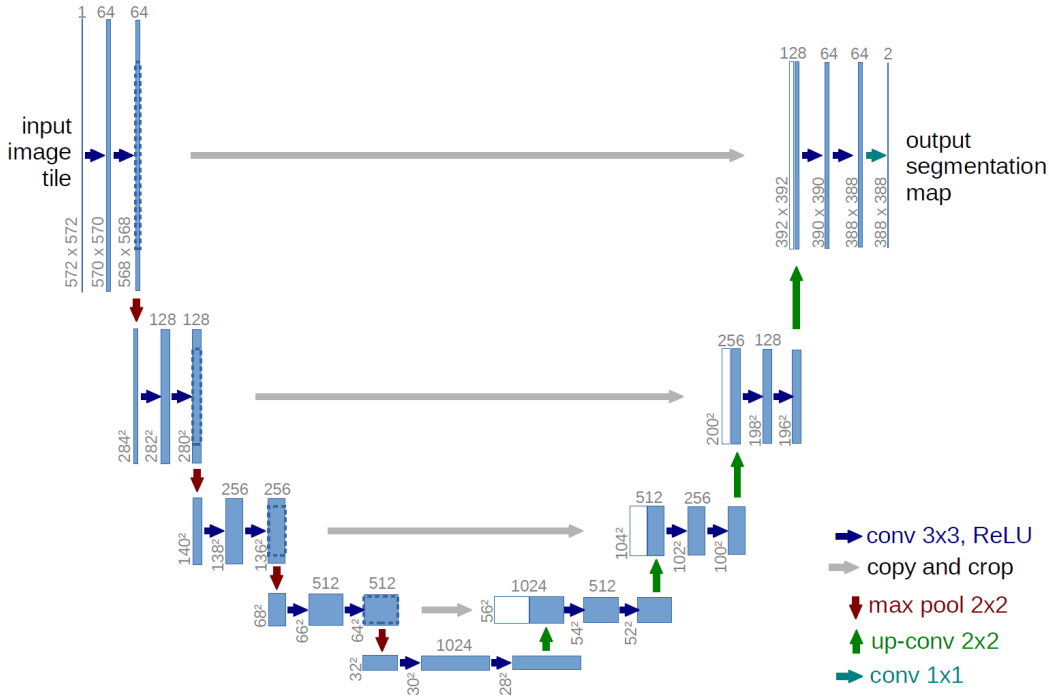


Figure 3.3 – U-NET Architecture

So, it can be divided into an encoder-decoder path or a contract-extensive path in an equivalent manner.

Encoder (left side): It consists of repeated blocks of two 3x3 convolutions. Each conv is followed by a ReLU and batch normalization. Then a 2x2 max pooling operation is applied to reduce the spatial dimensions. Again, with each step of downsampling, we double the number of feature channels, while halving the spatial dimensions.

Decoder path (right side): Each step of the decoder path consists of an oversampling of the feature map followed by a 2x2 transpose convolution, which halves the number of feature channels. Then, it is concatenated with the corresponding skip connection characteristic map of the encoder block. After that, a 3x3 convolution is used, where each convolution is followed by a ReLU activation function. Finally, the output of the last decoder goes through a 1x1 convolution with sigmoid activation, which is used to map the channels to the desired number of classes.

3.3.2 Implementation details

We used the following details when implementing our network. First, we applied DataLoader to our dataset to defines the number of samples that will be propagated through the network. we set up batch size equal to 5. In fact, typically, networks train faster with mini-batches. This is because we update the weights after each propagation. Second, we initiate our loss function and optimizer. we have used Adam optimizer (see Appendix A).

At last, our problem is classification of classes, therefore we have used a Cross Entropy Loss. The Cross Entropy Loss for the output S(x) of size H * W * C is evaluated only for the labeled samples x‘. It is calculated as shown in Equation,

$$L_{ce} = -\sum_{h,w_f} \mathbf{y}^{\ell}(h,w,c) \log S\left(\mathbf{x}^{\ell}\right)(h,w,c) \tag{3.1}$$

where y‘ is the ground-truth segmentation mask.

3.3.3 Metrics

To evaluate the performance of our model, we relied on the best by examining the degree of similarity of the output produced by such methods to the Ground Truth. This can be done in a mathematical way by calculating IoU (Intersection over Union) between the two. This procedure takes into account the common area (ground truth and predicted output) and calculates what percentage it is similar to the real one.

