Assignment 2 Report

Advanced Concepts in Machine Learning

Using a Deep Learning Framework: Autoencoders

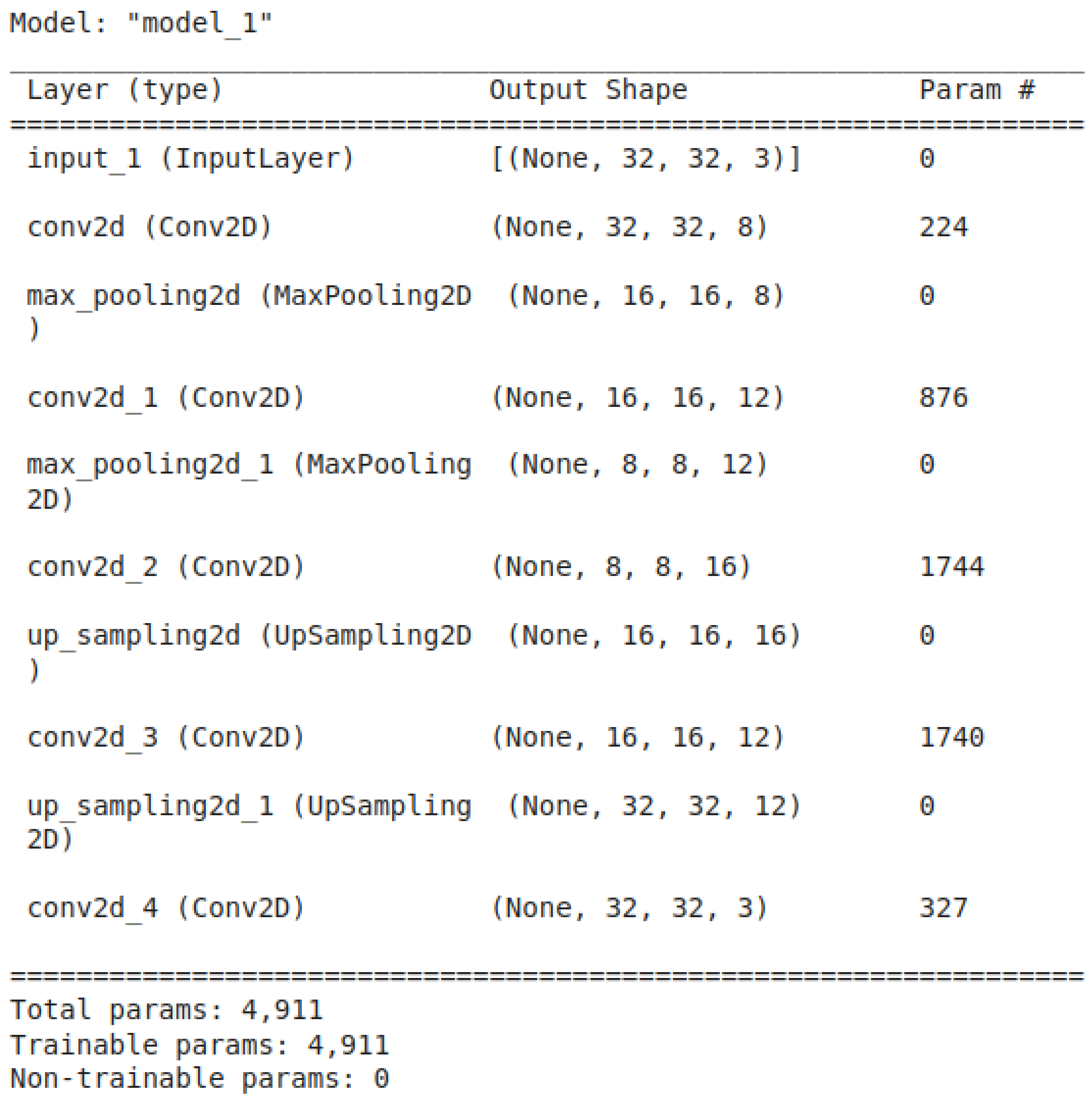
Foti Kerkeshi

November 2022

Image Reconstruction

The architecture

To construct the given architecture in the assignment instructions I used Keras deep learning framework. A summary of the model is shown below, with a total of 4,911 parameters. After each convolutional layer, I applied **ReLu** non-linearity and I made sure to follow the instructions about the padding and stride, meaning that stride was always kept at 1, and padding was 1 on the convolutional layers and 0 on pooling and upsampling layers. The decoder architecture begins from the first UpSampling. Below are also shown the dimensions after each layer operation and the latent space dimensions.



```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 3)]       0

 conv2d (Conv2D)             (None, 32, 32, 8)         224

 max_pooling2d (MaxPooling2D  (None, 16, 16, 8)        0
 )

 conv2d_1 (Conv2D)           (None, 16, 16, 12)        876

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 12)         0
 2D)

 conv2d_2 (Conv2D)           (None, 8, 8, 16)          1744

 up_sampling2d (UpSampling2D  (None, 16, 16, 16)       0
 )

 conv2d_3 (Conv2D)           (None, 16, 16, 12)        1740

 up_sampling2d_1 (UpSampling  (None, 32, 32, 12)       0
 2D)

 conv2d_4 (Conv2D)           (None, 32, 32, 3)         327

