Advanced Concepts in Machine Learning
Assignment 2

Eliott Simon, Kevin Müller

November 2021

1 Task 1 - Implement the given architecture

For the assignment we started by creating a colab-file so that we could run the trainings on colab. We started by just splitting the data only into test and training sets, because the test_train_split for the data did for some reason not work how intended and planned. So after the Q&A we saw how to properly split the data and borrowed the code you used to split it for our model.

1.1 Binary Cross-Entropy Loss Function

The evolution of errors of the given architecture is displayed in Figure 1. We can see that already after the first test the training loss stops decreasing significally and test and training error only decrease in narrow margins for more epochs.

With that architecture the autoencoder is able to reconstruct the images given as input, but only in a very blurry way as can be seen in Figure 4. The architecture works, but seems not to be a good architecture, because the error is high and when decoding the the images they are much more blurry as the input.

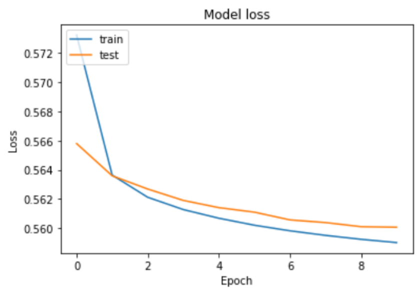

Figure 1: Loss Evolution

1.2 Mean-Squared-Error

Using the mean squared-error, the loss and accuracy over time converge in such a way:

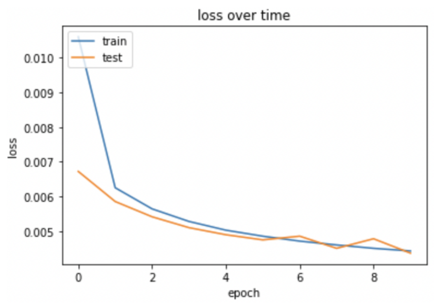

Figure 2: Loss Evolution

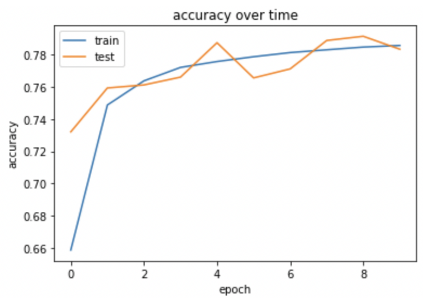

Figure 3: Accuracy Evolution

The binary-cross-entropy loss function is usually used for classification tasks, whereas the mean-squared-error is used for regression tasks. For this particular task, it is better to use the MSE loss function.

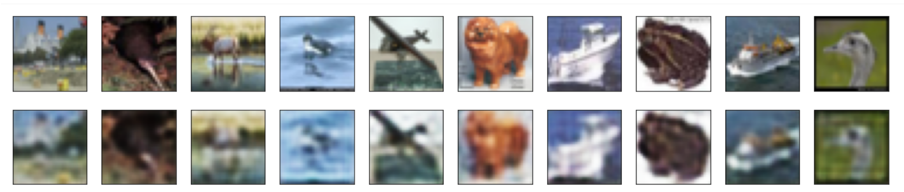

Figure 4: Input and predictions of the given autoencoder structure

2 Task 2 - Improve the given architecture

2.1 Latent space

To compute the size of the latent space representation of the given network, we use the given formula:

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

which in our case translates to:

- W = 8x8 = 8. Because we're using max pooling 2x2 twice and therefore the original input gets halfed twice

- K = 3x3 = 3. Kernel size is still 3
- P = 1. It is "same", therefore 1 for convolutional layers
- S = 1. Stride is always 1
- C = 16. Number of channels

$$\left(\frac{8-3+2*1}{1}+1\right)^2 \cdot 16 = 1024 \tag{2}$$

2.2 Experiments

To start the experiments with the given network, we thought about where obvious flaws in the given network could be that lead to bad performance. Using a different loss function just came to our mind, when we finished all other experiments and no real improvements could be made. So all the following experiments use binary-cross-entropy instead of MSE.

We started with the same network structure, but changing the max pooling method to the average pooling one. With max pooling the maximum value pixel of a batch is chosen, while the average one computes the average out of all values in a batch. With the average pooling method we thought we predicted pixels can get closer to the input data.

To reduce computation complexity and variance, we can use Pooling to compress the input data of the images we feed into the network. In max pooling, the maximum value of a pool is chosen as a feature, while average pooling computes the average over all values in a pool. So the values of a pool get smoother when taking the average in comparison to max pooling, where the most important feature is extracted.

