IMAGE SEMANTIC SEGMENTATION

EZEQUIEL SARACENO
September 2022

Index

List of figures

Abstract

This paper details the extent of the background of the field of semantic image segmentation on a deep neural network based architecture. To this end, we have reviewed and summarized scientific papers. We have added to these papers a presentation of our models by explaining our approach and our results. In a first step, we explain why the field of image segmentation is related to convolutional networks by reviewing the state of the art and contextualizing it with the research, in a synthetic way and for our particular case. In a second step, we discuss the preparation of our data and some of the solutions we have developed in deep learning. Finally, we present the results of the architecture that stands out in our experiments, U-Net, which achieved XX% mIoU (mean of intersection over union) and mobilenet-Segnet, which achieved XX% mloU on the Cityscapes database.

Introduction

Deep learning has been very successful in working with images as data. It is now at a stage where it performs better than humans in many use cases. The problems that humans have wanted to solve with computer vision are image classification, object detection and segmentation, in increasing order of difficulty.

At the beginning, in the old image classification task, we were only interested in obtaining the labels of all the objects present in an image. In object detection we go a step further and try to know, along with all the objects present in an image, the location where the objects are present with the help of bounding boxes. Image segmentation takes it to a new level by trying to precisely find the exact boundary of the objects in the image.

In this technical paper, we aim to demonstrate how to **semantically segment** images using different models like **U-Net** and **Segnet** models, most of the common architectures used nowadays to solve most of the related tasks, using the cityscapes dataset.

Image Segmentation

Image segmentation is a computer vision task that segments an image into multiple areas by assigning a label to every pixel of the image. It provides much more information about an image than object detection, which draws a bounding box around the detected object, or image classification, which assigns a label to the object. Segmentation is useful and can be used in real-world applications such as medical imaging, clothes segmentation, flooding maps, self-driving cars, etc. There are two types of image segmentation:

- *Semantic segmentation:* process of classifying each pixel belonging to a particular label. It doesn't different across different instances of the same object. For example, in the image bellow, , semantic segmentation gives same label to all the pixels of both cats (same colour) and a different label to the dog (different colour).

- *Instance segmentation:* Instance segmentation differs from semantic segmentation in the sense that it gives a unique label to every instance of a particular object in the image. As can be seen in the image bellow all 3 animals (meaning cats and the dog) are assigned different colours (i.e different labels.) With semantic segmentation all of them would have been assigned the same colour.

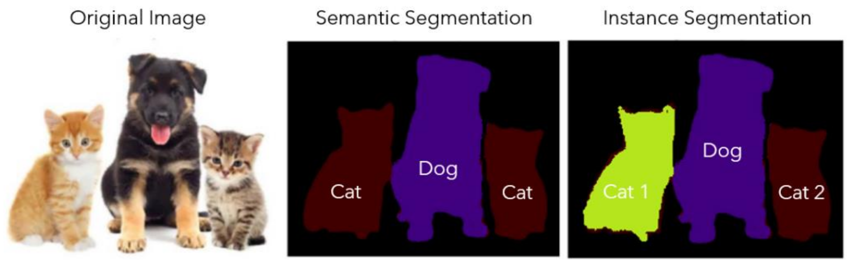


Figure 1. Examples of semantic and instance segmentation.

Dataset

Researcher usually teste their segmentation algorithms using different available datasets (PASCAL VOC, PASCAL Context, COCO, etc). For the purpose of this project we take advantage of one of the most popular dataset. The Cityscapes dataset has been released in 2016 and consists in complex segmented urban scenes from 50 cities (Figure 2). It is composed of 23.5k images for training and validation (fine and coarse annotations) and 1.5 images for testing (only fine annotation). The images are fully segmented with 32 classes (within 8 sub categories: flat, human, vehicle, construction, object, nature, sky, void). It is often used to evaluate semantic segmentation models because of its complexity. It is also well known for its similarity with real urban scenes for autonomous driving applications.