=================================================================
Total params: 4,911
Trainable params: 4,911
Non-trainable params: 0
```

Figure 1 *- Initial Model Summary*

Training

For the training, I split the dataset into 3 sets namely the training set, test set, and validation set. Each of them got respectively 50k, 5k, and 5k samples from the whole CIFAR-10 dataset.

In the initial stage with the above-constructed architecture, I used **Adam** as the optimizer with the default parameters and for the loss function, I chose **Mean Squared Error**. Initially, I trained it for 10 epochs with batch size 32, but it was prematurely stopping the learning of the model as the accuracy had the potential to increase even more. Taking advantage of the GPU runtime accelerators from Google Colab, I trained it for 50 epochs and the results were as follows:

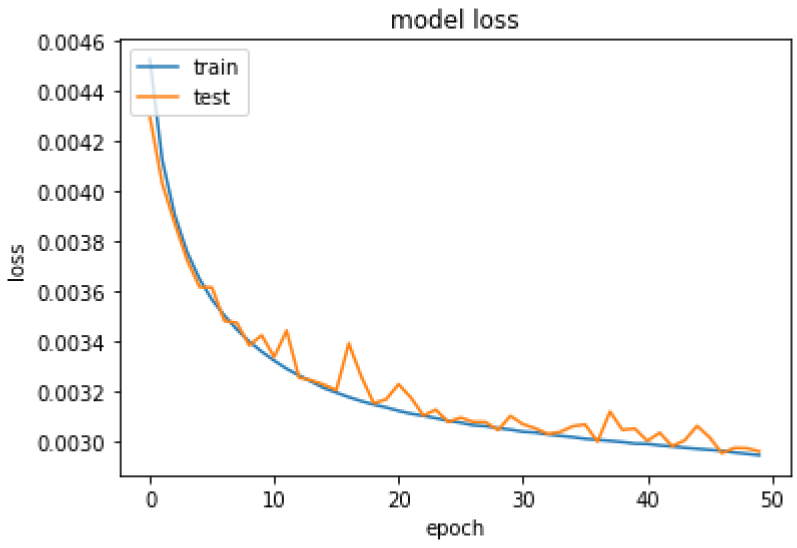


Figure 2 *- Model loss logs for training and validation set*

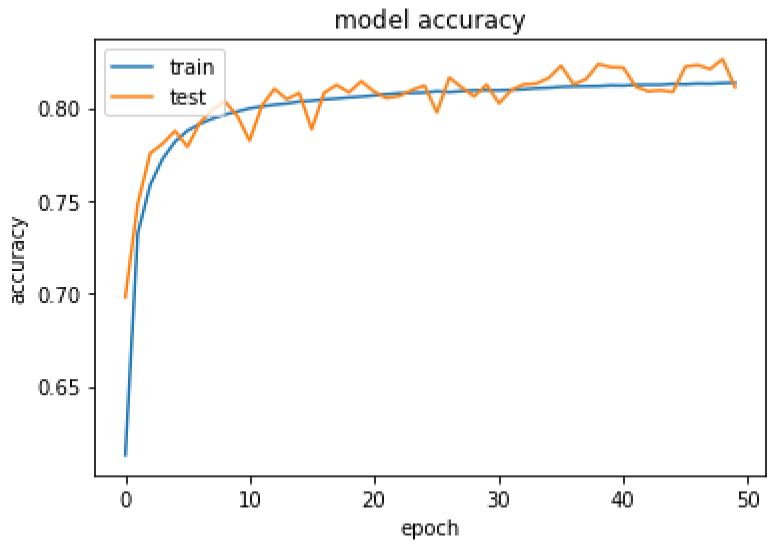


Figure 3 *- Model accuracy logs for training and validation set*

Results

Below are some of the results after the reconstruction of the input image. The model has learned very well to distinguish the object from the background and also the details which have changes of color intensity are also learned very well (e.g., the front leg of the horse and its tail). Although the images look blurry and there are rapid changes of colors between local pixels, the main object of interest is well clear-cut.

