AI Engineer PATH

Project 9

Build a computer vision system for an autonomous vehicle

Axel Favreul

Contents

1. Overview

2. Architecture

3. Training

4. Conclusion

01

Overview

Introduction

Future vision transport is a company that designs onboard computer vision systems for autonomous vehicles. Computer vision system have four main components:

- Real-time image acquisition
- Image processing
- Image segmentation
- Decision making system

The scope of this project focuses on the third component: image segmentation.

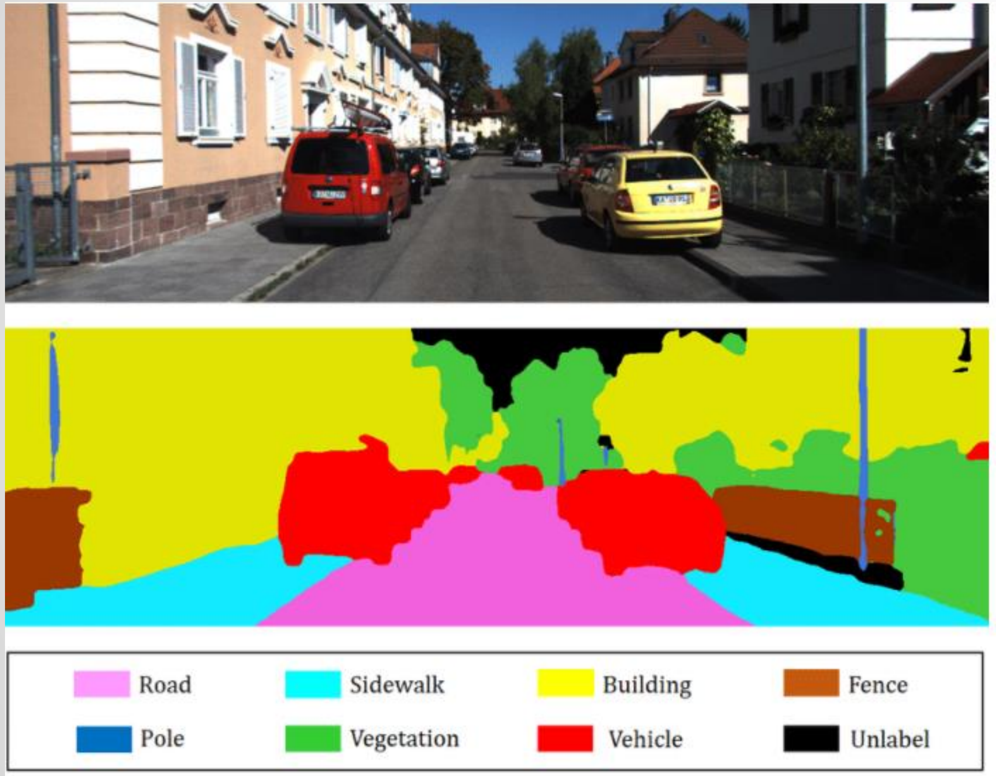
This component input/ output

Input : image

Output : a mask with a label for each object detected

Image segmentation

The mask provide a label and the contour of the detected object



Dataset from cityscape contain 1500 pictures of urban traffic from different European cities.

The label were divided into **8 main classes**:

- Void
- Flat
- Construction
- Object
- Nature
- Sky
- Human
- Vehicle

Metrics

For image segmentation we want to predict a label for each pixel.

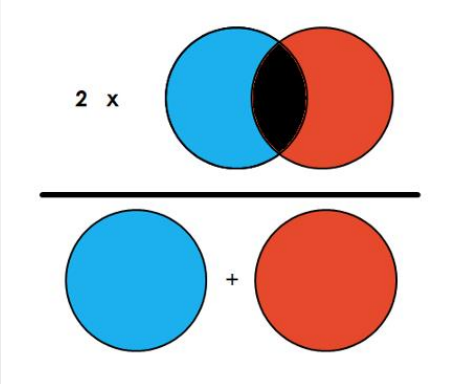
Pixel accuracy: number of pixel with the correct label predicted

Raises issue for classes with small amount of pixel like human the same it does with accuracy and imbalanced dataset with two label.

