Maastricht University

Advanced concept in machine learning
Assignment 2

Authors
Luca Brugaletta
Glauco Lorenzut

Supervisor
Kurt Driessens

November 10, 2020

1 Introduction

The first part of this assignment was the implementation of an Autoencoder to reconstruct the input image from its encoding. The second part is the implementation of another Autoencoder who can reconstruct a coloured image from a grey-scale one.

2 Implementation

In this section are presented the choices made for our implementation. The program is realized in Python 3.7 with the using of libraries Tensorflow, Keras and OpenCV.

2.1 Dataset

Our models were tested and evaluated with the dataset CIFAR-10, which includes sixty thousand coloured 32x32 images divided into the training set (50000 images) and testing set (10000 images) with their respective classes. The subdivision in classes was not useful in our project, so we took into account only the images. We preprocessed the dataset merging training and test sets. Subsequently, we split them again in 80% for training (48000), 10% for the evaluation (6000) and the remaining 10%(6000) for the test.

We also created another dataset with the same images and subdivisions but encoded in YUV, so we could obtain the luminance Y and chrominance UV for the second task. Both datasets were normalized and saved with the pickle library to avoid the preprocessing time (around 22-25 seconds) at every run.

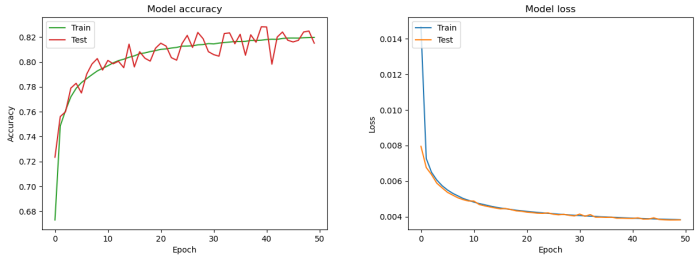
2.2 Standard auteconder network

We implemented the given Autoencoder and we ran it for 50 epochs, using a batch size of 50 and mean_squared_error as loss function [Figure 1]. This brought to an accuracy of 0.8209 and a loss of 0.0039 on testing set. The training took almost 202s on Google Colab.

2.2.1 Size of the latent space representation

The size of the latent space representation using this network was 1024. The result was obtained with the following parameters W=8, K=3, P=1,S=1, C=16.

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



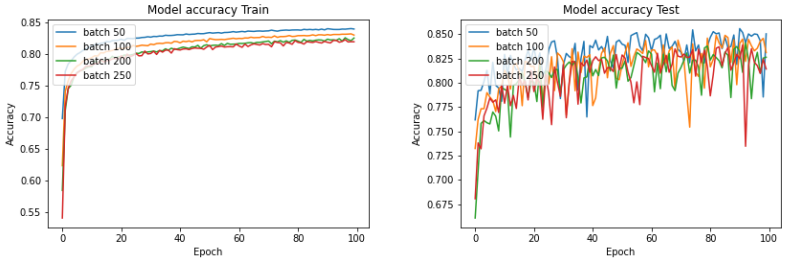
(a) (b)

Figure 1: Standard autoencoder

3 Experiments

3.1 Batch size

In our experiments we noticed that our custom Autoencoder was able to reach a lower loss value and an higher accuracy using a smaller batch size. While increasing the batch size gave more fluctuating results, especially a bigger loss. The tested bach sizes were 50, 100, 200 and 250. The best result was obtained using a size of 50 batches, with a loss of 0.0018 and an accuracy of 0.8510 [Figure 2] [Figure 3].

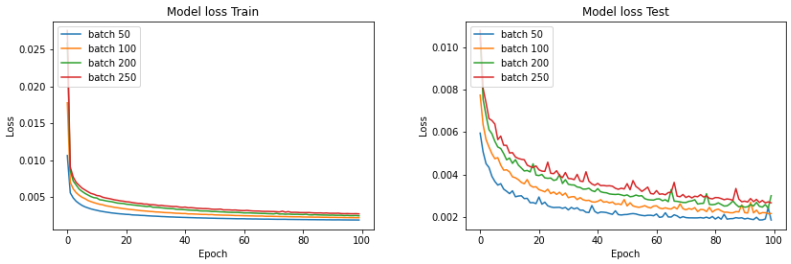


(a) Accuracy train (b) Accuracy test

Figure 2: Accuracy with differents batch size

3.2 Number of channels

Tests on channels returned better results when the channels number was increased. In every Conv2D layer we doubled it. Starting from 8 channels in the first layer and up to 64 channels on the last convolutional layer in encoding phase. During decoding we just halfed it to 32 in the only convolutional layer before the output.