Because in our case, each object has to be given a different label and treated accordingly in the IoU calculation. We therefore propose a method that can take such cases into account and determine the total IoU for several classes present in one image. To do this, we need to find the mean of the IoUs corresponding to different classes that match the actual degree of similarity. This mean value is considered to be the Mean IoU (MIoU).

So, to calculate the MIoU [4], we need the labeled matrix of the predicted result and the expected result (ground truth) and follow the following steps:

Step 1 : Find the frequency number of each class for both matrices.

Step 2 : Convert the matrix to 1D format.

Step 3 : Find out the category matrix.

Such as, Category = (number of classes x actual 1D) + pred 1D

Step 4 : Construct the confusion matrix.

Step 5 : Calculate IoU for individual classes.

Such as, I = diagonal elements of confusion matrix (CM 2D)

U = actual count + pred count - I

Step 6 : Calculate MIoU for the actual-predicted pair.

3.3.4 Training

The model was trained and tested on Google Colab. Colabs' free GPU was used to speed up model training. We have built our model from scratch using cityscape dataset. For the dataset, we have split data as follows :

* 2,975 used to train the model.

* 500 used for validation.

* 1,525 used to test the model.

We have trained our model for 25 epochs. We have obtained the results displayed in Figure 3.4. As we can see in the beginning of the learning process, the loss reaches higher values (1.4). But, from the 10th epoch, the loss gradually stabilizes.

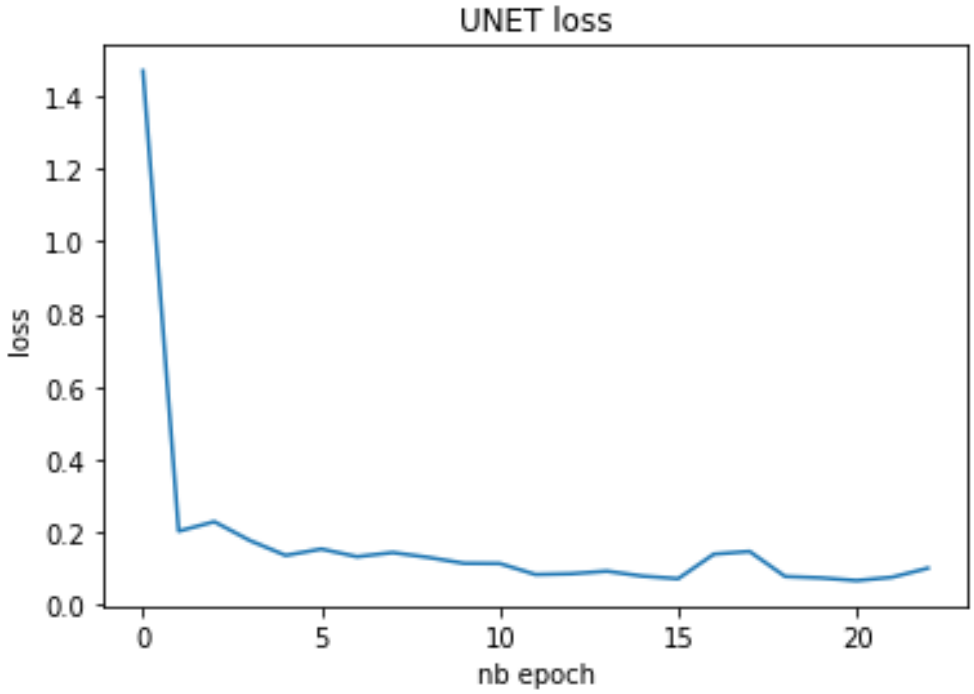


Figure 3.4 – Loss for UNET Model

3.3.5 Results and discussions

Our model has a mIoU of 86% on train, and when we have tested it on the test set it gives us a mIoU of 87.25%.

The figure above is a comparison between the one predicted from our semantic segmentation model and the labeled image.