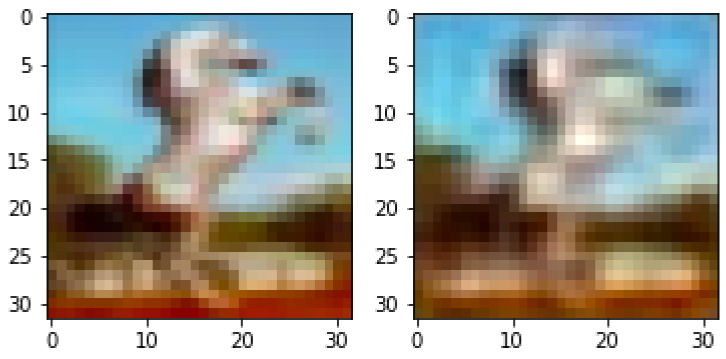


Figure 4 *- Image Reconstruction with AE (horse)*

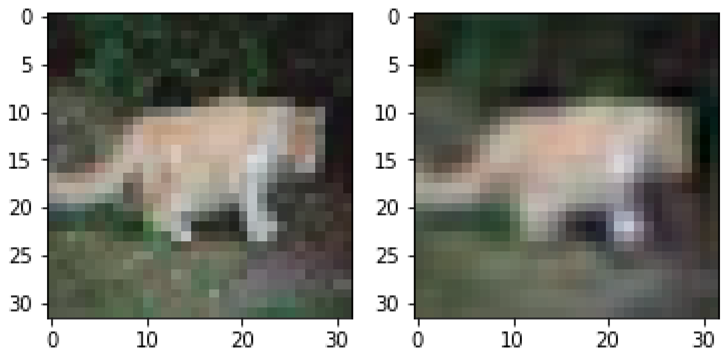


Figure 5 *- Image Reconstruction with AE (cat)*

Other Experiments - testing and observations

1

The first trial I did was to change the Autoencoder architecture by increasing the number of channels in each convolution layer and increasing so the feature map. So the number of channels during the forward pass was symmetric between the encoder and decoder going from 3-8-16-32-32-16-8-3 However, the results did not turn out to be better than the previous one as the test set accuracy was slightly over 80%.

2

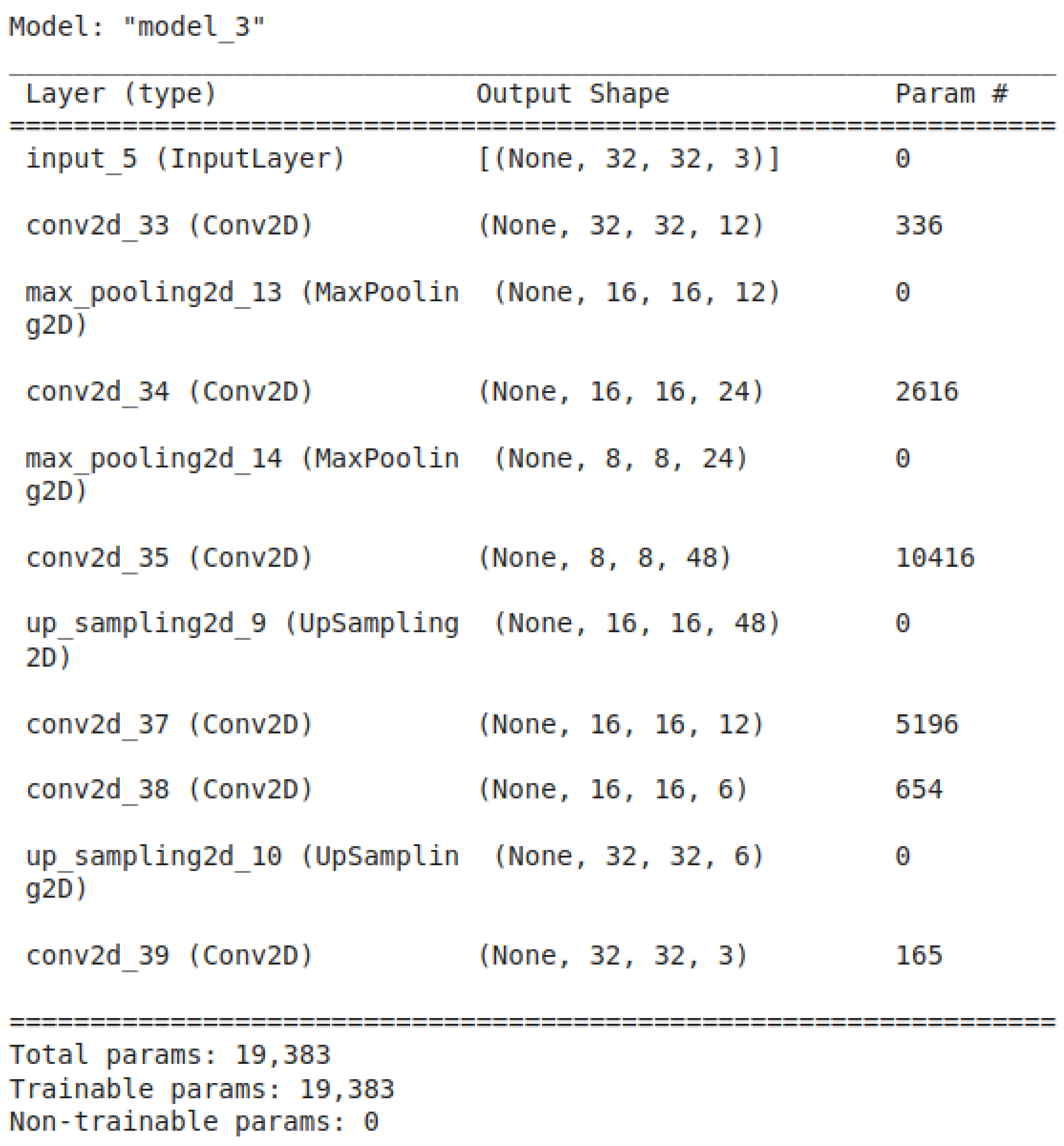
Another thing I tried was to increase the number of convolutional layers and increasingly change the size of the filter from one convolutional layer to another. What I tried to achieve is to induce the spatial-filter hierarchies between convolutional layers such that successive convolutional layers look at increasingly larger windows. This is important in our case problem as we want to initially obtain information from smaller local patches and then obtain information about the entirety of the input. The results were more or less the same as the previous trials, and the reason for that is that the input samples are too small to have any added information by constantly increasing the window size.