Solution = equivalent to F1score

Dice coefficient : The dice coefficient is 2 * the area of Overlap divided by the total number of pixels in both images:

$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$



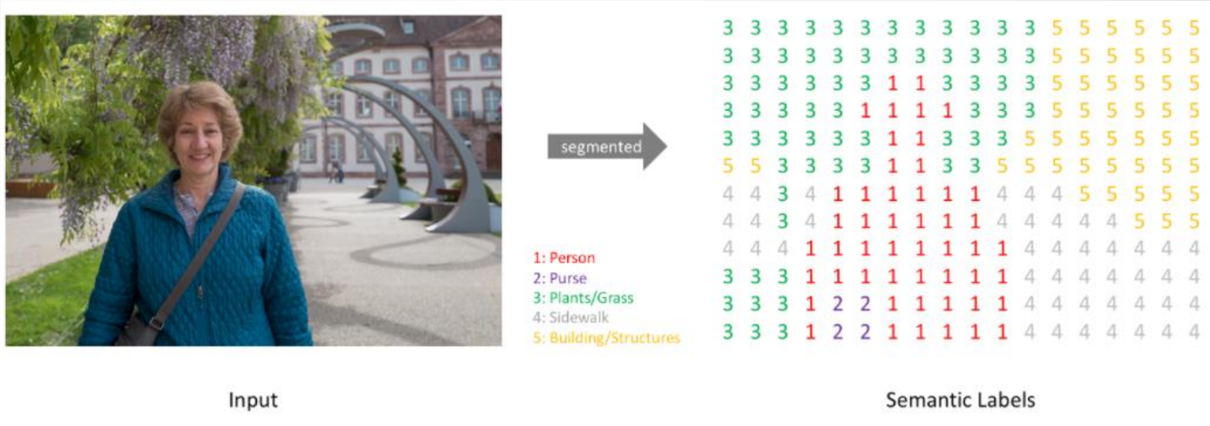
02

Architecture

Target

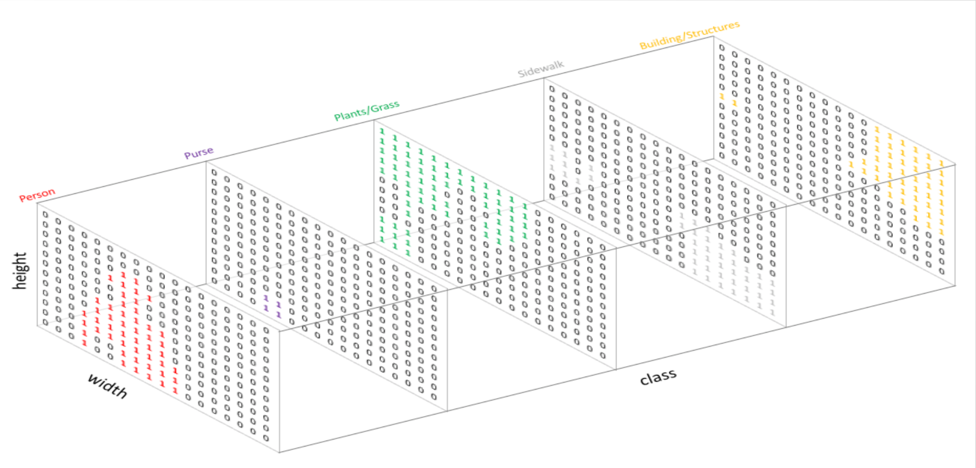
The model predict **a number per pixel** between 1 and n, n being the number of classes.

Each number corresponds to a **class**.



Equivalent of **one hot encoded**.

