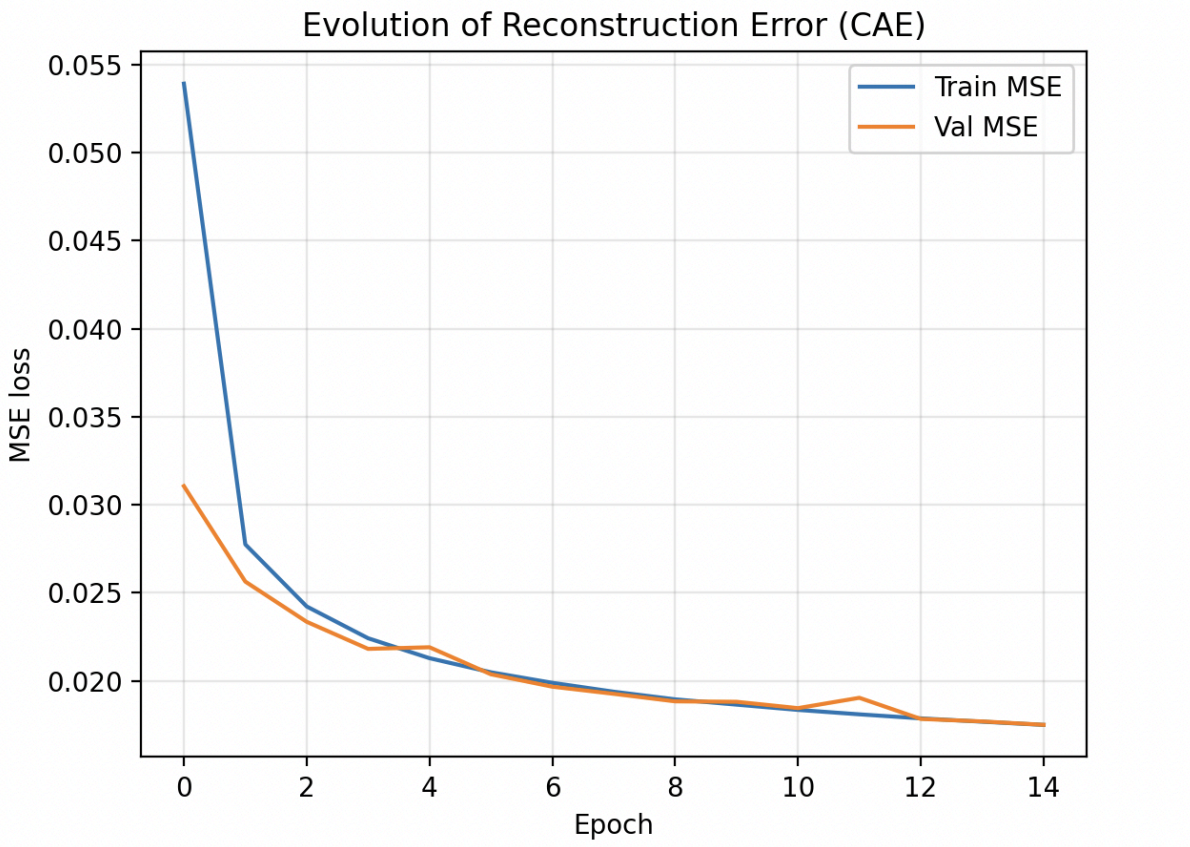
Report Autoencoders - Popa Stefan Andrei & Levinschi Eduard

Exercise 1

First step on doing exercise 1 was to split the dataset which was already provided by tensorflow library in train, validation and test. This was quite challenging because the load of the dataset split in 2 different sets. First of all, it was normalized dividing the values (/127.5 -1) because a pixel value is between 0 and 255 and afterwards we need values between -1 and 1 (the requested activation function was ReLU and values should be between -1 and 1

The second part of exercise 1 was to implement the autoencoder specified on the architectures from the assignment. The first step on doing this is building the CAE model using already done functions from Keras library.



Above, it represents the plot with the evolution of the MSE during the 15 epochs(from 0 to 14), based on validation and training data. It can be observed that MSE decreased from almost 0.055 to less than 0.020 in 15 epochs. The best model gives 0.017909 (MSE) for test data which will be shown in a table of comparison after defining the rest architectures.

Exercise 2

Size of the latent space representation has been computed using the following formula given already in the lab.

$$(\frac{W-K+2P}{S}+1)^2 \cdot C$$

From there, all that I had to do was to substitute values for each parameter(of course respecting all the details given above about my architecture) and use the function called conv2d_output_size. In our case , the value was 1024(W,K,P,S,C already explained in the assignment).