Figure 3.5 – Original Image

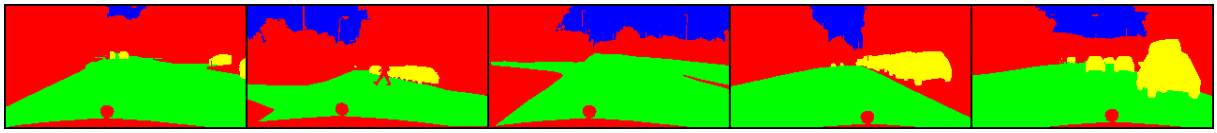


Figure 3.6 – Labeled Image

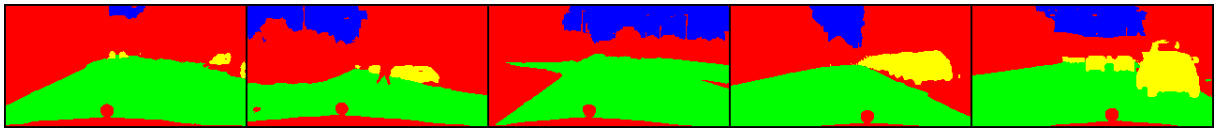


Figure 3.7 – Generated Image

With all data splits, we achieve improved results compared to the previous method. Our method achieves a performance increase of 30 percent compared to the initial value for various data divisions through the use of labeled samples. In particular, the approach works well for labeled data.

CHAPTER 4

S4GAN

In this chapter, we will detail the architecture proposed by Sudhanshu Mittal, Maxim Tatarchenko and Thomas Brox to improve the quality of semantic segmentation and the employed architecture. Next, we will discuss the obtained results.

4.1 Work of Sudhanshu Mittal, Maxim Tatarchenko, and Thomas Brox

4.1.1 Presentation of the discriminator architecture

The GAN model's discriminator network was a standard binary classification network. The network consists of 4 layers of convolution with 4 * 4 kernels with 64, 128, 256, 512 channels. Each layer is followed by a leaky ReLU activation with a negative slope of 0.2 and a dropout layer with a dropout probability of 0.5. The last convolution layer is followed by global average pooling and a fully connected layer.

4.1.2 Implementation details

The discriminator receives the chained input of the original image and its corresponding predicted segmentation. Its task is to compare the distribution statistics of the predicted and the real segmentation maps. Therefore, the output vector representation generated after global average pooling is used to assess the feature loss of match.

For the optimization method, they have worked with the Adam optimizer(see Appendix A).

At last, our problem is to differentiate between the real y‘ and the fake segmentation masks S(x) concatenated with the corresponding input images, that is why they chose the feature matching loss to be their loss function.

The feature matching loss function is calculated as shown in Equation 4.1.

$$L_D = \mathbb{E}_{(\mathbf{x}'\mathbf{y}^\ell)\sim\mathcal{D}^\ell}\left[\log D\left(\mathbf{y}^\ell \oplus \mathbf{x}^\ell\right)\right] + \mathbb{E}_{\mathbf{x}^u\sim\mathcal{D}^4}\left[\log\left(1 - D\left(S\left(\mathbf{x}^u\right) \oplus \mathbf{x}^u\right)\right)\right] \tag{4.1}$$

where $\oplus$ denotes concatenation along the channel dimension.

4.1.3 Performance

The proposed model achieved a 65.8% mIoU for full labeled data, 61.9% for 1/4 labeled data and 59.3% for 1/8 labeled data.

4.1.4 Discussion

we see that the performance is better for data labeled 1/8 and 1/4.

4.2 Employed Architecture

4.2.1 Overview of the architecture

In fact, the architecture of the discriminator that we built is inspired by the work of Sudhanshu Mittal, Maxim Tatarchenko and Thomas Brox. the only things we changed were the number of classes in the input. we have 4 classes.

Layer (type)	Output Shape
Conv2d-1	[-1, 64, 161, 396]
LeakyReLU-2	[-1, 64, 161, 396]
Dropout2d-3	[-1, 64, 161, 396]
Conv2d-4	[-1, 128, 80, 198]
LeakyReLU-5	[-1, 128, 80, 198]
Dropout2d-6	[-1, 128, 80, 198]
Conv2d-7	[-1, 256, 40, 99]
LeakyReLU-8	[-1, 256, 40, 99]
Dropout2d-9	[-1, 256, 40, 99]
Conv2d-10	[-1, 512, 20, 49]
LeakyReLU-11	[-1, 512, 20, 49]
AvgPool2d-12	[-1, 512, 1, 1]
Linear-13	[-1, 1]
Sigmoid-14	[-1, 1]

Figure 4.1 – Discriminator Architecture

4.2.2 Implementation details

We have used the following details during the implementation of our network. Firstly, we concatenated the original image with its mask on the one hand and with the image generated from the segmentation network on the other hand.

