**Convolution Autoencoder with MNIST Dataset**

Saroj Kumar, Kaushik Das, Simon Callewaert, Rumi Asano

Introduction:

We are creating a convolution autoencoder with TensorFlow. In this project we use the random noise to corrupt the image. Then using the autoencoder we recover the image from the corrupted image. We use the mnist dataset as input to our program.

Requirements:

To execute this python program, you need to install Tensorflow. This program will use following python modules tensorflow, numpy, math, and matplotlib. To develop and execute the software we used Anaconda Navigator 1.8.7, Jupyter Notebook 5.5.0. This program will use MNIST database as input data for the program. It will use pyplot from matplotlib for generating output.

Descriptions of the Python program:

There are multiple functions and one main module in this program. Here is the list of the functions:

1. corrupt

This function will accept a tensor placeholder as input. Then it will add random noise to corrupt 50% of the values in the tensor and return the value to the caller module.

1. relu

This function will accept a tensor placeholder and optional parameters leak factor and name. It will modify the tensor to non-liner values using the optional parameter leak factor. Then it will return the non-linear tensor to the caller module.

1. autoencoder

This is the main function which build the autoencoder. It takes following inputs: input shape, number of filters, filter size and corruption It will return tensor for the input placeholder, tensor for inner-most latent representation, tensor for the reconstruction of the input and overall cost for the training.

1. test\_mnist

This function will test the convolution autoencoder built in autoencoder function using mnist input data. First it will extract the mnist input data. Then it will call autoencoder function process the input data for 10 epochs in 100 samples per batch. Then it will display the cost of each epoch. It will create a TensorBoard image which will show TensorFlow runs and graph. It will also later plot the original and reconstructed image.

1. \_\_main\_\_

This is the main entry point of the program which will call the function test\_mnist.

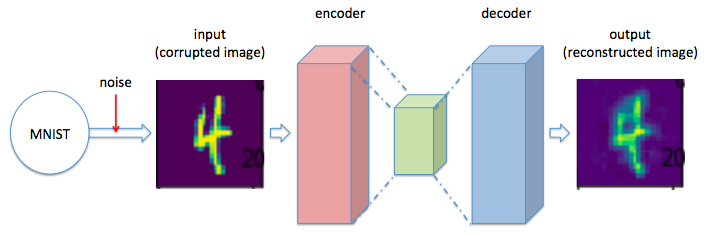


Figure 1: Architecture of the convolution autoencoder

Screenshot of the program output:

Here is the screenshot of the program output:

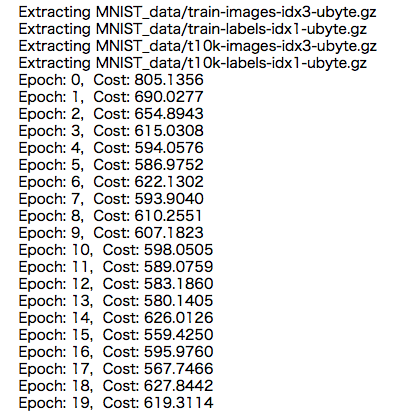


Figure 2: Cost output

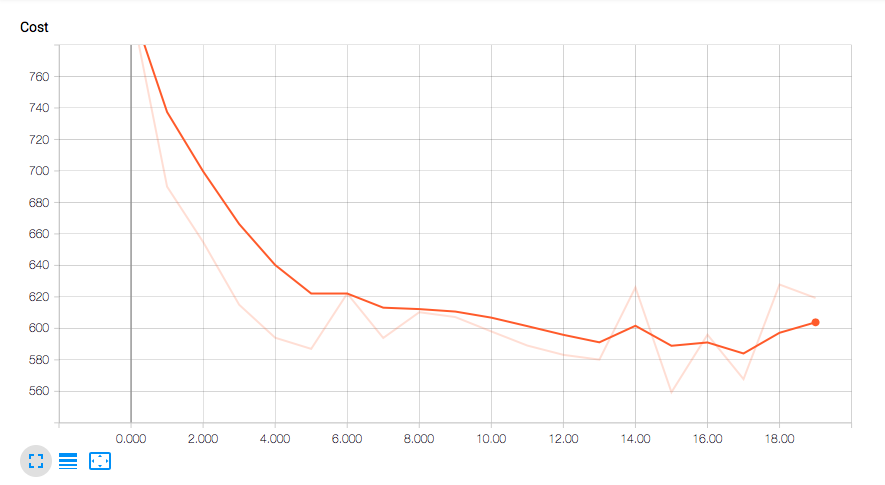


Figure 3: Cost graph

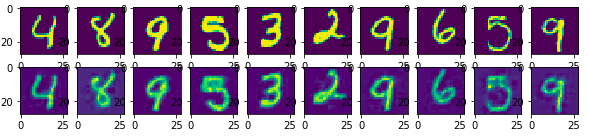


Figure 4: Corrupted images(upper) as input and reconstructed images(lower) as output

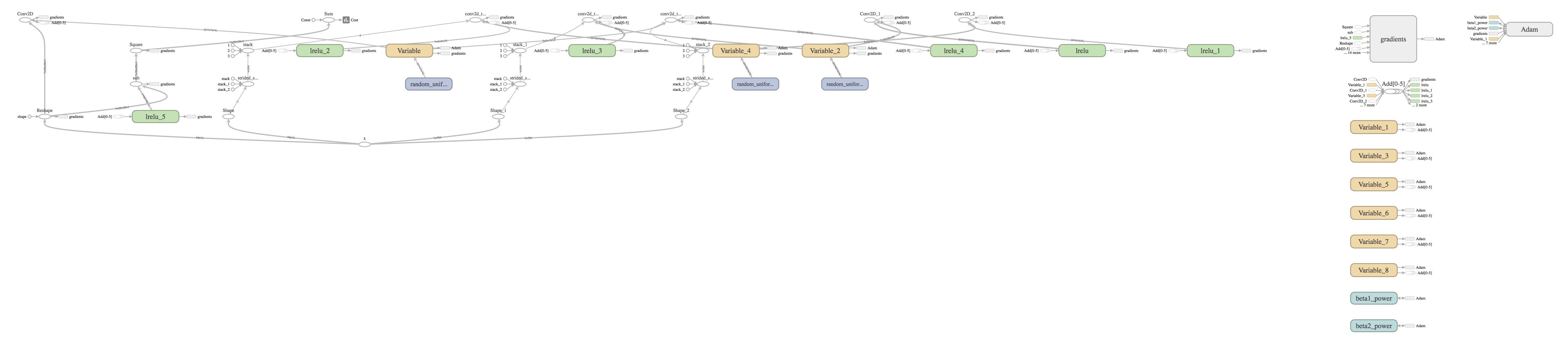


Figure 5: Computation graph (TensorBoard output)

Conclusion:

In this project we used the convolution autoencoder that denoised the input picture. We recovered the images from the corrupted images. We were able to train and capture the output of the result successfully.

Python program:

Corresponding python program has been attached with this report.