Deep Learning Homework #6

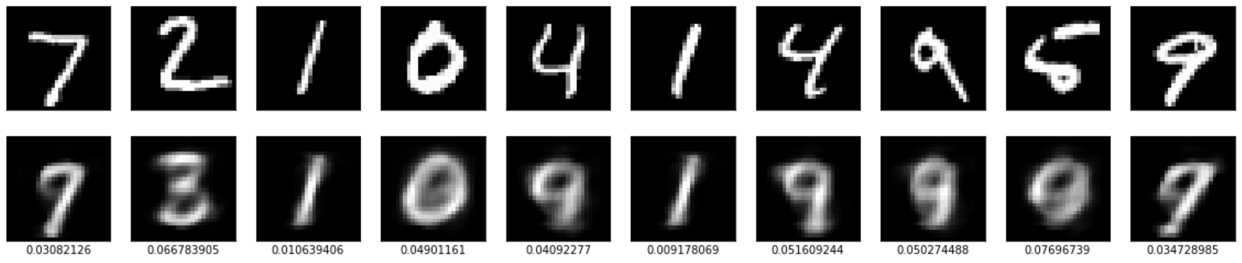
Indra Imanuel Gunawan / 20195118

No. 1 – Autoencoder on MNIST & CIFAR-10 Dataset

A. Run the experiment & plot the reconstruction error

The code for this experiment can be found in the "AutoEncoder_MNIST.ipynb" file. In this experiment, I ran the code from the link provided in the question. The code is for building an autoencoder with the MNIST dataset, in which the autoencoder will try to reconstruct the MNIST images. I modify the code a little bit by changing the loss from "binary_crossentropy" to "mse" (Mean Squared Error). I use MSE to calculate the reconstruction error.

The image below is the result of the experiment, the first row is the original image, the second row is the reconstructed images, and the numbers below them are the reconstruction error for each images (calculated using MSE).



As we can see from the image above, the more similar the reconstructed image to the original image, the lower the MSE for that image. On the other hand, if the reconstructed image differs greatly with the original image (different label even), the MSE will be higher.

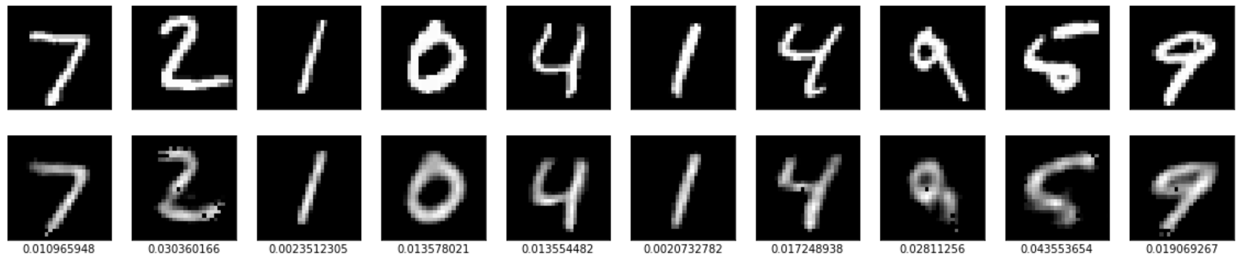
B. Improving the autoencoder performance on the MNIST dataset

The code for this experiment can be found in the "AutoEncoder_MNIST_Improved.ipynb" file. To improve the autoencoder performance in reconstructing the MNIST dataset, I made several changes to the network:

1. I change the activation function from "sigmoid" to "relu". This increases the performance of the autoencoder, but not by much. I think this increases performance, as ReLU does usually perform better than sigmoid, because it doesn't saturate (in the positive area).
2. I increase the number of neurons in the hidden layer. Initially, the number of neurons in the hidden layer are 3, I increased it to 10 neurons. This in turns, increase the autoencoder performance quite a lot. I think this is because the more neurons it has, the more features it can extract in details.

3. I increase the number of epochs for the training, from 5 epochs to 10 epochs. This also improves the autoencoder performance, as it seems longer training on the autoencoder will make it generate better images.

