TEK5040/9040 Autumn-2022 Compulsory Assignment No 1 Semantic Segmentation

Adrien Komaroff-Kourloff[1]

[1]Ecole des Mines de Saint-Etienne ISMIN / University of Oslo

September 2022

Contents

List of Figures

Introduction

This document is an assignement report based on the TEK5050 course at UiO. Reader is assumed to have read the assignement documents.
In that document, we perform semantic segmentation on the Kitti road data set. We perform binary segmentation between ROAD(1)(the pixel belong to the road) and NOROAD(0)(the pixel do not belong to the road). The set is made of 287 $256x256$ pictures.
We will be using a Simple CNN ($53K$ parameters) that achieved 0.858 accuracy and 0.273 loss on the validation set. That first CNN isn't designed to attain high performances but rather to experiments with CNN and gives reference of easily attainable performances.
We will then be using a simplified U-net architecture ($486K$ parameters) that achieved 0.920 accuracy and 0.171 loss on the validation set.

1 Global consideration

The two networks are trained and validated with :

- epochs : 12
- train batch size: 4
- validation batch size: 8
- train set size : 272
- validation set size : $287 - 272 = 15$

2 Simple CNN

You will find model architecture in outputSimple/modelMicture.png .

2.1 Results

Figure 1 and 2 presents the validation and training results of the Simple CNN.