3

I also did a quick trial by changing the MaxPool Layer to Average Pooling. However, the results were pretty decent. The reason for that might be that Max Pooling looks for the maximal presence of a certain feature in the image, which is very important in this case to learn the characteristic features that distinguish between classes. Using Average Pooling the model was missing the presence of certain feature information.

4

The last trial from which I got the best results was a refinement of the #1 trial, but in addition, I also included two more convolutional layers. Doing so the size of the *latent space representation is increased*, meaning that it is strongly correlated with the measured error. A bigger latent space provides more input information for the decoder to reconstruct the image. The final model architecture and the results are described below. With this new architecture, I was able to achieve the results from the initial architecture within the first 20 epochs (instead of 50 used before).



```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 32, 32, 3)]       0

 conv2d_33 (Conv2D)          (None, 32, 32, 12)        336

 max_pooling2d_13 (MaxPoolin  (None, 16, 16, 12)       0
 g2D)

 conv2d_34 (Conv2D)          (None, 16, 16, 24)        2616

 max_pooling2d_14 (MaxPoolin  (None, 8, 8, 24)         0
 g2D)

 conv2d_35 (Conv2D)          (None, 8, 8, 48)          10416

 up_sampling2d_9 (UpSampling  (None, 16, 16, 48)       0
 2D)

 conv2d_37 (Conv2D)          (None, 16, 16, 12)        5196

 conv2d_38 (Conv2D)          (None, 16, 16, 6)         654

 up_sampling2d_10 (UpSamplin  (None, 32, 32, 6)        0
 g2D)

 conv2d_39 (Conv2D)          (None, 32, 32, 3)         165

=================================================================
Total params: 19,383
Trainable params: 19,383
Non-trainable params: 0
```

