Homework Assignment 2

Lenart Rupnik, 63220742

1 Introduction

For our second homework, we worked on different CNN architectures and tested them on a few datasets. At first, we focused on classification problem and later on segmentation and colourization. All tasks were done using Pytorch library.

2 Experiments

For the first task, we wanted to implement ResNet-18 architecture. In terms of implementation, we used parameters presented in Table 1. To increase our accuracy over 90 % we first increased the number of epochs. Yet we didn't manage to get past 87 %. Then we tried experimenting with different learning rates but still didn't significantly improve our model. After we included learning rate regularization with step learning rate decay and tested different configurations, we managed to push our model to 93 % accuracy. In the end, we trained our model with 30 epochs. All configurations were based on Adam optimizer, since we figured we would only lower results with something like SGD.

Table 1: Task 1 parameters configurations.

Architecture	Optimizer	LR	Epochs	Regularization	Cls. Acc.
ResNet-18	Adam	0.0005	30	StepLR Decay	93%

For the second task, we used two architectures, FCN-32 and U-net. For the backbone of the Fully convolutional network, we used our implementation of Resnet-18. Since the task was to achieve segmentation, we had to add a segmentation head. With that, we removed the last fully connected layer from Resnet-18 and instead added a deconvolutional layer for upsampling the output with kernel of a size 32x32. This is a robust way to adapt the typical Resnet18 architecture to produce output of the initial image size. The results are shown in Table 2.

Next, we implemented U-net architecture based on the given article. This model consists of a series of convolutional layers that progressively reduce the spatial dimensionality of the input image. The next step is a so-called expanding path, which is a series of transposed convolutional layers that gradually upsample the feature maps to the original image resolution. The main difference between U-net and FCN are skip connections in U-net architecture. The skip connections link corresponding feature maps in the contracting and expanding paths that allow the network to preserve both high-level and low-level features in the final segmentation output. Results of U-net architecture can be seen in Table 1.

Table 2: Task 2 architecture comparison.

Architecture	FCN32	U-net
LR	0.0001	0.0001
Epochs	5	3
Regularization	StepLR	/
Mean IoU	0.43	0.528
Mean IoU Jaccard (macro)	0.477	0.582

Based on our tests, we can say that in this type of dataset, U-net performed better than FCN32. This is not only confirmed by mean IoU measure, but can be also seen by looking at a few examples of produced segmentation masks. In Fig. 1 we can see how FCN32 isn't able to segment smaller things like street lamps. Also, the background is blurrier and the shape of the segmented car isn't correct. The most problematic thing in my opinion is not recognizing the traffic lights, which might be crucial if we used it in some kind of self-driving applications. On the other hand, we get quite precise results from U-net. For example, in Fig. 2 the car shapes are more precise and with more details. We manage to detect both traffic lights and street lamps. There are still some failures, but overall it gives out much better results than FCN32.

During training, we found out that the results varied between different trainings with the same parameters. Sometimes even for +/- 5 % mean IoU. This is probably due to low batch size and a few epochs. We would more or less converge to the same results if we could use more epochs and bigger batch size, but we were constrained with given resources for learning. Since training differed between 15-25 minutes, we couldn't afford to run longer periods because of Google Colab restrictions. Also, we needed Google GPU resources for other tasks as well.

For the last task we used our already implemented U-net architecture for colourization problem where we tried to colourize gray images. After training the model, we removed skip connections and learned another model. Results of both models are seen in Fig. 3 and Fig. 4. The difference in result is mainly that when omitting skip connections, our output is blurrier and less precise. I would assume that this is because

without skip connection, we lose some information of high level features. In terms of colours, the results are quite impressive. We got even better results than the one shown in figures but I wanted to show comparison of these two where we can clearly see how model underperforms in untypical situations. So in both images the grass is yellow, which confuses the model to paint it green. This could be fixed, mainly by providing bigger dataset.