Instead of having a target of dimension **height*width*1** with value for each pixel ranging from 1 to n, we create a **target of dimension height*width*N**.

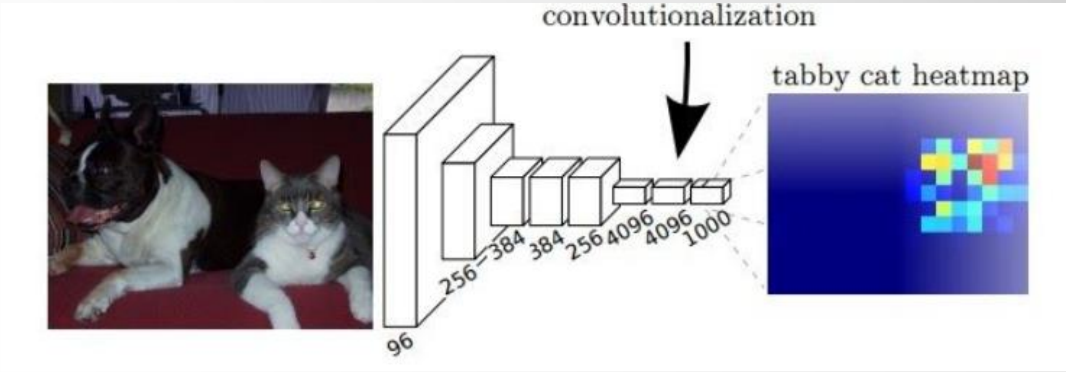


Model : down and up-sampling

Regular **Convolutional Neural Network** : **convolutional and pooling layers** + **fully connected layers**

Output → **Heatmap of the required class**.

→ First block : **down sampling**



Second block : **up-sampling**

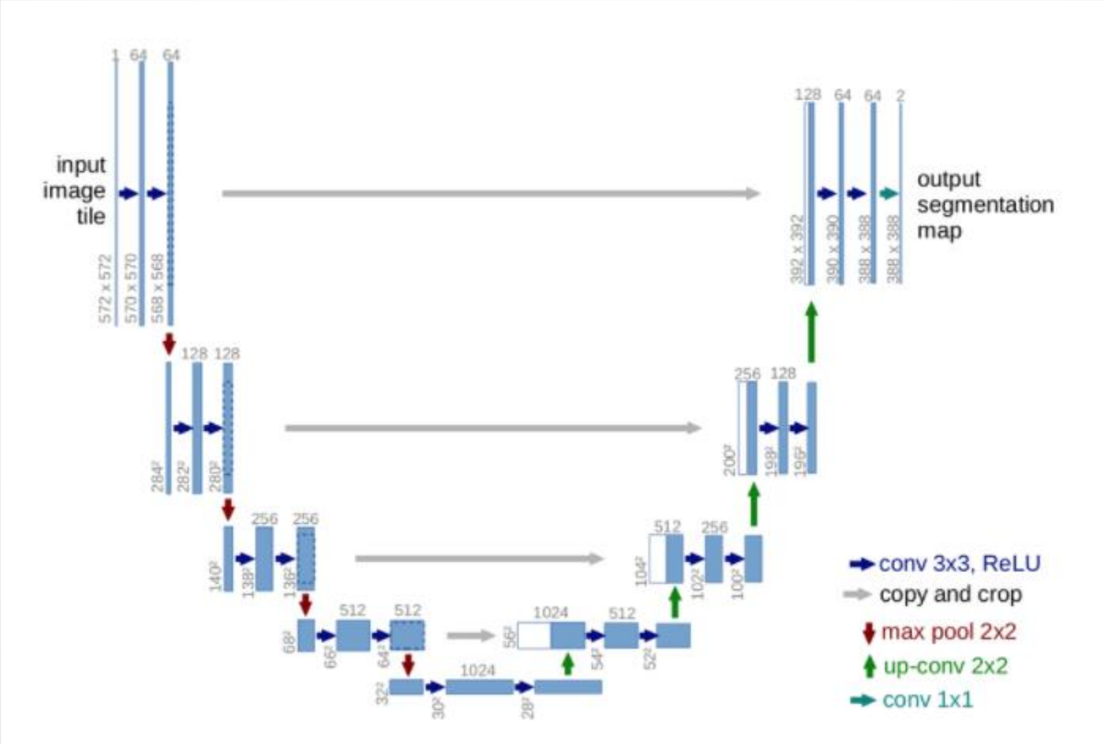
→ **interpolation**

Main **issue** with this structure :