Figure 6 *- Best Model Summary*

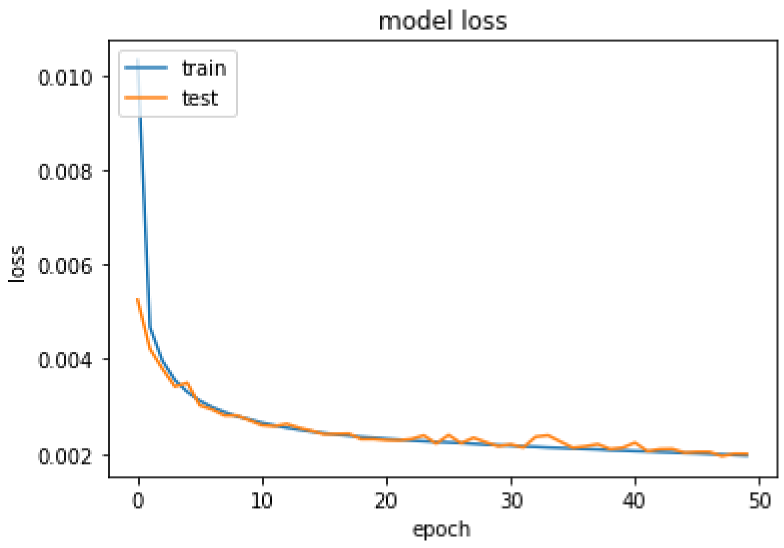


Figure 7 *- Model loss logs for training and validation set*

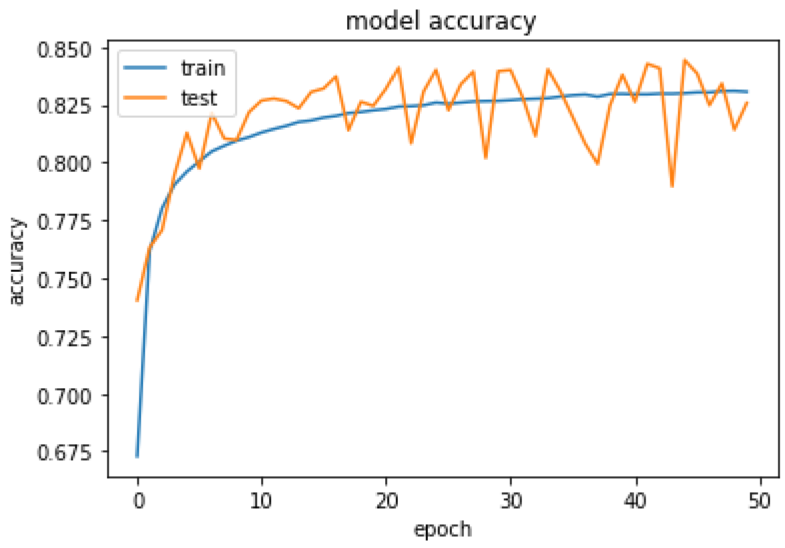


Figure 8 *- Model accuracy logs for training and validation set*

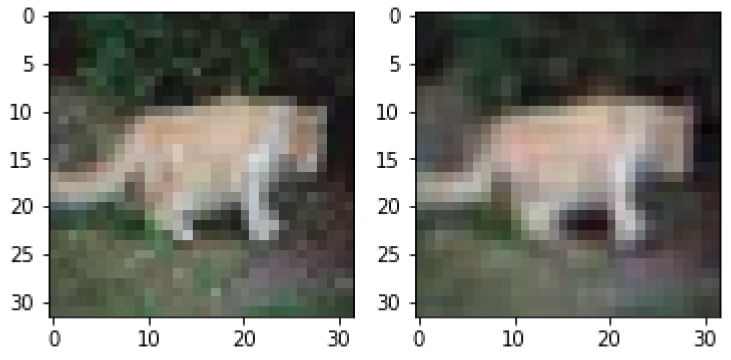


Figure 9 *- Image reconstruction with AE (best architecture)*

Comparing the reconstructed image with the previous results, we can visibly tell the success of the second architecture. For example, if we analyze the cat image, we can see that the front leg is now clearly visible and the pixels describing the gap between the two legs are not interfering with the surrounding pixels as much as in the first picture. Below are some other image reconstruction results inferred from this model.

