Using a Deep Learning Framework: Convolutional Autoencoders

Advanced Concepts of Machine Learning

I6278383 - Pierre Onghena
I6263878 - Simon Hilt

Exercise 1

The figure 1 shows the evolution of the error over 10 epochs. First it seems counter-intuitive that the training error is higher as the error of the validation set. This can be explained with the averaging of the error over all batches of one epoch. Since the model is usually increasing its performance too the end of an epoch during training, the error might be less on the validation set. From epoch one onwards the train and validation error decreases implying a working model.

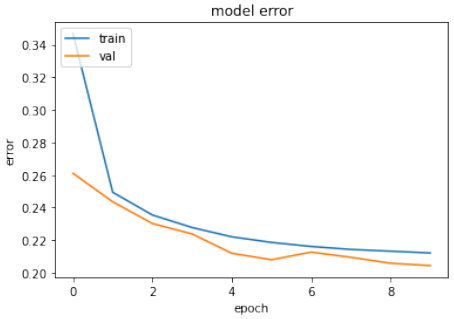

Figure 1: Evolution of the error with epochs

The error of the test rate is around 20.48% after training the model for 10 epochs. This matches with the error of the validation and training set. A similar graph exists while visualizing the validation loss over the 10 epochs (see figure 2). Also here the training loss is always higher (except for epoch 9 and 10) as the validation loss. For the later epochs over-fitting is taking place.

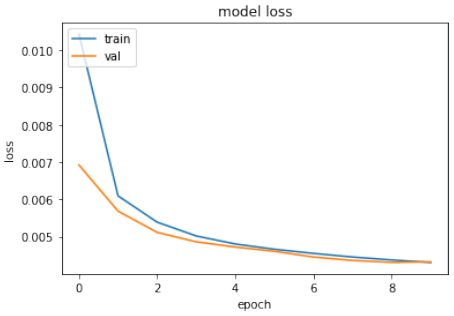

Figure 2: Evolution of the validation loss with epochs

During training, the 'accuracy' metric was added as an additional metric in Keras. As previously illustrated in figure 2, the loss went down and accuracy would go up when using this metric. However, Keras documentation describes the working of the accuracy metric as follows: *"Calculates how often predictions equal labels"*. Therefore, as this assignment involves image reconstruction

and not image classification, the validation loss is a more appropriate metric to evaluate and enhance the model. Moreover, the test error could be defined as the loss when assessing the test dataset.

In figure 3 examples of the test set are displayed in the first row and their corresponding output of the convolutional autoencoder are displayed in the second row. They are a blurry version of their original image.

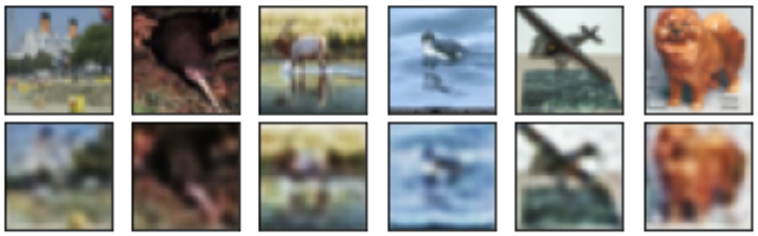

Figure 3: Example results from test set [1st row original, 2nd row results]

Exercise 2

Part 1

By first incorporating the output shape of previous layers with equations 1 and 2 with maxpooling, the size of the latent space can be calculated in equation 3.

$$\frac{32-3+2\cdot 1}{1}+1=32 \rightarrow Maxpool: \frac{32-2}{2}+1=16 \tag{1}$$

$$\frac{16-3+2\cdot 1}{1}+1=16 \rightarrow Maxpool: \frac{16-2}{2}+1=8 \tag{2}$$

$$\left(\frac{8-3+2\cdot 1}{1}+1\right)^2 \cdot 16 = 1.024 \tag{3}$$

Part 2

Figure 4 shows the influence of the different parameters in regard to the resulting validation loss. It becomes obvious, that the kernel size seems to have the largest impact on the validation loss of the model in that sense, that preferably larger kernel sizes generate lower validation loss.

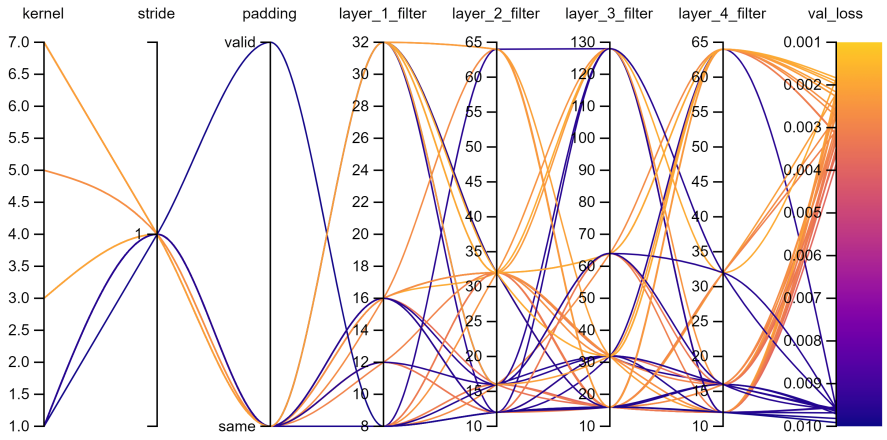

Figure 4: Parameter influence on the validation loss

The best resulting model, based on the validation loss (validation loss of 0.001836) has the following properties:

- kernel: 7
- stride: 1
- padding: same

- layer 1 filter: 32
- layer 2 fitler: 32
- layer 3 fitler: 32
- layer 4 fitler: 64

However, the layers seem to have an impact to certain extent with respect to the error rate. The layer 2 channel seems to have a larger impact on the resulting error rate than the other layers, but compared to the kernel size it seems less important in the outcome of the validation loss. This becomes also obvious in figure 5.