The other architectures computed results:

Nr	Classic	4-6-8	16-24-32	16-24-32 filter: 5x5	32-48-64-single	32-48-64-double
MSE Test	0.017909	0.026883	0.047514	0.003731	0.007290	0.006165
SoLSP	1024	512	2048	1152	4096	4096

MSE = mean squared error
SoLSP = size of latent space representation

As can be seen above, architectures with more representational power (more filters, deeper blocks) tend to achieve better reconstructions, we observe this looking at the MSE computed in the end. I want to mention the fact that 32-48-64-double filter architecture took twice at long rather than 32-48-64 single, and where the differences between their MSE are small , it should be a great idea to choose 32-48-64 as the architecture with best performance(using it for colorisation).

In terms of latent space size and MSE , there is no direct correlation between them, it can be noticed just the decrease which occurred with a higher LSP.

Exercise 3

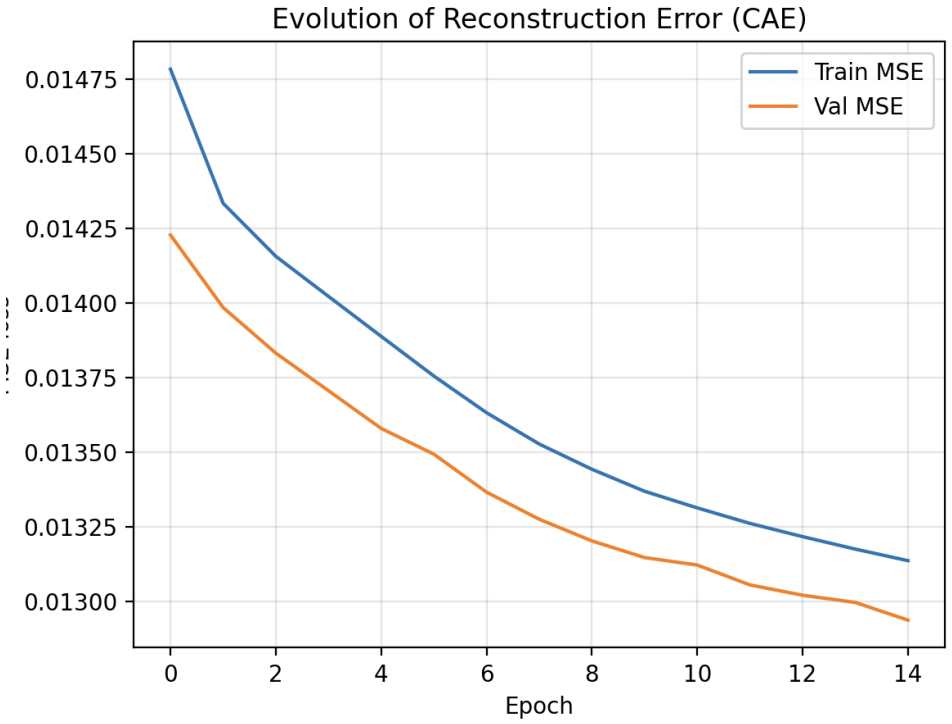
The chosen architecture was 32-48-64 single which gave us the best results in terms of MSE and time computation. First of all, an RGB-gray afterwards trying to turn again into the RGB version was hardcoded. Moreover, we tried from Y to UV , which was the hint from the lab splitting into luminance and chrominance. The most important part in the colorisation exercise was to figure out what is the input and what is the output, thus:
RGB:
from RGB -> GRAY -> RGB(1 channel as input, 3 channels as output)

YUV:
from RGB -> Y -> UV(1channel as input, 2 channels as output)

No	RGB(32-48-64 architecture)	YUV(32-48-64 architecture)
MSE	0.006633	0.003041

A great idea for improvement would be trying the 32-48-64 architecture in these approaches to check the results because it helps extract richer features before downsampling. It was already observed that using a large kernel, results were better and MSE value was the lowest.

I tried a lower learning rate, from 10^(-3) to 10^(-5) and the MSE results was not expected, on 32-48-64 single filtering architecture and YUV one, we had 0.013091 MSE and a computation time of 5 minutes and 14 seconds.



It could be proposed another improvement which is based on the number of epochs, we focused only on 15 epochs, it gave certain results that can be improved with the increase of n.o. epochs. We had weaker results using a lower learning rate, but most likely in another form (ex: 50 epochs and 10^(-4)) can improve the MSE results at the end of the test process.