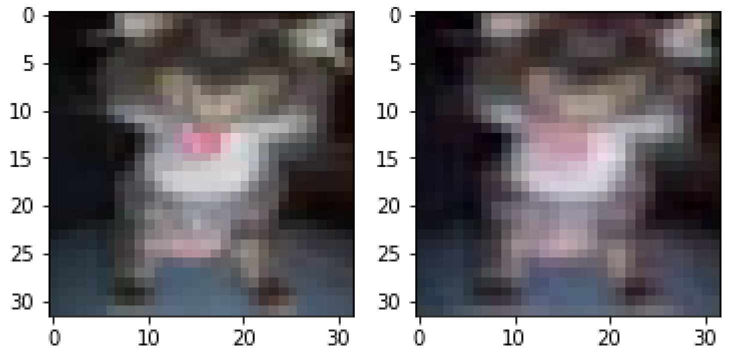


Figure 10 *- Image reconstruction with AE (best architecture)*

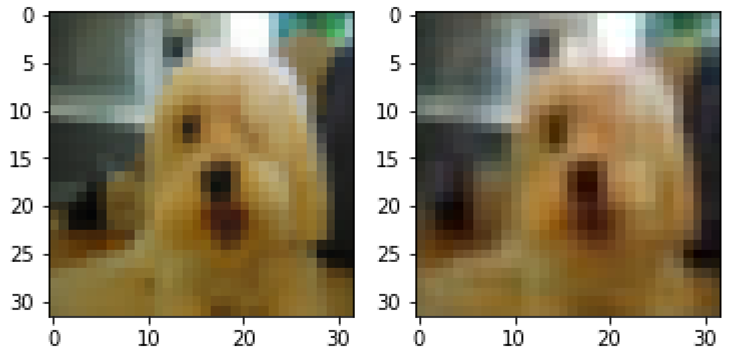
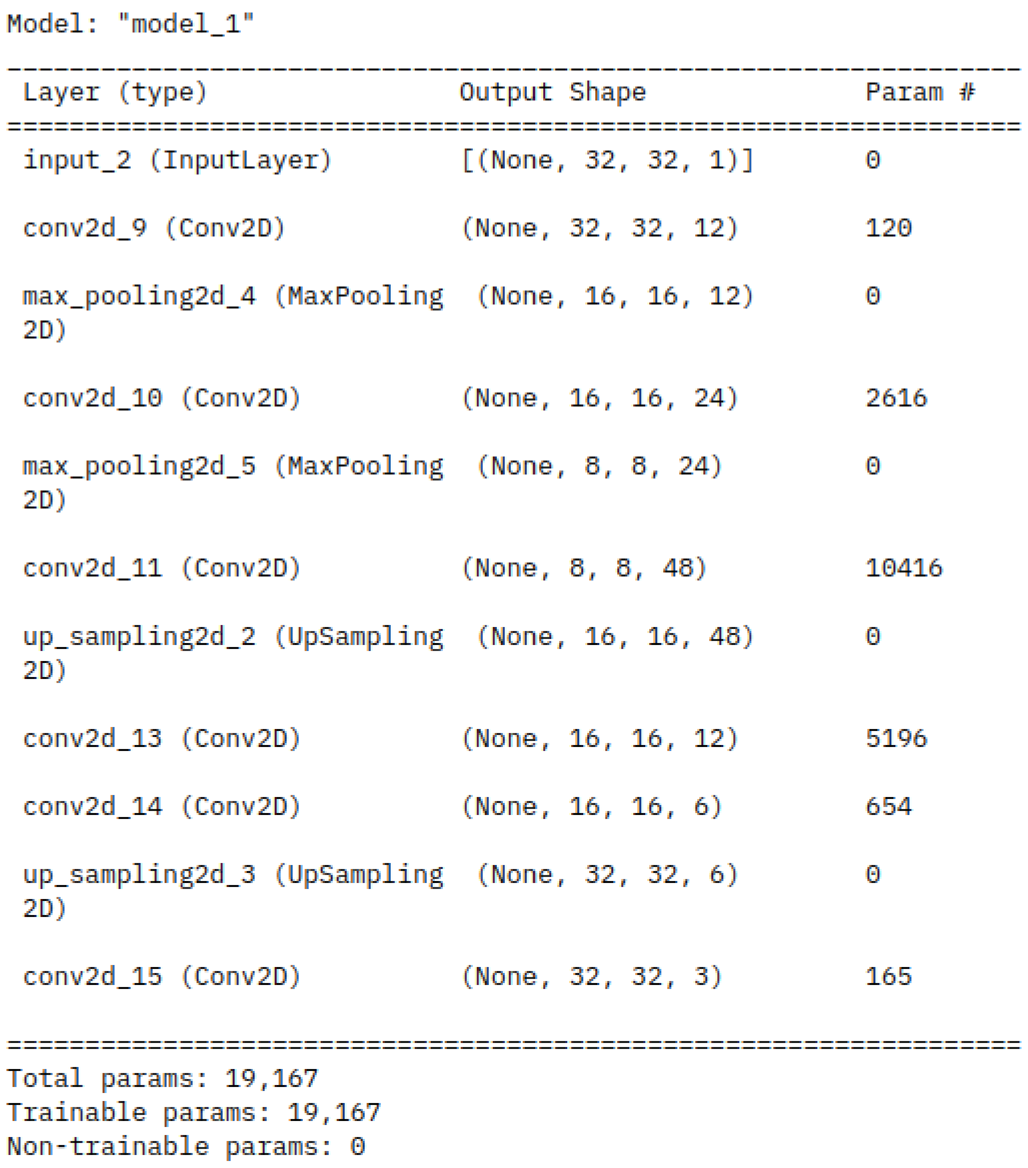