3 Image data

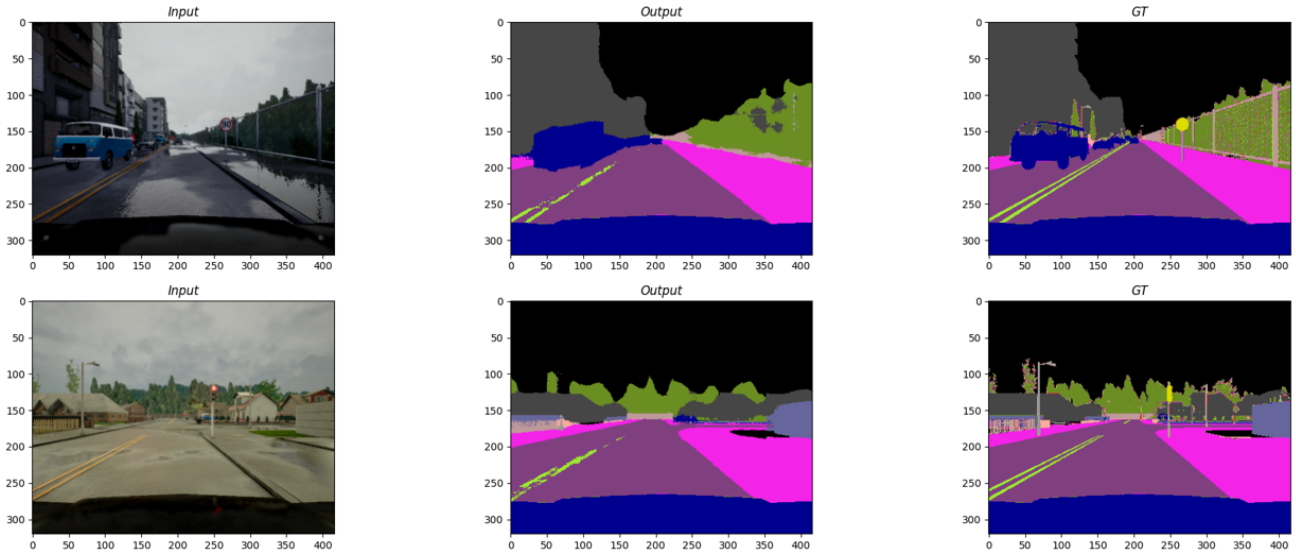

Figure 1: Examples of FCN32 results

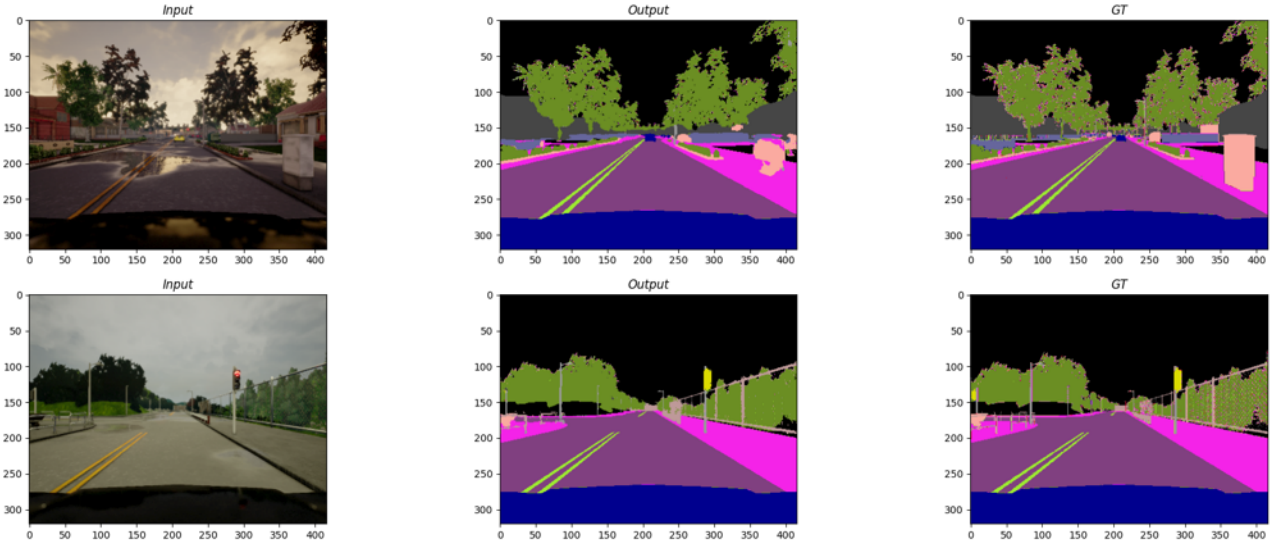

Figure 2: Examples of U-net results