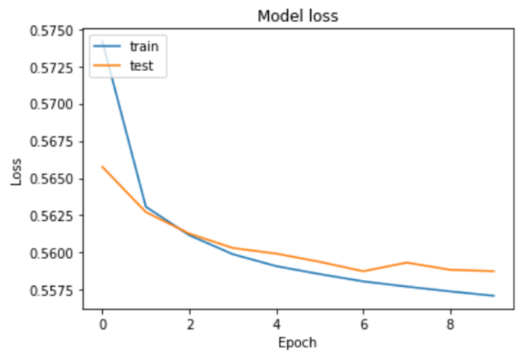

Figure 5: Evolution of errors of the given autoencoder architecture & AveragePooling2D

But as we can see in Figure 5 only changing the max pooling method to the average pooling method did not directly influence the results. After some research on the pooling method, we found out that "for low-level image restoration problems, we use neither pooling nor unpooling in the network as usually pooling discards useful image details that are essential for these tasks"(Mao et al. [2016]). So using the pooling method might be a reason for the output being more blurry than the input data. Where the pooling method has its strengths seem to be classification tasks for images, but not in image restoration.

So the next idea was to find out what happens if we don't do pooling at all and therefore also not do upsampling, as the excercise set out. As we can see in Figure 6 by not using the pooling and upsampling method, the error got better than trying different filter sizes and different amounts of layers using the pooling method. The improvement to 0.548 is only narrow in comparison, but by using the old architecture without pooling, the error improved. So pooling seems not to have an positive impact in our task of image reconstruction. When looking at the predictions of the autoencoder without pooling and upsampling, we can clearly see in figure 7 that in comparison to figure 4 the predicted output is not as blurry anymore.

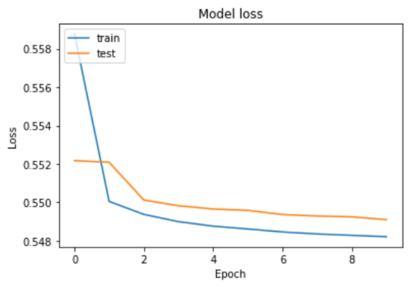

Figure 6: Evolution of errors of the given autoencoder architecture without pooling and upsampling

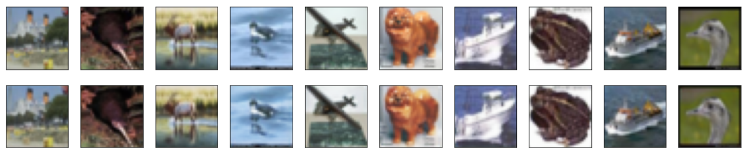

Figure 7: Predictions of the given autoencoder architecture without pooling and upsampling

Different experiments with different filter and layer sizes also did not improve the results when using binary-cross-entropy.

3 Task 3 - Colorization

In order to investigate chrominance of the image, we converted the RGB image into the CiLab domain. This allows to compute the luminescence (L) of the image, based on its chrominance (a,b).
When training the model, the luminescence of the image is represented as the X-values, whereas the Y-values represent the chrominance of the image.

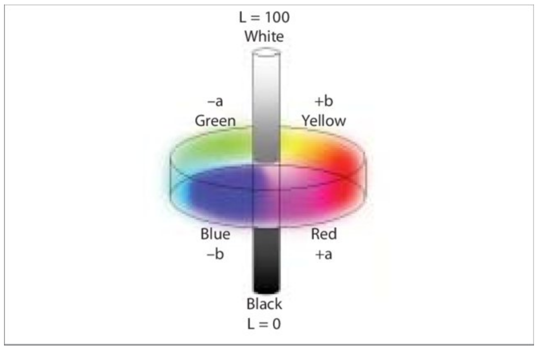

Figure 8: CiLab colors (Knösel and Attin [2008])

Using this, the model gives the following output:

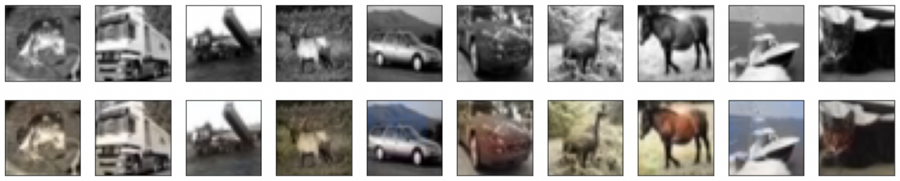

Figure 9: Grayscale (top) vs. reconstructed image (bottom)

References

Michael Knösel and Rengin Attin. A randomized cie l*a*b* evaluation of external bleaching therapy effects on fluorotic enamel stains, 2008.

Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using convolutional auto-encoders with symmetric skip connections, 2016.