Figure 11 - *Image reconstruction with AE (best architecture)*

Colorization

The architecture

For the colorization assignment I used the best model architecture I found previously for the image reconstruction. The only thing that changed here is the input layer which now has 1 channel. I also split the datasets into training, test, and validation, and converted the input samples into grayscale images. The model summary can be found below, as well as the history plots after training with 30 epochs.



```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 32, 32, 1)]       0

 conv2d_9 (Conv2D)           (None, 32, 32, 12)        120

 max_pooling2d_4 (MaxPooling  (None, 16, 16, 12)       0
 2D)

 conv2d_10 (Conv2D)          (None, 16, 16, 24)        2616

 max_pooling2d_5 (MaxPooling  (None, 8, 8, 24)         0
 2D)

 conv2d_11 (Conv2D)          (None, 8, 8, 48)          10416

 up_sampling2d_2 (UpSampling  (None, 16, 16, 48)       0
 2D)

 conv2d_13 (Conv2D)          (None, 16, 16, 12)        5196

 conv2d_14 (Conv2D)          (None, 16, 16, 6)         654

 up_sampling2d_3 (UpSampling  (None, 32, 32, 6)        0
 2D)

 conv2d_15 (Conv2D)          (None, 32, 32, 3)         165

=================================================================
Total params: 19,167
Trainable params: 19,167
Non-trainable params: 0
```

Figure 12 - *Colorization model architecture*

Results

As shown below the model accuracy on the validation set was fluctuating during the training, meanwhile the training accuracy was increasing and reached a convergence point after 50 epochs, with a training accuracy of slightly more than 50%.