Loss of information

Model : Unet

Decoder which up samples the feature map to input image size using **learned deconvolution layers**.



To solve the **information loss problem** U-net send information to every **up-sampling layer** in the decoder from the **corresponding down-sampling layer in the encoder**.

04

Training

Data augmentation

Augmentor package advantages : easy to implement and modification as a pair image and mask.

Modification:

- Rotation (radom rotation of the picture)
- Horizontal flip (reversing the entire rows and columns of an image pixels)
- random zoom
- Perspective skewing left, right, top, bottom and corner (transform the image so that it appears that you are looking at the image from a different angle)
- Elastic distortion (random distortions while maintaining the image's aspect ratio)

Augmentor main downsides :

- automatically save the augmented images and masks in the same output folder.

Number of augmented picture : 2000.

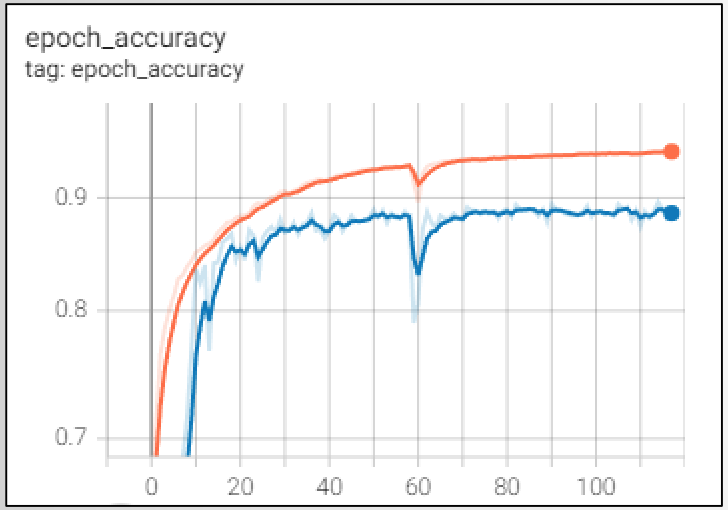
Baseline will be tested on **1500** pictures.

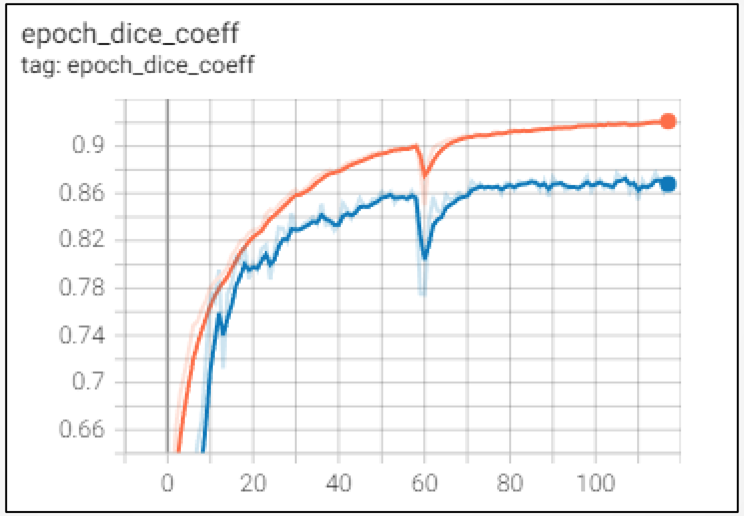
Test the **effect of data augmentation** on a set of **3500** pictures.