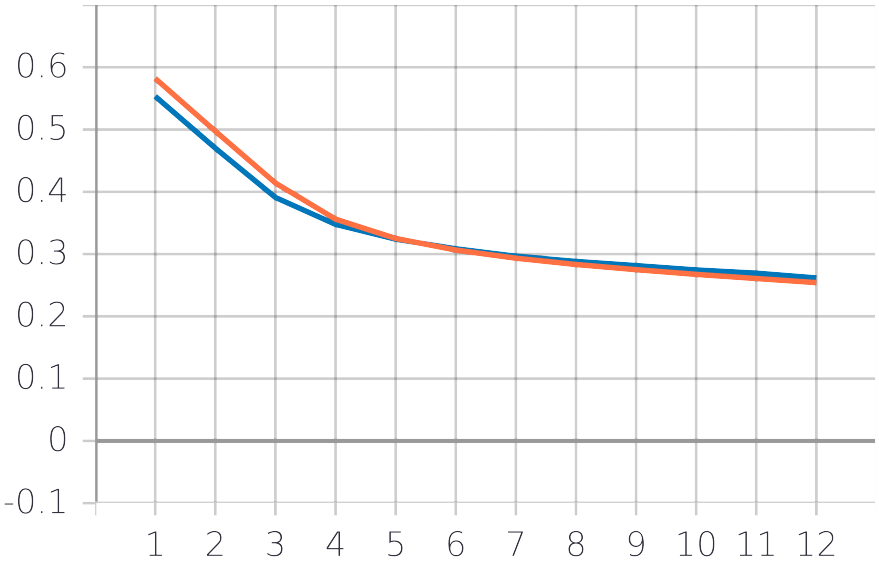

Figure 1: Loss of Simple CNN for train and validation

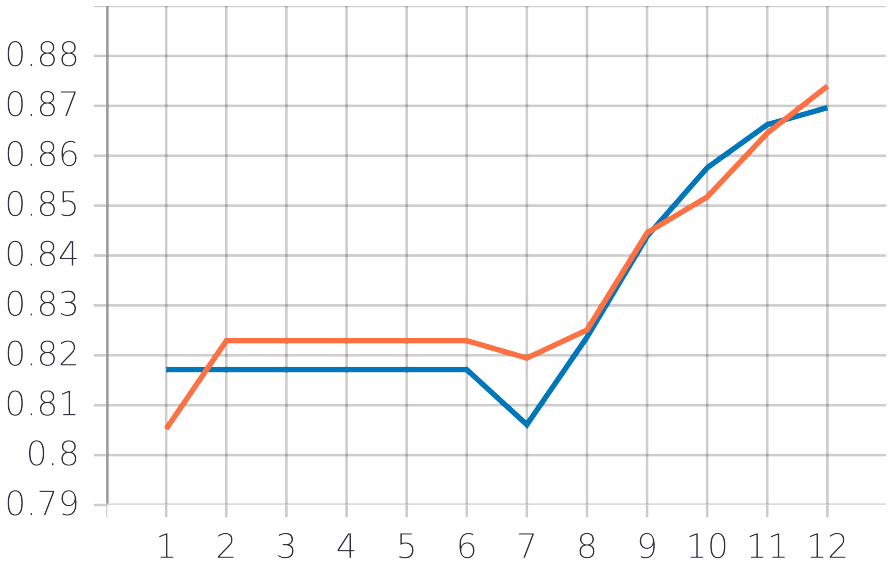

Figure 2: Accuracy of Simple CNN for train and validation

2.2 Questions

Q : **You may observe that the accuracy is quite high already after one epoch, what is the main reason for this?**

R : *The accuracy is the metric of the correct guesses between (ROAD/NOROAD) for each pixel. Hence a model giving always NOROAD would have a high accuracy considering the ground truth pictures are mainly made of NOROAD labels. We can see that the the output of the model is mainly NOROAD for the first epoch.*

Q : **The training loss should be decreasing from the start, but it might take some time before the accuracy increases after the first epoch, why do you think this is?**

R : *The improvement of the number of pixel correctly guessed "ROAD" is not immediatly visible in the accuracy because it is being hidden by the sheer number of correctly guessed "NOROAD". It is the result of the mathematical properties of accuracy. Contrary to that, the binary squared error function used to calculate the loss is the direct reflect of errors that are directly reduced with the increase of correctly guessed "ROAD".*

Q : **How many training steps are there per epoch?**

R : *batch size is 4, train set size is 272, hence the number of step is* $\frac{272}{4} = 68$.

Q : **How many training steps are there in total?**

R : *there is 12 epochs, hence the total number of steps is* $68 * 12 = 816$.

Q : **Can you think of any cases where you can get good accuracy result without the model having learned anything clever?**

R : *Giving always the same label class in the case the vast majority of the instance of the classes are corresponding to that label.*

Q : **Can you think of any other metrics that might shed some additional light on the performance of your model?**

R : *We could use False negative rate, False positive rate, and compare it with recall , precision, F-measure etc.*

3 Unet

You will find model architecture in outputUnet/modelPicture.png .

3.1 Results

Figure 3 and 4 presents the validation and training results of the Unet.