Secondly, in each Conv layer, we have used a Leaky ReLu as activation function. Since we have a binary classification problem, we have used a Sigmoid as an activation function for the output layer.

For the optimization method, we have worked with the Adam optimizer(see Appendix A).

At last, our problem is a binary class classification, that is why we chose the Binary Cross Entropy loss function. The BCE loss function is calculated as shown in Equation.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\left(p\left(y_i\right)\right) + \left(1 - y_i\right) \cdot \log\left(1 - p\left(y_i\right)\right) \tag{4.2}$$

where y is the label (1 for real image and 0 for fake image) and p(y) is the predicted probability of the image being real for all N images.

So, To improve the quality of semantic segmentation prediction, we update Unet model with unlabeled image. In fact, we trained the segmentation network with a discriminator responsible for distinguishing the ground truth segmentation maps from those generated. The segmentation network takes images and predicts segmentation maps. Next, the discriminator obtains the concatenated input of the original image and its corresponding predicted segmentation to predict if it is fake or real. After that, we update the segmentation network with the outputs of the discriminator.

4.2.3 Results and discussions

Our model has a mIoU of 87%. So, we notice that the result does not undergo much change. This is maybe due to the size of our labeled dataset. Since the size is big, we train our model on much example, which are big compared to the size of unlabeled dataset.

CHAPTER 5

MEAN TEACHER

In this chapter, we will detail the architecture used to classify the objects in the image. Then we will discuss the results obtained.

5.1 Presentation of the mean teacher architecture

We used ResNet-101 pre-trained on the ImageNet dataset as the base architecture for a Semi-Supervised Multi-Label Classification Network.

In fact, because of targets that change only once per epoch, the Temporal Ensembling becomes unwieldy when learning large datasets. So, we propose Mean Teacher, a method that averages model weights instead of label predictions, to solve this problem.

Our model evaluates each data point with and without noise, and then applies consistency costs between the two predictions. In this case, the model takes on a dual role as teacher and student. As a student, it learns as before; As a teacher, it generates targets that can then be used as a student to learn.

5.2 Implementation details

This model consists of two networks: a student network G and a teacher network H. Both networks receive the same images under different small perturbations. The teacher network weights (θ^0) are the exponential moving average (online ensemble) of the student network weights (θ). The predictions made by the student model are encouraged to be consistent with the predictions made by the teacher model using the loss of consistency, which is the mean square error between the two predictions.

We optimize the student network with the categorical cross-entropy loss L_{cce} for labeled samples x^l and with the consistency loss L_{cons} for all available samples ($x^{u;l}$)

$$L_{MT} = \underbrace{-\sum_{c} \mathbf{z}^{\ell}(c) \log\left(G_{\theta}\left(\mathbf{x}^{\ell}\right)(c)\right)}_{L_{cce}} + \lambda_{cons} \underbrace{\left\| G_{\theta}\left(\mathbf{x}^{(u,\ell)}\right) - H_{\theta}\left(\mathbf{x}'^{(u,\ell)}\right) \right\|^2}_{L_{cons}} \tag{5.1}$$

So, to extend a semi-supervised classification method, we followed the following steps:

Step 1 : Pass the image for preprocessing.

Step 2 : Reshape, crop, and normalize the input tensor for feeding into network for evaluation.

Step 3 : require Resnet to be put in evaluation mode in order to do prediction / evaluation.

Step 4 : Get the predictions of image as scores related to how the loaded image matches with 1000 ImageNet classes. In fact, the out variable is a vector of 1000 scores

Step 5 : Load the file containing the 1,000 labels for the ImageNet dataset classes.

Step 6 : Find the index (tensor) corresponding to the maximum score in the out tensor. In fact, Torch.max function can be used to find the information.

Step 7 : Find the score in terms of percentage by using torch.nn.functional.softmax function which normalizes the output to range [0,1] and multiplying by 100

Step 8 : Print the name along with partition of the object identified by the model.

Step 9 : Print the top 5 scores along with the image label. Sort function is invoked on the torch to sort the scores.

5.3 Results and discussions

This figure above is an example of input and output for mean teacher model.

Figure 5.1 – Example of input for MLMT

```
streetcar, tram, tramcar, trolley, trolley car 50.523040771484375
[('streetcar, tram, tramcar, trolley, trolley car', 50.523040771484375),
 ('traffic light, traffic signal, stoplight', 17.123807907104492),
 ('trolleybus, trolley coach, trackless trolley', 13.007492065429688),
 ('garbage truck, dustcart', 2.5044479370117188),
 ('street sign', 2.3374452590942383)]
```

Figure 5.2 – Example of output for MLMT

So, in the image above there are cars with score 50.52% and traffic light with score 17.12%. But, In certain situations our method produces imprecise predictions, for example, the street sign with score 2.33%.