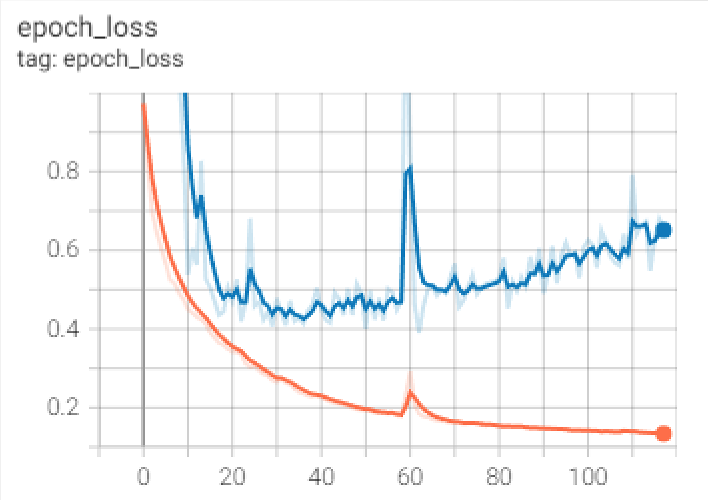
Baseline

Parameters : optimizer : Adam | learning rate : 0,001 | dataset : 1500 pictures

epoch : 150 | batch size : 32







Results : training time : 8h20 | parameters : 135,294,656 | Max val_dice : 0,865

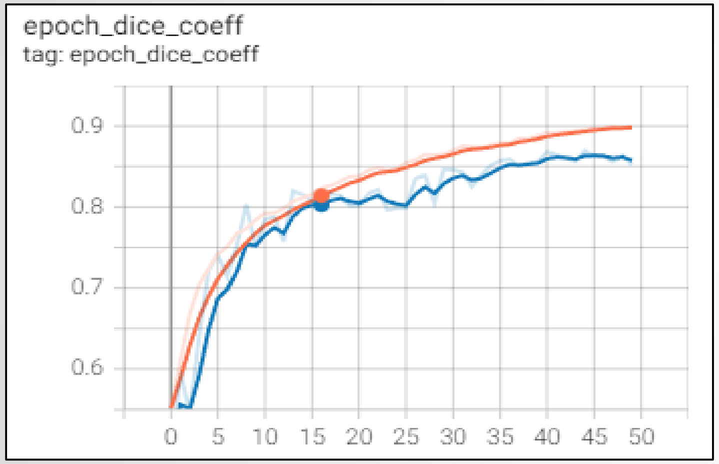
Remarks : validation dice coefficient converge to max value after 20 epoch

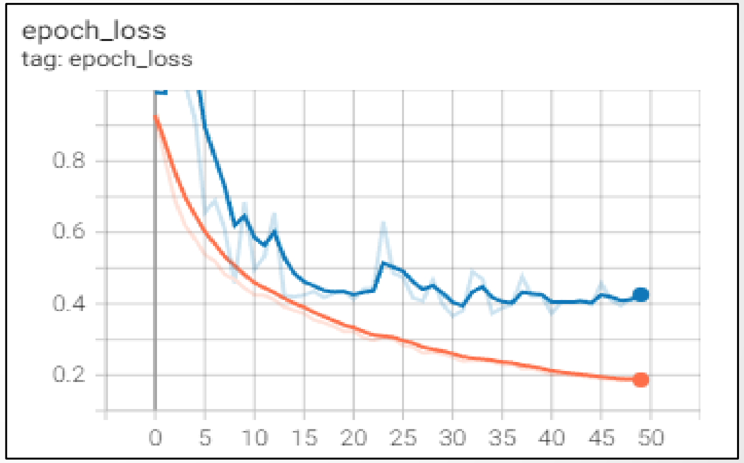
Loss divergence indicates overfitting after 35-40 epoch