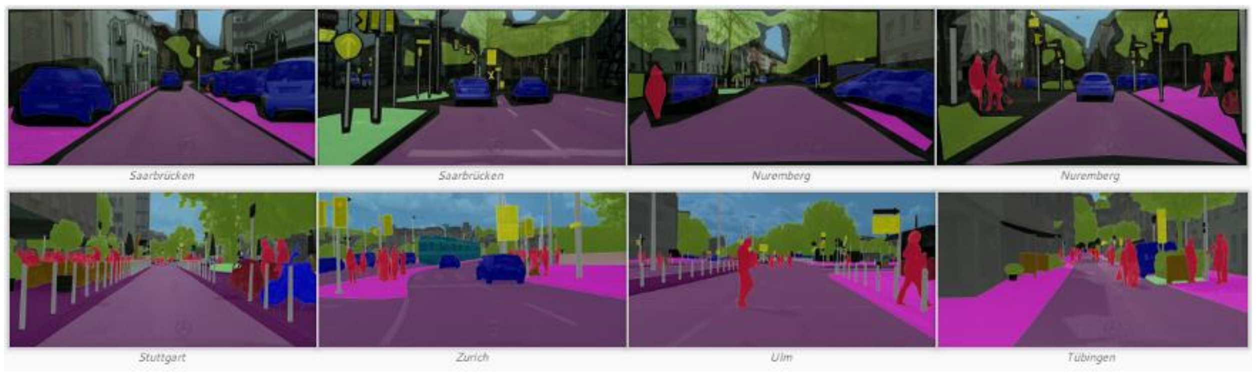


Figure 2. Examples of the Cityscapes dataset. Top: coarse annotations. Bottom: fine annotation.

State of the Art: Methods and techniques

Before the advent of deep learning, classical machine learning techniques such as SVM, Random Forest, K-means Clustering were used to solve the image segmentation problem. But as with most image-related problem statements, deep learning has outperformed existing techniques and has become a standard when it comes to semantic segmentation. Let's review the techniques that are used to solve the problem:

Convolutional Neural Network

In deep learning, a convolutional neural network (CNN) is a class of artificial neural network (ANN), most commonly applied to analyze visual imagery. Convolutional networks were inspired by biological processes where the connectivity pattern between neurons resembles the organization of the animal visual cortex. Mammals detect patterns regardless of their position in the image (translation invariance: pooling), and assign millions of neurons to this task for all pattern orientations, which are the contours of objects.

The application of CNN in semantic segmentation models has started with a huge diversity. Among different CNN based semantic segmentation models, Fully Convolutional Network (FCN) gained the maximum attention and an FCN based semantic segmentation model trend has emerged (Figure 3). To retain the spatial information of an image, FCN based models removed fully connected layers of traditional CNN. In studies and, the authors have used contextual features and achieved state of the art performance. Recently, in, the authors have used fully convolutional two stream fusion network for interactive image segmentation.

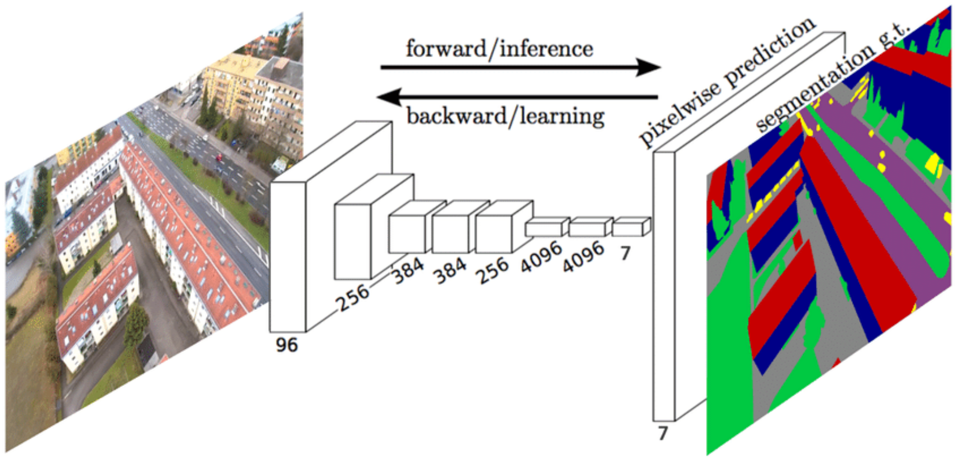


Figure 3. FCN architecture.