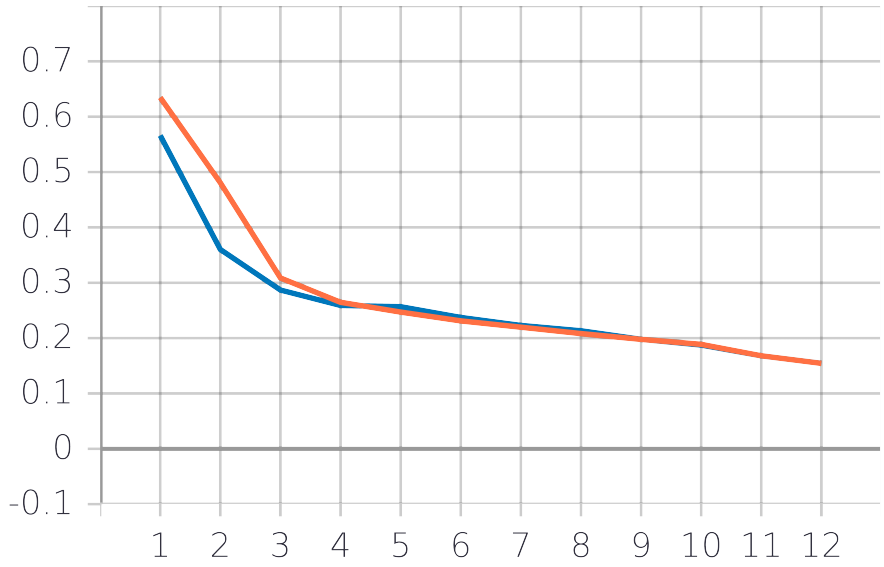

Figure 3: Loss of Unet for train and validation

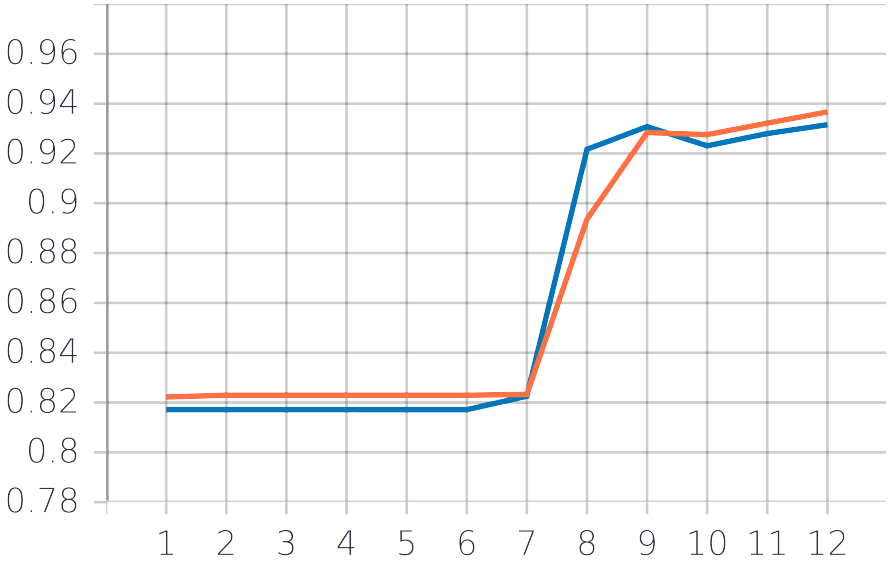

Figure 4: Accuracy of Unet for train and validation

3.2 Questions

Q : **Briefely describe transposed convolution also known as deconvolution. (Hint: It may be easier to consider 1D-case.)**

R : *In transposed convolution, each pixel multiplies the kernel (scalar-matrix multiplication) wich slides trough the output (wich is firstly initialized as zero) according strides and padding. The multiplied matrix is added with the spatial intersection of the kernel and the output. Contrary, a classic convolution uses the dotted product between the kernel and the spatial intersection of the kernel and the input space followed by the addition of all the numbers of the resulting dotted product in 1 pixel. We can see that the transposed convolution map each pixel of the input as a matrix, whereas classic convolution map sub-matrix of the input into pixels. That is why transposed convolution is an up-sampling operation whereas classic convolution is a sub-sampling operation.*

Q : **How many trainable parameters do your model have?**

R : *For each convolution layer and up conv layer, there is* $(ChannelNumber_{input} * FilterNumber * kernelSize^2) + FilterNumber$ *parameters, where the last addition account for the biases.*

Considering that the ouput number of channnels of a layer is the number of filter used for this layer, that the max pooling and up-conv layer respectivly divide and multiply (H,W) by two, that the concatenation is along the channel axis ,we can find with the formula above that the total number of parameter is 485,817.

Q : **Do you expect your model to behave differently under training and test modes (i.e. for a given input do you get the same output from the model in train and test modes)? State the reason for your answer.**

R : *We can't expect so since the weights are updated after the training.*

Q : **If your task was to perform segmentation with more than two classes (eg: four classes ROAD, BUILDINGS, CARS, OTHER), how would you change the activation function and number of output channels?**

R : *The number of channels of the last convolution layer would be the number of classes, and we would use softmax as an activation function. To respect the Unet architecture, we would need to multiply the size of all the output channel by* $\frac{2}{NumberOfClasses-1}$.

Q : **If your task is classification of a given image, how do you modify the model architecture and the loss function?**

R : *The last layer would be a dense layer instead of a convolution layer. The number of neurons of that dense layer would be the number of classes. We could use categorical-cross-entropy either with hot encoded vectors or with sparse vectors.*

Q : **Did you notice any improvements over the orignal model?**

R : *Yes, definitely.The last loss and accuracy of those models are presented in the Introduction : accuracy is better by 0.062 and loss is better by 0,102 on the validation set. Those improvement are especially striking when looking at the red colored pictures on Tensorboard (front page pictures).However, the Unet architecture efficiency can only be estimated by comparaison of a simple CNN with the same number of parameters.*