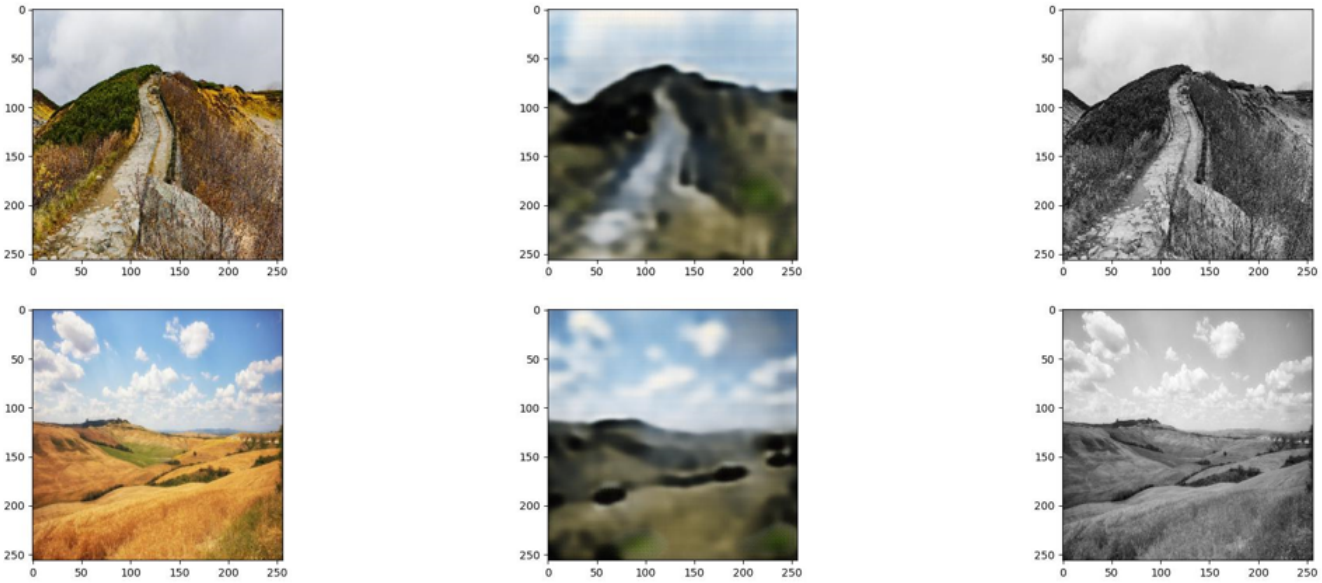

Figure 3: Colourization without skip connections.

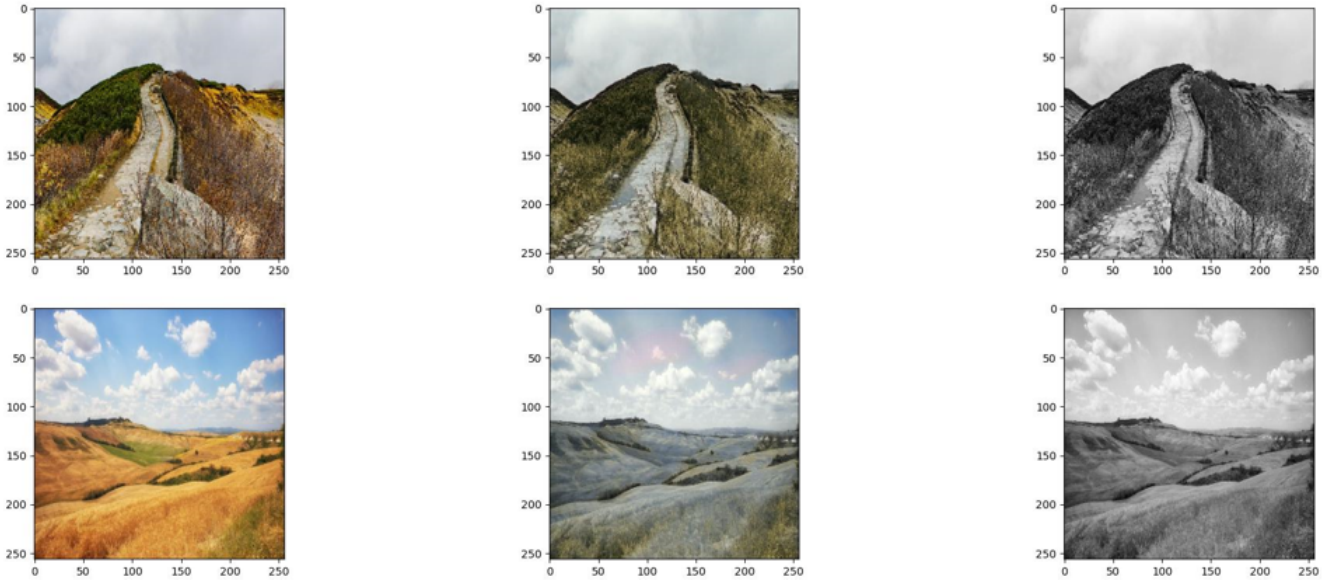

Figure 4: Colourization with skip connections.