Hyper-parameters tuning 1

Parameters : optimizer : Adam | learning rate : **0,001** | dataset : 1500 pictures

epoch : 50 | batch size : **16**





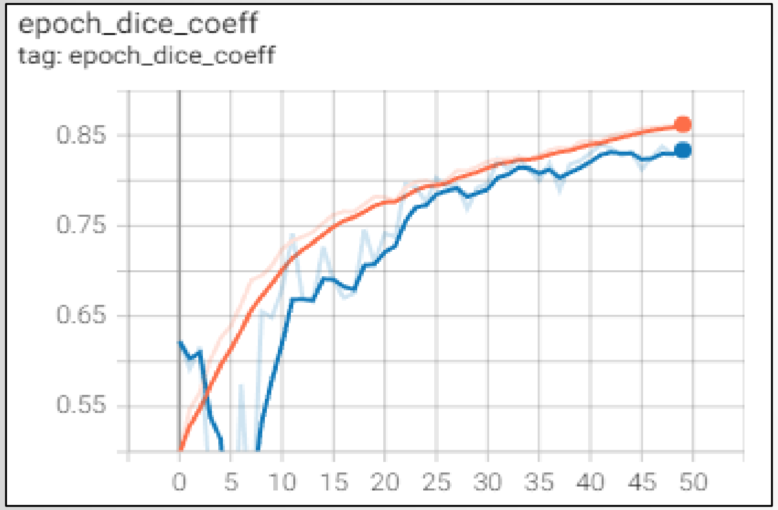
Results : training time : 3h45 | | Max val_dice : **0.856**

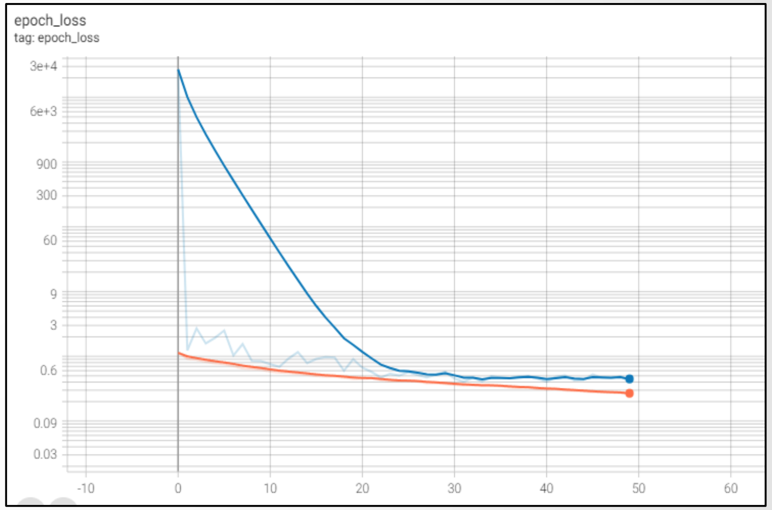
Remarks : almost same result as baseline

Hyper-parameters tuning 2

Parameters : optimizer : Adam | learning rate : **0,01** | dataset : 1500 pictures

epoch : 50 | batch size : **32**





Results : training time : 3h42 | | Max val_dice : **0.8335**

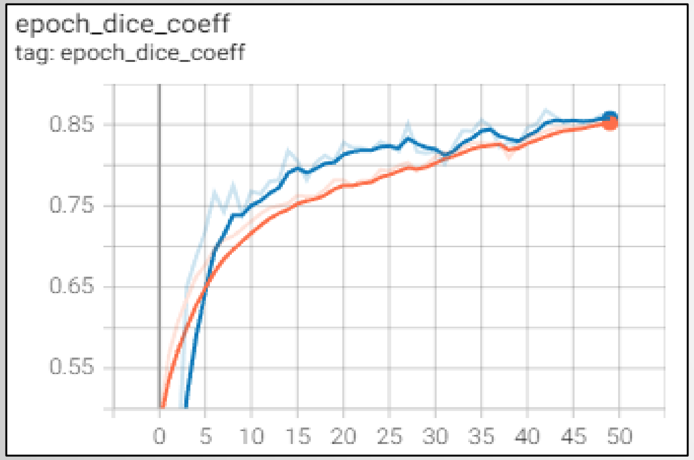
Remarks : validation dice coefficient doesn't reach max value ➔ increase in training time