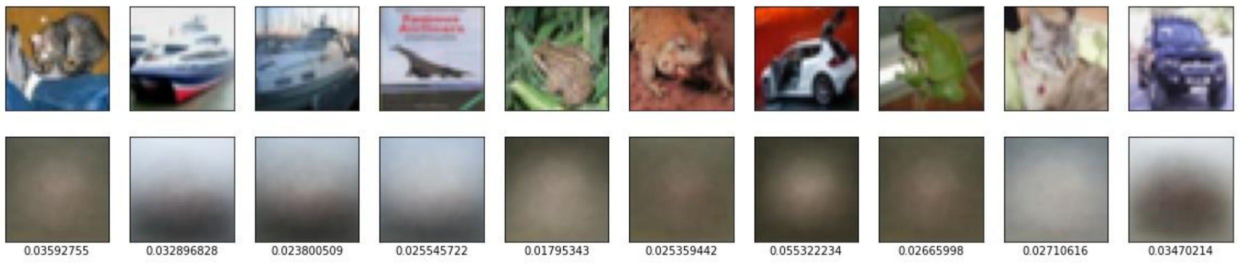
Here are the results of my autoencoder reconstructed images after I change the parameters:



As we can see from the image above, the reconstructed image is better if compared with the network without the parameter modification. This will also result in turns, result in lower MSE.

C. Running the autoencoder on CIFAR-10 dataset

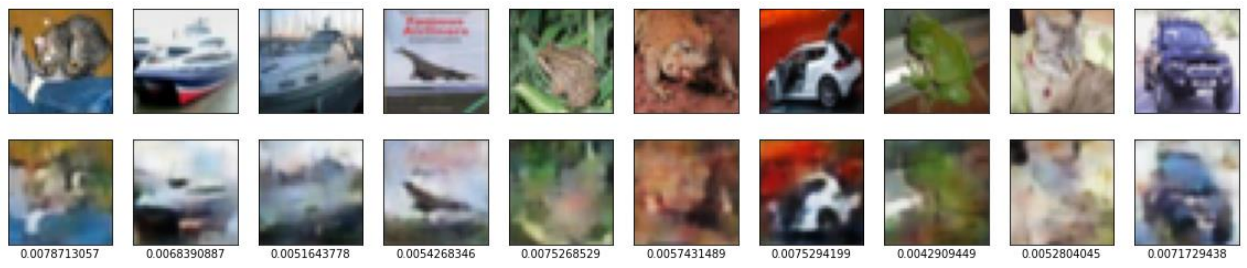
The code for this experiment can be found in the “AutoEncoder_CIFAR10.ipynb” file. The next experiment is running the same network as in the first problem (before parameter modification) on the CIFAR-10 dataset. The only difference I made is I use 20 epochs in the training process. CIFAR-10 data is RGB data, and the image is also more complex than MNIST. Because CIFAR-10 data is more complex than MNIST, the reconstructed image result is not as good. Here are the reconstructed images for the CIFAR-10 data.



From the result above, we can see that the autoencoder reconstructed images differ greatly with the original image. This is due to how complex CIFAR-10 data are, making a simple network structure for the autoencoder is not enough.

D. Improving the autoencoder performance on the CIFAR-10 dataset

The code for this experiment can be found in the “AutoEncoder_CIFAR10_Improved.ipynb” file. To improve the autoencoder performance on the CIFAR-10 dataset, a big network structure change is necessary, because a small tweak to the model won’t really increase the performance of the autoencoder by much. What I did is I completely remove all the Dense layer, as it seems fully connected layer is not performing well on the CIFAR-10 dataset, and change it with convolutional layer. Other than that, I also added batch normalization, and upsampling. I also made it a denoising autoencoder, in which I added random noise to the original image, and the autoencoder is trying to denoise that. This in turns improved the autoencoder performance greatly. Here are the reconstructed images from the autoencoder:



As we can see from the image above, the reconstructed images are now more similar to the original image compared with the autoencoder from before. This is due to the convolutional layer perform better than the Dense layer. Batch Normalization, Upsampling, and De-noising also help in increasing the performance.