When the size of the latent space representation (layer 3) is lower, the correlation would induce a lower validation loss in comparison with the other layers. Other good architectures demonstrate this, a favorable dimension for the latent space would be defined at 32 and not higher to retain a low validation loss.

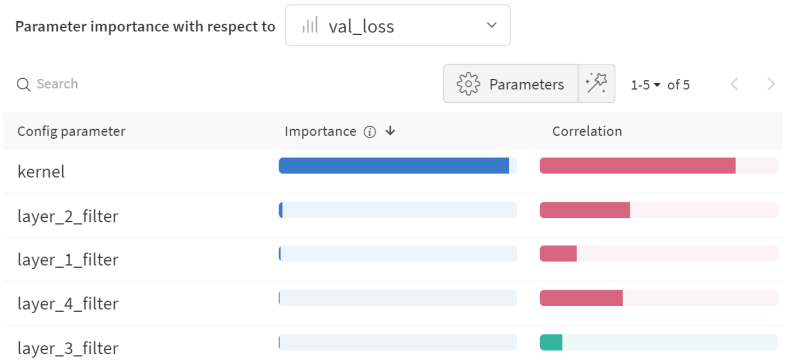

Figure 5: Correlation and importance of validation loss on the tested parameters

Exercise 3

Part 1

Taking the best model structure from exercise 2, based on the validation loss, the new evolution of the grey-scale images can be seen in figure 6.

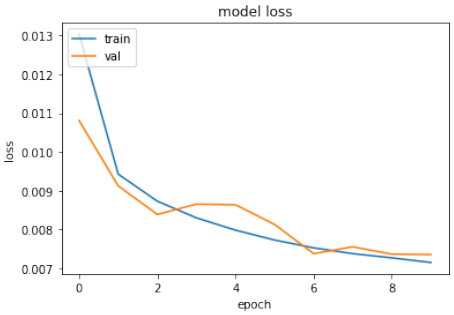

Figure 6: Evolution of the training and validation loss with epochs

Example results of the predicted color images can be viewed in figure 7. These results are based on the prediction of the test set.

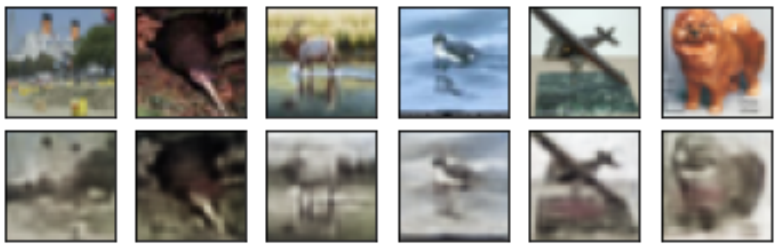

Figure 7: Example results from test set [1st row original, 2nd row results]

Part 2

For this part, which concerns the colorization of the grey-scale image, the color-space YIQ can be used. This relieves the autoencoder to having to predict also the content of the image, since the greyscale image and the two color space representations will be matched again at the end. So the autoencoder can fully concentrate on the color. The Y or gray-scale component represents the luminance or sharpness of the image, the AE has to learn the two remaining I and Q channels for colorization. This reduces the size of the network and stimulates a faster convergence.

One improvement that can be done is using the tanh cost function for the last layer instead of sigmoid. They are both very similar however the tanh func-

tion can also output negative values for y. Since the colorspace of Q can also be negative, it is leading to more representative results.

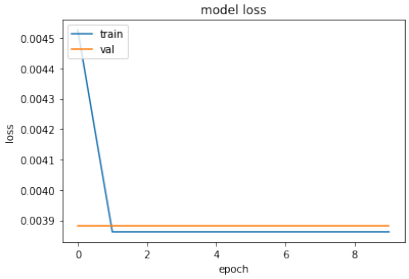

Figure 8: Loss of sigmoid

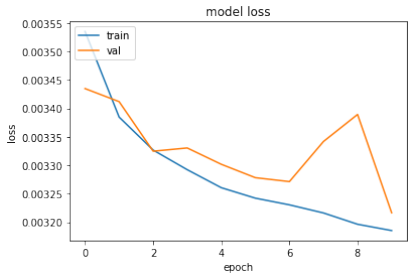

Figure 9: Loss of tanh

Further techniques to decrease overfitting can be implemented in the model. In figure 9 one can see that the training loss is lower than the validation loss which is an indicator for overfitting. In order to decrease the impact of retraining with the same examples dropout layers can be included. More training data would also increase the performance of the model. If one would have more training data, the model complexity could also be increased in increasing the number of layers such that more features can be detected and the coloring is more accurate.

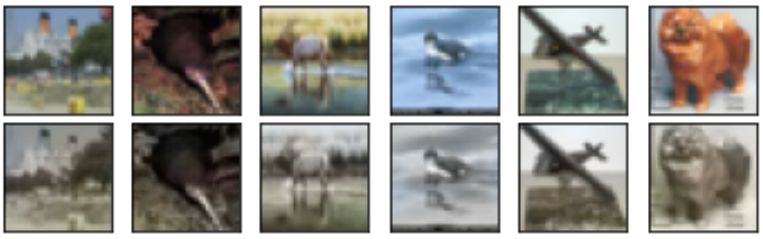

Figure 10: Example results from test set [1st row original, 2nd row results]