U-Net: U-Net is a U-shaped semantic segmentation network which has a contracting path and an expansive path. Every step of the contracting path consists of two consecutive 3 ×3 convolutions followed by ReLU nonlinearity and max-pooling using 2 × 2 window with stride 2. During the contraction, the feature information is increased while spatial information is decreased. On the other hand, every step of the expansive path consists of up-sampling of feature map followed by a 2×2 up-convolution. This reduces the feature map size by a factor of 2. Then the reduced feature map is concatenated with the corresponding cropped feature map from the contracting path. Then two consecutive 3 × 3 convolution operations are applied followed by ReLU nonlinearity. In this way, the expansive pathway combines the features and spatial information for precise segmentation. The architecture of U-Net is shown in figure 4.

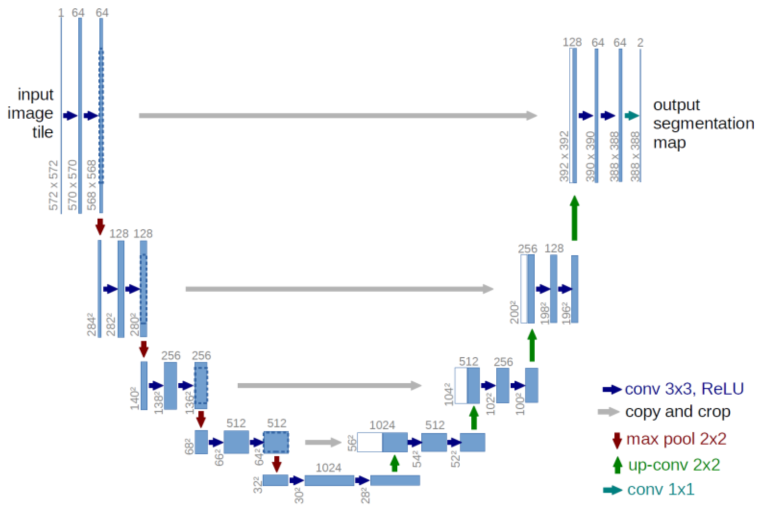


Figure 4. U-Net architecture.

SegNet: SegNet has encoder-decoder architecture followed by a final pixel-wise classification layer. The encoder network has 13 convolutional layers as in VGG16 and the corresponding decoder part also has 13 de-convolutional layers. The authors did not use fully connected layers of VGG16 to retain the resolution in the deepest layer and it reduces the number of parameters from 134M to 14.7M. In each layer in the encoder network, a convolutional operation is performed using a filter bank to produce feature maps. Then, to reduce internal covariate shift the authors have used batch normalization followed by ReLU nonlinearity operation. Resulting output feature maps are max-pooled using a 2×2 non-overlapping window with stride 2 followed by a sub-sampling operation by a factor of 2. A combination of max-pooling and sub-sampling operation achieves better classification accuracy but reduces the feature map size which leads to lossy image representation with blurred boundaries which is not ideal for segmentation purposes where boundary information is important. To retain boundary information in the encoder feature maps before subsampling, SegNet stores only the max-pooling indices for each encoder map. For semantic segmentation, the output image resolution should be the same as the input image. To achieve this, SegNet does up-sampling in its decoder using the stored max-pooling indices from the corresponding encoder feature map resulting high-resolution sparse feature map. To make the feature maps dense, the convolution operation is performed using a trainable decoder filter bank. Then the feature maps are batch normalized. The high-resolution output feature map produced form final decoder is fed into a trainable multi-class softmax classifier for pixel wise labeling. The architecture of SegNet is shown in figure 5.