Therefore, we don't have a more optimized version which has to take into consideration all the objects in the image. This means that we need more time to create our own model.

CHAPTER 6

GENERAL CONCLUSION AND FUTURE PERSPECTIVES

This report describes the various steps we took on our engineering internship project. It represents the host organization, where I have worked. Then we detailed the concept of the Deep Neural Network and explained our mathematical background to create a deep model that learn from limited pixel-wise annotated samples. Next, we relied on semantic segmentation as a tool to classify each pixel of an image belonging to a certain class. Finally, we focus on the mean teacher model to deal with high-level errors.

Through this project, we have shown that dense pixel-level classification with limited data is not an easy task, which explains why it is an active research topic. In fact, the results we have received are insufficient, but quite acceptable, to motivate future work on this topic

For future work, we want to create a network fusion, which is a fusion of spatial and class information to remove false positive class channels. Such an ambitious project will certainly require more time and significantly more resources.

BIBLIOGRAPHIE

[1] A. Tarvainen and H. Valpola. “Weight-averaged consistency targets improve semi-supervised deep learning results”. In: *CoRR* abs/1703.01780 (2017). arXiv: 1703.01780.

[2] S. Mittal, M. Tatarchenko, and T. Brox. “Semi-Supervised Semantic Segmentation with High- and Low-level Consistency”. In: *CoRR* abs/1908.05724 (2019). arXiv: 1908.05724.

[3] L. Chen et al. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *CoRR* abs/1606.00915 (2016). arXiv: 1606.00915.

WEBOGRAPHIE

[4] C. NITR. *MIoU Calculation*. URL: https://medium.com/@cyborg.team.nitr/miou-calculation-4875f918f4cb (visited on 05/09/2020).

Appendices

Academic Year 2020 – 2021

APPENDIX A

ADAM OPTIMIZER

Adam's optimization algorithm is an extension of stochastic gradient descent that has recently found wider application for deep learning applications in computer vision and natural language processing.

It attempts to calculate adaptive learning rates for each parameter. This is very useful in complex network structures because different parts of the network have different sensitivity to weight adjustment. A very sensitive part usually requires a smaller learning rate. It is difficult or complex to manually identify the sensitive part and set a learning rate. It may be the best optimizer at present. The Adam algorithm is detailed in the equations below :

$$m_t = \beta_1 m_{t-1} - (1 - \beta_1)\, g_t \tag{A.1}$$

$$v_t = \beta_2 v_{t-1} - (1 - \beta_2)\, g_t^2 \tag{A.2}$$

$$\hat{m}_t = \frac{m_t}{1 + \beta_1^t} \tag{A.3}$$

$$\hat{v}_t = \frac{v_t}{1 + \beta_2^t} \tag{A.4}$$

$$w_t = w_{t-1} - \eta \frac{m_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{A.5}$$

Where :

η : the step size.

β_1,β_2 : Exponential decay rates for moment estimate.

m_t : The first moment estimate (the mean).

v_t : The second moment estimate (the uncentered variance).

g_t : The gradient.

$\hat{m}_t$: The bias corrected estimators for the first moment.

$\hat{v}_t$: The bias corrected estimators for the second moment.

w_t : the model weight.

APPENDIX B

SGD OPTIMIZER

Gradient descent is one of the most popular algorithms to perform optimization, and by far the most common way to optimize neural networks.

It is a way of minimizing an objective function J (θ) parameterized by the parameters of a model R^d by placing the parameters in the opposite direction of the gradient of the objective function ∇_θ J (θ) updated to the parameters.

The reason for this is twofold: first, it reduces parameter update variance, which can lead to more stable convergence, and second, it allows the computation to take advantage of highly optimized matrix operations, which should be used in a well vectorized cost and gradient computation.

SGD, on the other hand, carries out a parameter update for each training example x (i) and label y (i):

$$\theta = \theta - \eta \cdot \nabla_\theta J\left(\theta; x^{(i)}; y^{(i)}\right) \tag{B.1}$$

Where :

the learning rate η determines the size of the steps we take to reach a (local) minimum.