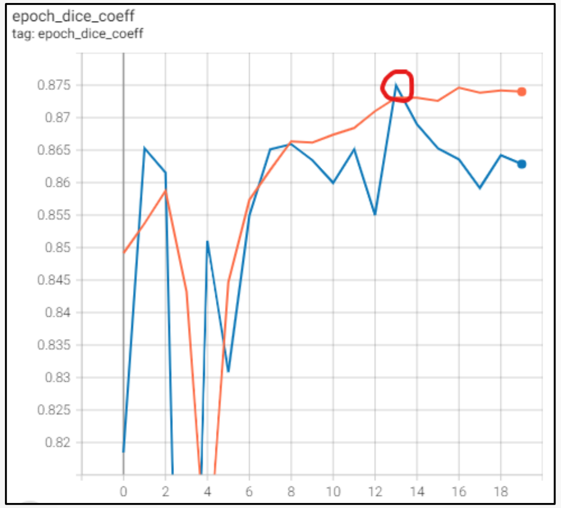
Loss divergence indicates possible overfitting

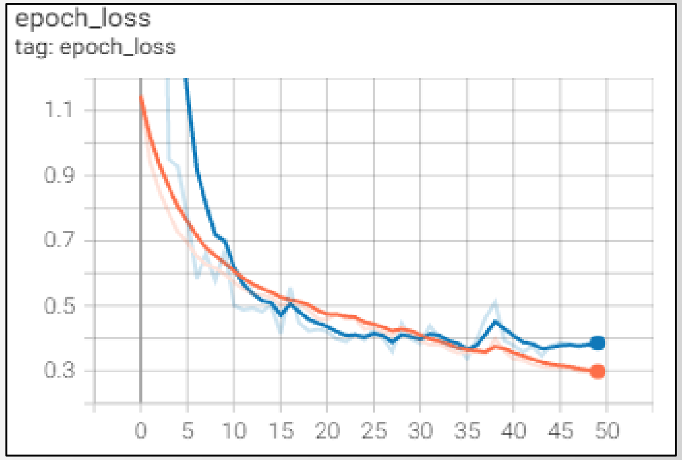
Data augmentation

Parameters : optimizer : Adam | learning rate : **0,001** | dataset : 3500 pictures

epoch : 50+20 | batch size : **32**







Results : training time : 10h | | Max val_dice : 0,88

Remarks : Impact of data augmentation ➔ **more consequent** than hyper-parameters tuning.

It is likely that training for more could **improve** performance.