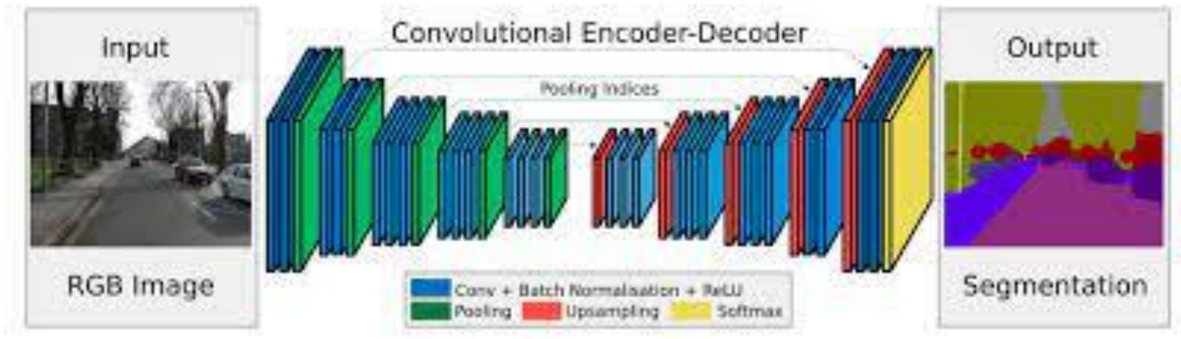


Figure 5. Segnet architecture.

Data augmentation

These types of networks require a lot of training data to get good results and avoid overfitting. However, it is often very difficult to get enough training samples. Image augmentation is a process of creating new training samples from existing ones. To create a new sample, you modify the original image slightly.

In order to perform the data augmentation we performed image retouching for all the images of the training. We have tested several libraries before choosing Albumentation. We take advantage of Albumentations, which is a fast and flexible image augmentation library. The library is widely used in industry, deep learning research, machine learning competitions and open source projects. Albumentations is written in Python, and it is licensed under the MIT license. Here are some examples of transformations of the original image that will create a new training sample (Fig 6).