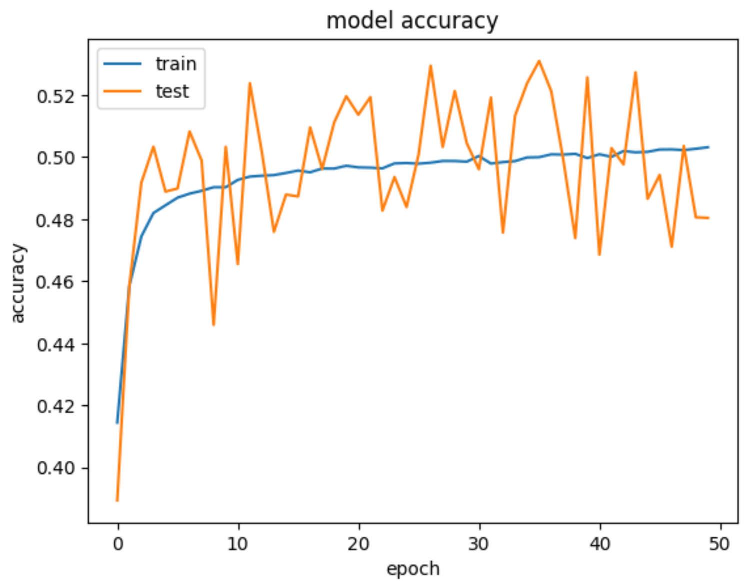


Figure 13 - *Model loss logs for training and validation set*

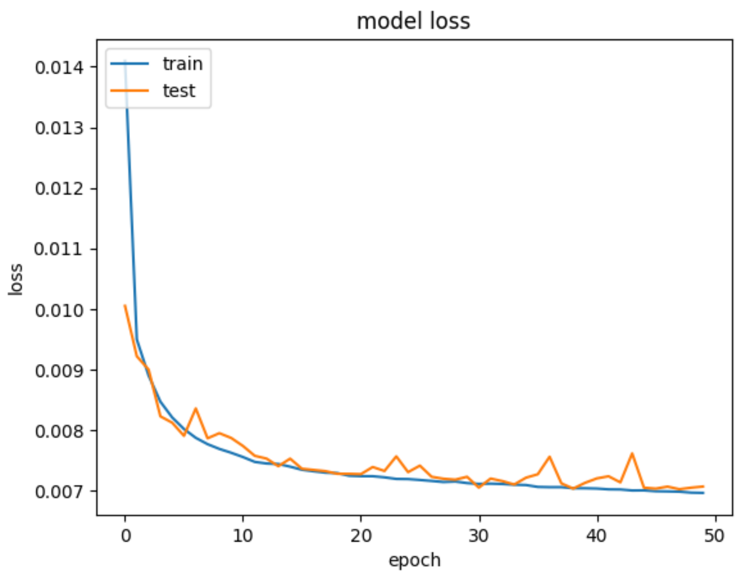


Figure 14 - *Model accuracy logs for training and validation set*

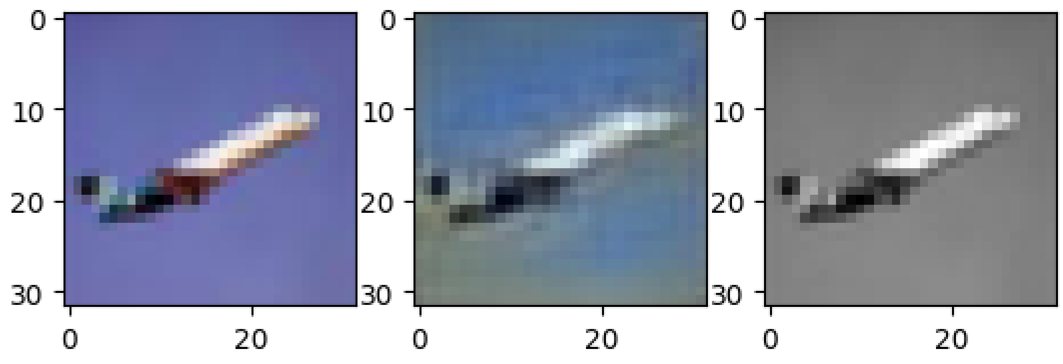


Figure 15 - *Colorization showcase #1 (left - original image, middle - colored image, right - grayscale image)*

Analyzing the image above, we can see that the model has learned how to discriminate the objects from the background, e.g. in our case the model has learned to colorize differently the skty and the plane. In addition, it also has learned to color the sky blue, althought the color intensity is one the weak points of it.

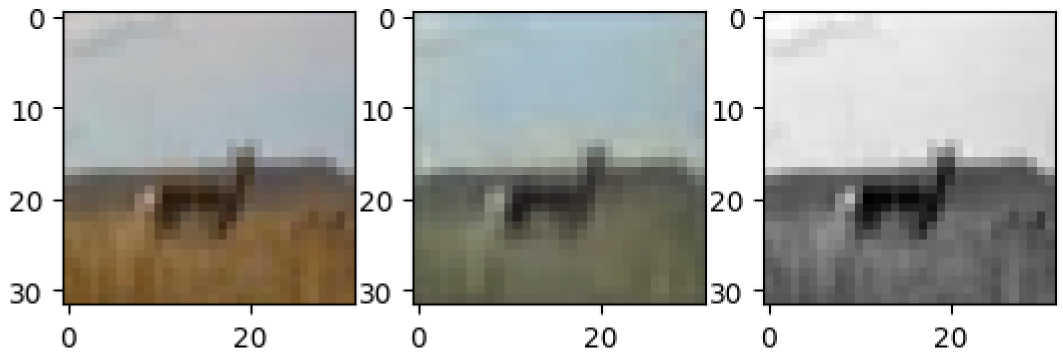


Figure 16 - *Colorization showcase #2 (left - original image, middle - colored image, right - grayscale image)*

A shortcoming of the current architecture is the encoder part which is not very deep in order to get all the global features of the image, which determine the local colorization of different parts of the image. However, the trade-off is that we don't want to go very deep as latent representations will be too small for the decoder. The results show that the model has learned to distinguish different objects in the image yielding different color intensities for them.

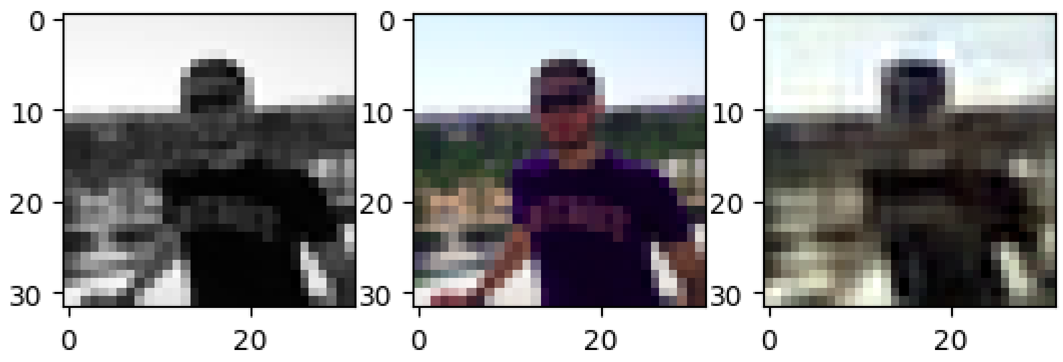


Figure 16 - *Colorization showcase #3, picture of myself not part of the dataset (left - grayscale image, middle - original image, right - colored image)*

A summary of the accuracy metric from all the models

Model	**Training**	**Test**
IR initial model*	80.5%	81.2%
IR best model*	83.2%	84.4%
colorization model	50.3%	48.6%

**IR - image reconstruction*