Results

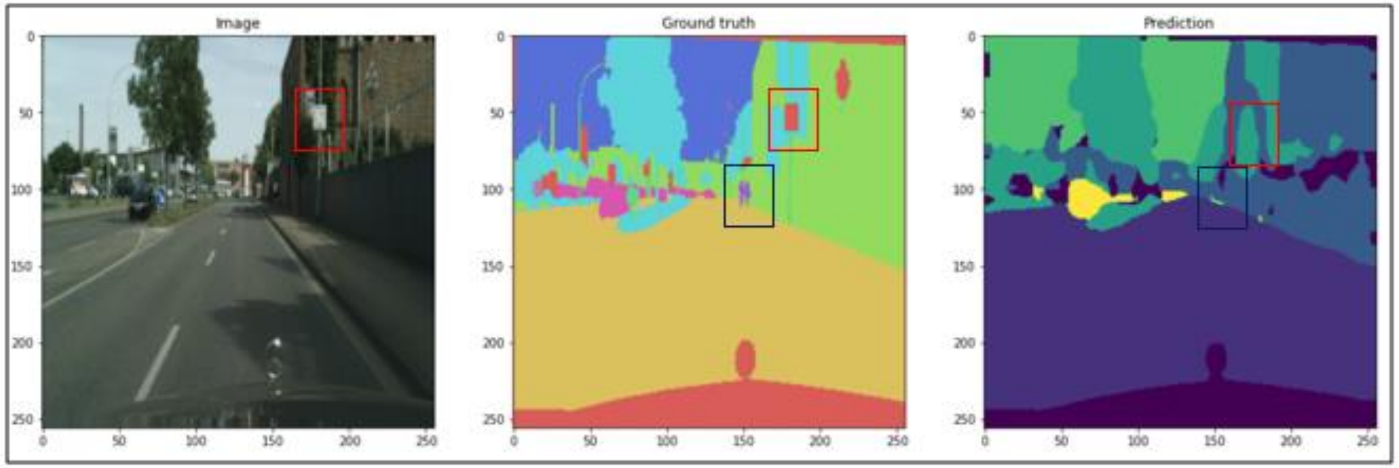
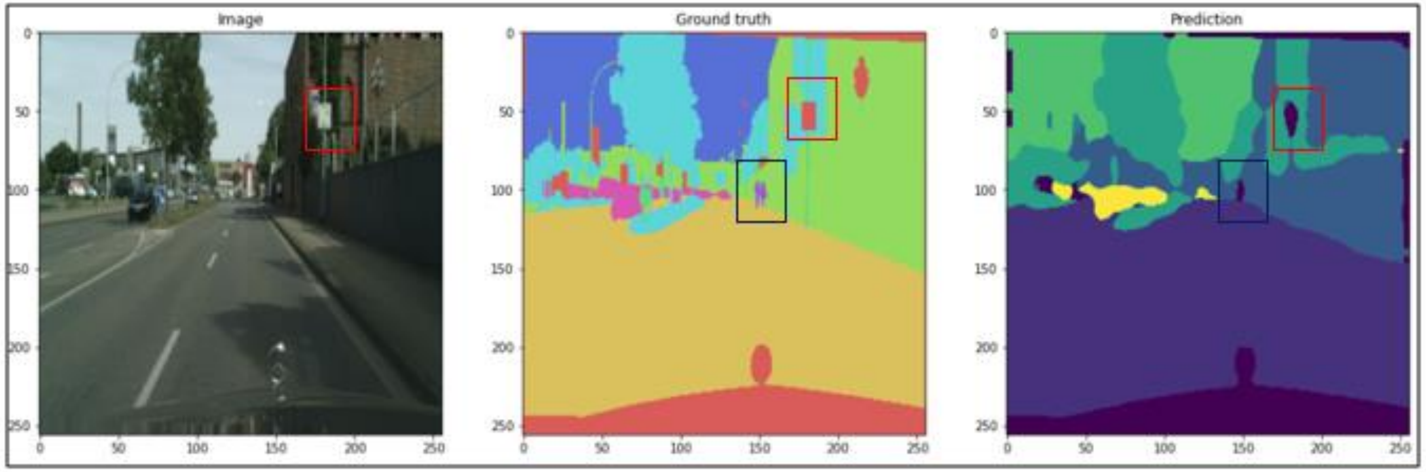


Figure 16 : prediction with baseline model.



05

Conclusion

Conclusion

Good performance

- Above 0,8 with baseline with “small” dataset
- Small impact of hyper parameters changes
- long training time (with basic GPU)
- Very heavy model

Data augmentation

- Significant impact
- Increase training time

Way ahead

- Increase training time and data augmentation
- Try data augmentation as an initial layer in the model
- Try other parameters

QUESTION ?

OpenClassRooms: Project 9

A x e l   F a v r e u l