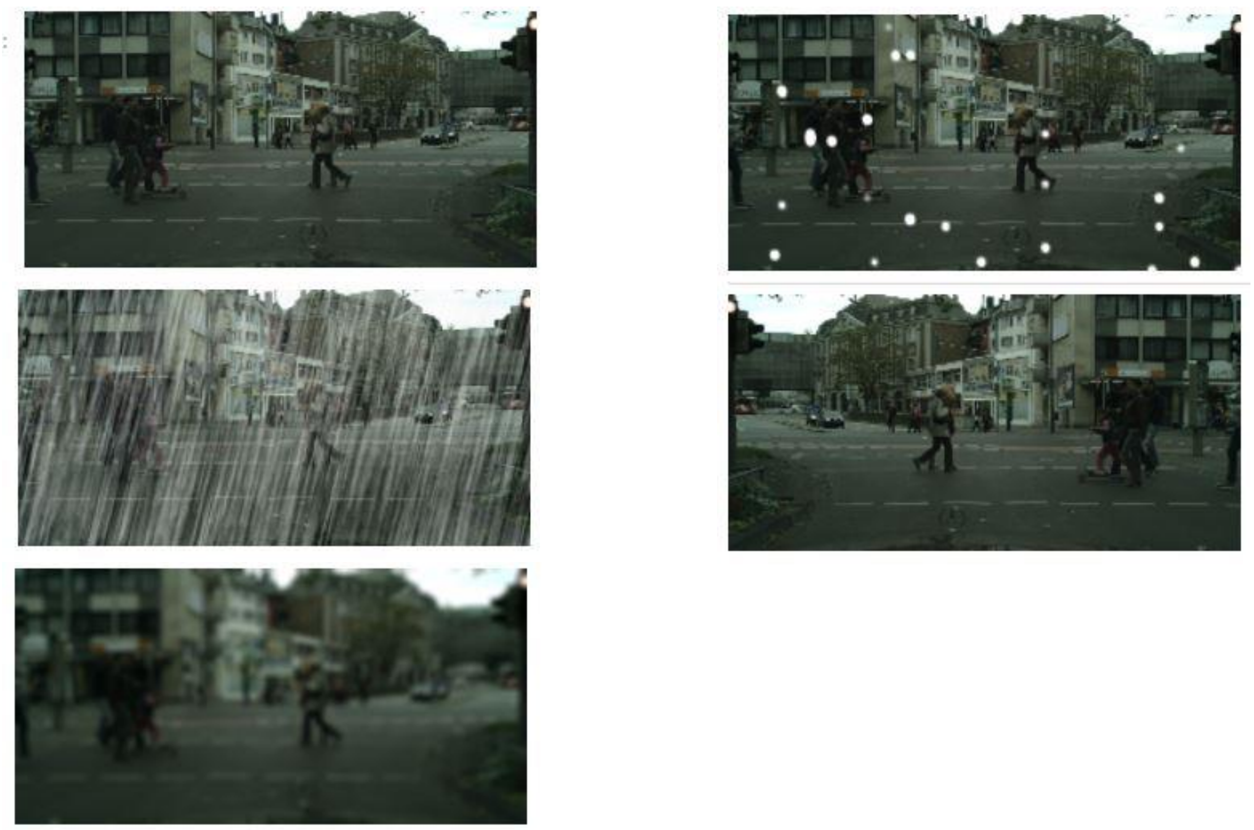


Figure 6. Examples of augmentation techniques. Top left: original image; top right: snowflakes; middle left: rain; middle right: flip; bottom left:blur

Data Generator

Due to the need for tons of data (in this case, images) to enter this type of network, we have been faced with the problem that the data does not fit in memory, so, we have to use some sort of data generator that creates data in batches and feeds it to our network for entry. In our case, we build a custom data generator, and we have to inherit from the Sequence class, which requires us to implement three methods:

- __len__ must: which returns the number of batches per epoch.

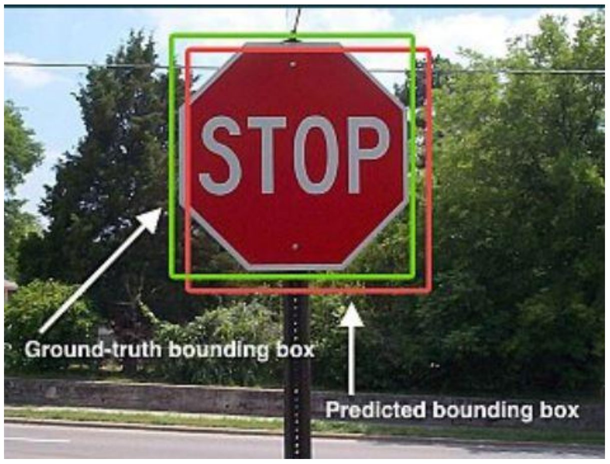
- on_epoch_end: which shuffles the indexes for training if shuffle=True.
- __getitem__: which returns a batch of images and masks if we make predictions.

