ACML - Convolutional Autoencoders

Dimitrios Gagatsis(6258651), Christos Kaparakis(6258164)

November 10, 2020

1 Introduction

This report provides information obtained by the creation and experimentation on different architectures of Convolutional Neural Networks. This is the objective of the 2nd Assignment for the course Advanced Concepts of Machine Learning.

2 Image Reconstruction

2.1 *Divide your dataset into training (80%), validation (10%) and test (10%). Normalize the data.*

For this assignment, the CIFAR-10 dataset is used. This dataset consinsts of 60.000 thousand 32 × 32 RGB images. The deep learning library we have used to implement the CNN's (Keras) provides the data in a format of 50.000 training images and 10.000 test images.

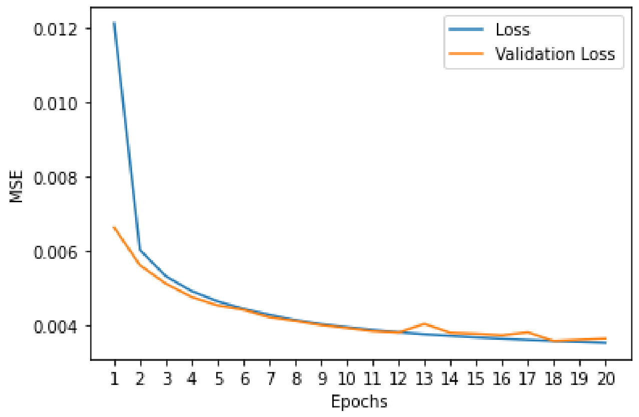
In order to do the correct division for the data, we first concatenate the provided training set and test set which we then break into (80%), (10%) and (10%) in a random manner.

The dataset contains 32 x 32 pixel images with 3 dimension since they are colored. Their intensity (pixel) values range from 0 to 255, so by dividing by 255 we bring the pixel values between 0 and 1, thus normalizing the data.

2.2 *Implement the autoencoder network specified above. Run the training for at least 10 epochs, and plot the evolution of the error with epochs. Report also the test error.*

The specified autoencoder network consists of the input and 9 more layers for the encoder and decoder. The purpose of this network is to get as an input a 32x32x3 RGB image and then output the same RGB image.

We let the network train for 20 epochs and the following figure plots the loss and validation loss for each one of the epochs.



After training the model, we get a 0.0036 evaluation loss for our test data.

3 Latent Space Representation

3.1 *What is the size of the latent space representation of the above network?*

The size of the latent space representation of the above network is 1024

3.2 *Try other architectures (e.g. fewer intermediate layers, different number of channels, filter sizes or stride and padding configurations) to answer questions such as: What is the impact of those in the reconstruction error after training? Is there a correlation between the size of the latent space representation and the error?*

- Removing layers from the network (model 1 and 2)

  By removing the first convolution filter, the latent space size increases to 4096 and after training the network the reconstruction error is 0.0051.

  By removing 3 intermediate layers, the latent space size stays the same at 4096, but the reconstruction error reduces to 0.00056. We have a reduced error because we removed a max pooling layer and an Up-sampling layer which raise it.

- Increasing the number of channels (model 3)

  By taking the original architecture and increasing the channels of every layer till the latent space, but then decreasing the channels of the layer resulting to the output, the latent space size is 8.192 and the reconstruction error is 0.0011.

- Changing the kernel sizes (filters) (model 4 and 5)

  By increasing the kernel sizes from (3,3) in the original architecture to (5,5) we get a latent space size of 576 and a reconstruction error of 0.0033.

  By increasing the kernel sizes from (3,3) in the original architecture to (7,7) we get a latent space size of 256 and a reconstruction error of 0.0032.

- Changing the stride to 3 (model 6)

  The latent space size becomes 16, and the reconstruction error 0.044.

- Changing the padding (model 7)

  The latent space size becomes 400, and the reconstruction error 0.015.

model	*latent space size*	*reconstruction error*
model 1	4096	0.0051
model 2	4096	0.00056
model 3	8.192	0.0011
model 4	576	0.0033
model 5	256	0.0032
model 6	16	0.044
model 7	400	0.015

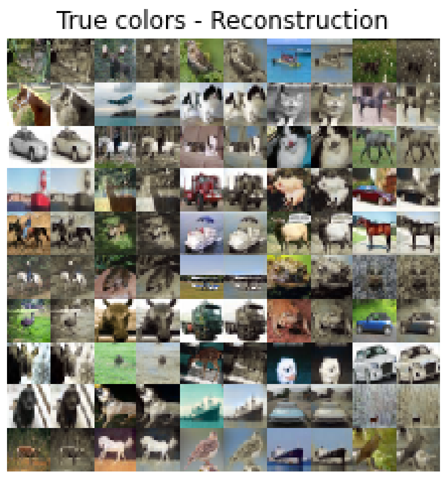
The above table illustrates the size of the latent space and the reconstruction error for each model we tested. We observe that the best model to choose is model 3 because it has a higher latent space size with the least reconstruction error.

4 Colorization

4.1 *Adapting the network from the previous part such that it learns to reconstruct colors by feeding in grayscale images but predicting all RGB channels.*

From the experiments made in the previous section, we conclude that the models with the best performance are the ones with the reduced layers. More specifically, the model 2 is more ideal because it has higher latent space size with the least reconstruction error. Hence, to create the following final model, we will choose the architecture of the second model above.

By choosing this model architecture and changing the number of channels we are getting a reconstruction error of 0.0056. The model is good enough to recognize some colors and shadows from the provided grayscale images but there is room for improvement.



4.2 ***Report on your results and reason about potential shortcomings of your network. What aspects of the architecture/hyperparameters/optimization could be improved upon to fit the model more adequately to this application? Try out some ideas.***

From the experiments done in the previous sections, we saw that also increasing the filter sizes leads to a better model performance. For that reason, we can create a new model to combine both of our previously best ones.

After training the network for 100 epochs we result in a reconstruction loss of 0.0057. In the following example we can see the accuracy of the model.

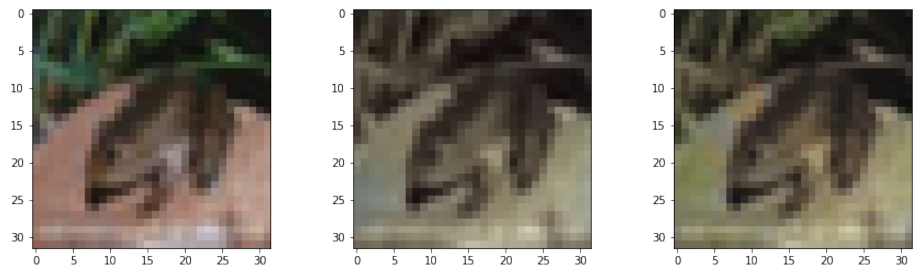


Figure 1: True image, final model 1 reconstruction, final model 2 reconstruction