(a) Loss on train (b) Loss on test

Figure 3: Loss with different batch size

3.3 Number of layers

Trying differents number of layers brought incoherent results. First of all, increasing the number of layers did not give any type of improvements, there was a major loss when compared to the structure given in the assignment. Reducing the number of layers was helfpul only when removing one MaxPooling2D and one UpSampling2D layer. In other cases the results were similar to the initial autoencoders structure.

3.4 Padding

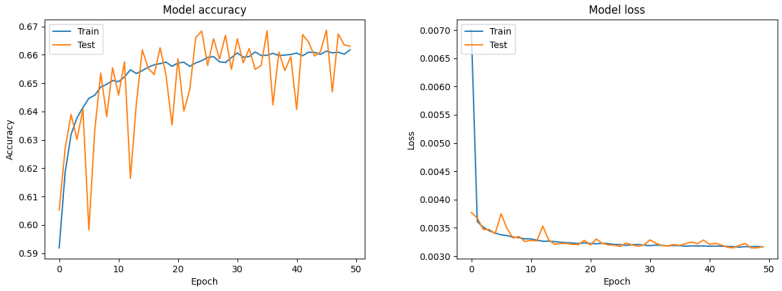
To perform a test on padding padding we used 'valid' to every convolutional layer in our network. After the latent space an UpSampling2D layer was added to grant equal size between input and output layer. Though, the loss on test was not going below 0.02 and the accuracy was steady on 0.70.

3.5 Custom Best Model

The network that we used was different from the standard one in few things. We increased the number of channels of convolutional layers used during the encoding. Whilst reconstructing the image, the Conv2DTranspose layer were introduced. These changes gave us the possibility to reach an accuracy of 0.8402 and a loss of 0.0022 over 50 epochs. The result is available in the fig 4 of the appendix.

4 Colorization

The result obtained from the Autoencoder with grey images were not as expected. The architecture coming from point 3.2 got stuck on 0.67 accuracy and 0.031 loss. The different networks tried in this point did not change much the result. Notwithstanding, the images obtained during the test prediction were not totally negative. Even if they were not fully coloured, there are some areas that got the same tint and shades from the original pic. The result is available in the fig 5 of the appendix.



A Appendix

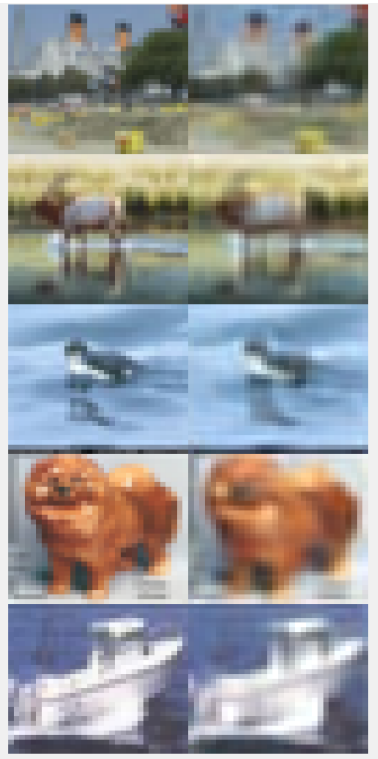


Figure 4: Resulting pic predicted with Custom Best Model

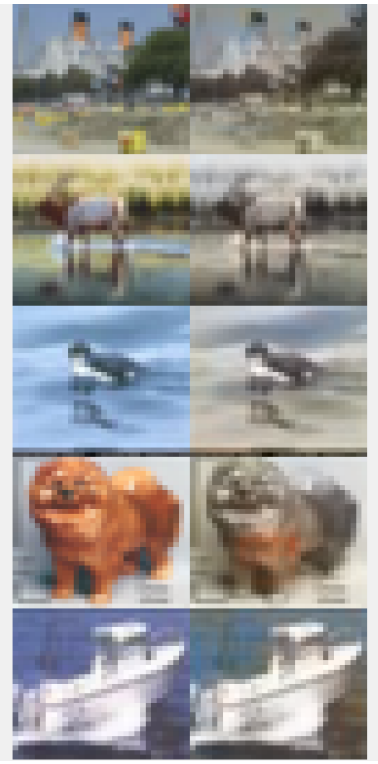


Figure 5: Resulting pic predicted with Colorization