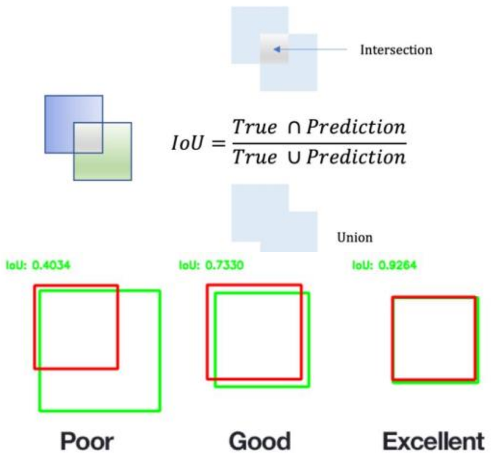
Loss and Metrics

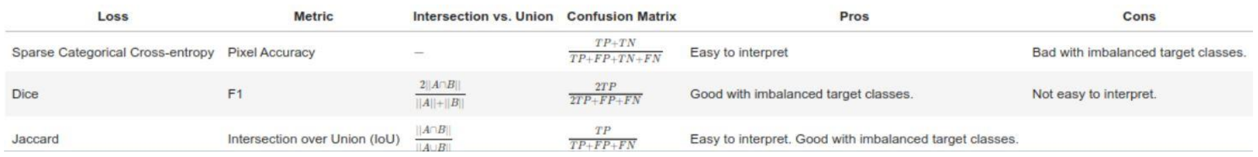
The performance measures must be defined before the training data is fitted. Deep learning neural networks are trained using the stochastic gradient descent optimization algorithm. As part of the optimization algorithm, the error for the current model state must be estimated repeatedly. This requires the choice of an error function, conventionally called the loss function, which can be used to estimate the loss of the model so that the weights can be updated to reduce the loss in the next evaluation.

Neural network models learn a mapping from inputs to outputs from examples and the choice of loss function must match the framing of the specific predictive modeling problem. For our problem, we will use cross-entropy, which is the default loss function to use for multi-class classification problems. Cross-entropy computes a score that summarizes the average difference between the actual and predicted probability distributions for all classes in the problem. The score is minimized and a perfect value of Cross-entropy is 0.

In terms of metrics for performance evaluation, since image segmentation involves assigning a class to each pixel in an image, a standard indicator of success is the intersection over union (IOU) coefficient. The IOU or Jaccard index measures the number of overlapping pixels between the actual and predicted masks divided by the total number of pixels in the two masks. Expression 3 (Fig 7) computes the IOU. In our approach, we used TensorFlow's MeanIoU function, which computes the average Intersection over Union for a sample of object detection results, and we will use the instance ids to construct the ground truth images.







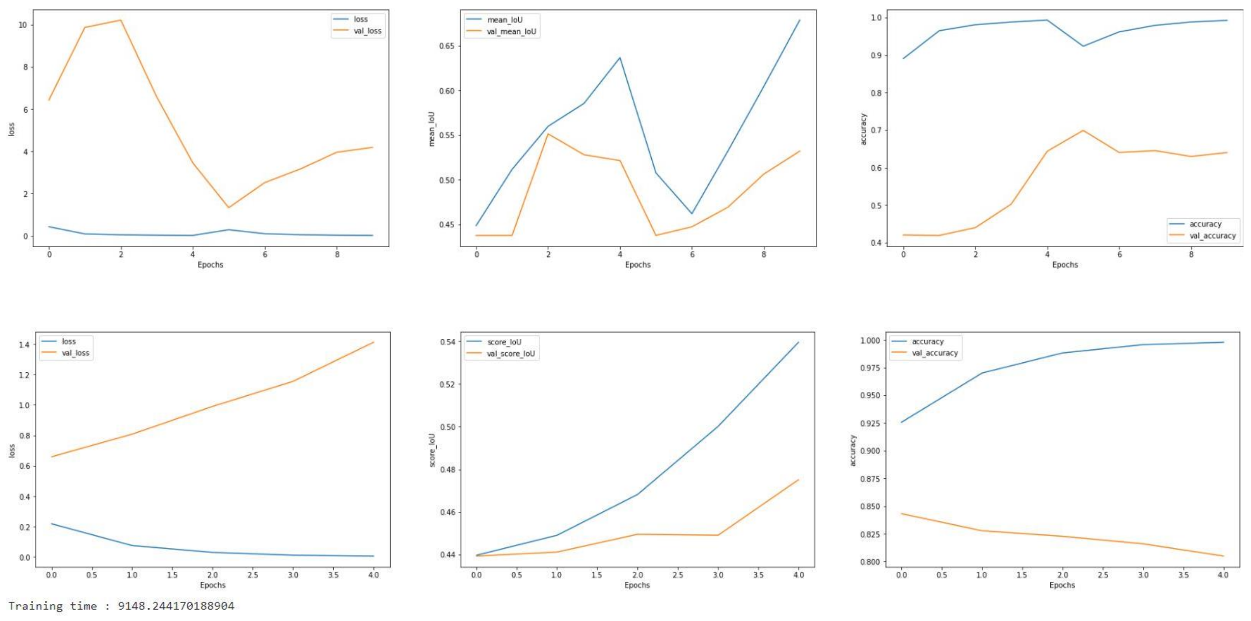
Loss	Metric	Intersection vs. Union	Confusion Matrix	Pros	Cons
Sparse Categorical Cross-entropy	Pixel Accuracy	–	$\frac{TP+TN}{TP+FP+TN+FN}$	Easy to interpret	Bad with imbalanced target classes.
Dice	F1	$\frac{2\|\|A \cap B\|\|}{\|\|A\|\|+\|\|B\|\|}$	$\frac{2TP}{2TP+FP+FN}$	Good with imbalanced target classes.	Not easy to interpret.
Jaccard	Intersection over Union (IoU)	$\frac{\|\|A \cap B\|\|}{\|\|A \cup B\|\|}$	$\frac{TP}{TP+FP+FN}$	Easy to interpret. Good with imbalanced target classes.	

Figure 7. Examples of metrics

Comparison

We worked on two kinds of algorithms. The first one, simple and classical, is an R-CNN network, based on the model of U-Net and Xception. It uses the structure of the model, but without the pre-trained model. The second one, Segnet, does not use the skip connections. After training the models, we test them in our validation dataset and create a figure showing two linear plots:

- The first with the cross-entropy loss over epochs for the train (blue) and validation (orange) datasets,
- The second plot showing the mean_IoU as a measure of performance, and
- the third plot shows the classification accuracy over epochs.



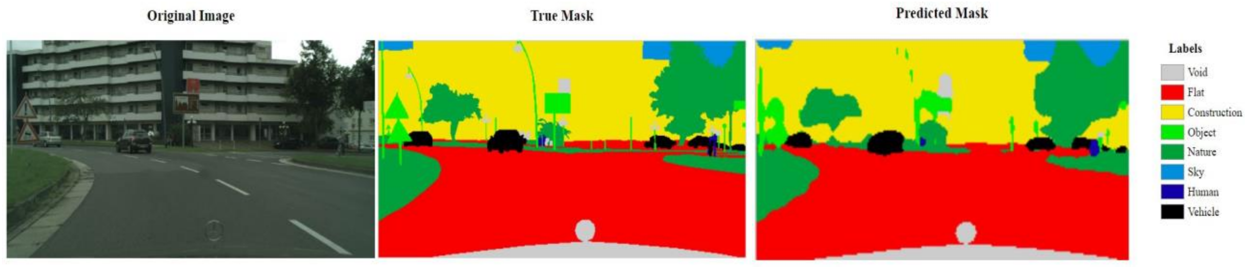
```
Training time : 9148.244170188904
```

Figure 8. Loss and metrics graphs from U-Net and Segnet respectively.

In both cases, the plots show that the models do not appear to have converged. The curves for cross-entropy, average IoU, and accuracy show poor, somewhat erratic convergence behavior. This could mean that the training was done with a lot of noise (sign of over or under fitting). To improve this situation, we can increase the epochs and the training rate or dataset size can be adjusted to even out the convergence in these cases.

Web App deployment

For the deployment of the application, we used Docker. Docker allows you to embed an application in one or more software containers that can run on any machine server, whether physical or virtual. Docker works on Linux as well as Windows Server. We created a Docker image, which we added to a container registry in Azure. This allowed us to copy the repo or flask application into Docker, and in this way, creating an application web service using Azure. The figure 9 shows one example of how our trained model perform a



mask prediction.

Figure 9. Example of our model prediction.

Conclusions

In this article, we have presented our different approaches, U-Net with Xception and SegNet-mobilenet. In addition, we have reviewed the history of convolutional networks from the last century to the present day. We then focus on the construction of an augmented database and the significant improvement it provides for model learning. We were able to evaluate the performance of different image segmentation models, with different parameters, and deploy the best model in production. For the next steps, we can train our model on many more images to improve the metrics and optimize the different optimization parameters.