**Autoencoder Project**

**I. Project requirements:**

Develop a multilayered autoencoder for removing noise from the dataset. As it is

shown, the noise is from the fashion MNIST. The goal of this project is to design a

network such that it removes the noise from the image to the extent that a person can

correctly determine the digits. The choice of the architecture of the autoencoder is

entirely up to you, however no more than 3,600 training parameters shall be used.

**II. Project designs and developments:**

After linking all the basic libraries, the project implementations were started by loading dataset into the working environment using pickle load. Some quick inspections were performed as normally required to ensure that the provided data set was clean, so no cleaning data processing was needed.

Graphical user interface, application

Description automatically generated

In order to have a better understanding of the data set, the first element of X and y data frame were plotted to visualize how the model should interpret this particular data set.

Chart

Description automatically generatedChart, histogram

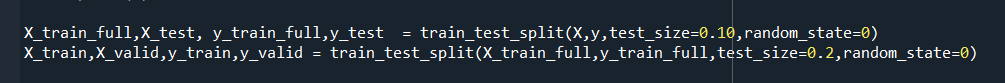
Description automatically generated

From the image observations, it can be seen that X contained images with noises while y contained clean images. The shape of each image was 28x28. The challenge of this project was to denoise the data set containing noisy images with shape of (28,28) into clean images in order to visualize the digits by human eyes. Specifically, the project’ mission was to denoise X data and mimic the outcomes as close as y data as possible.

The next step was to reshape the images in data set in order to feed the training model. Additionally, the X data set also needed to be normalized into range [0,1] to prevent the loss function from potentially returning negative value because the range of value in y data is in range [0,1]

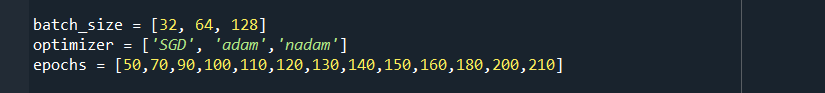


After performing data rescaling process, the data set was then split into training, valid, and test as usual using train\_test\_split



The data set was ready to be trained now. Next step was to design a model to fit the training set and perform the denoising autoencoder. Based on the nature of images’ shapes, the convolutional autoencoder was chosen as the designed model of this project. In addition, because the project was provided with noisy images set and clean images set, the methodology of adding Gaussian’s noise would not be necessary because the training process would take the noisy images directly from the X set. Therefore, simply having a convolutional autoencoder to train noisy and clean images set as X and y would be enough to produce a decent outcome.

After the model was created, a list of parameters was generated to perform the Grid Search when compiling the model. Because the number of epochs was important to prevent the model from overfitting and underfitting; therefore, the main focus of the Grid Search was the number of epochs, hence the list of epochs has the most elements.



During the Grid Search stage, there was an error raising that y shape was invalid. Despite further research, the issue was not solved. My assertion was that in this project the shape of y was different from the last project where y was one hot encoded to array of 0 and 1. This issue raised a new challenge on how to Grid Search the parameters.

After reviewing the purposes of Grid Search, an alternative methodology was used. The model would take each of the parameters in the list and then compiled accordingly using a nested for loop.

Text

Description automatically generated

The model then was able to be trained according to each combination of parameters. After the Grid Search was finished, each combination was carefully reviewed and picked manually to ensure that the picked parameters would be the best combination among the available grid parameters.

The following figure represented one of the Grid search actual running trials.

Text

Description automatically generated

After the GridSearch was conducted and recorded, the code was then updated to create a model with the best tuned parameters. The model was then trained and saved to Model4.h5 using Keras.

In order to generate the Model4.h5 file, the checkpoint and early stopping methodologies were used in order to have an early stop when necessary after no progress on validation set for 5 consecutive number of epochs and to save the best available model.

After the model was loaded back from the Model.h5 file, a quick overview was conducted to see if the Model was saved successfully.

Model.sumarry() was used to summarize the designed model, including each layer’s parameters.

A picture containing graphical user interface

Description automatically generated

The total number of parameters were less than 3600, indicating that the design requirements were met.

The next step was to test on model’s performance against the test set. A function called plot was created to plot the denoising images from the test set to see if the autoencoder was designed correctly.

Text

Description automatically generated

A quick inspection on the generated plot of first 10 images, showing that the images were able to be detected by human’s eyes. The model was successful denoising the noisy images.

A picture containing text

Description automatically generated

Generally, in order to confidently estimate the outcome against a new test set, it would be necessary to answer the question that whether the model was underfitting or overfitting. Looking back into the implementations to answer this question, the training loss was not higher than the validation loss; therefore, the model was not underfitting. The training loss and validation loss have also been plotted to see how things went and whether the model converged.

Text

Description automatically generated

From the generated plot, it can be said that the model was not overfitting. The training loss has gone down, and the validation loss also decreased. The validation loss seemed to converge at 0.1310; hence, the model converged. The number of epochs was 100 because increasing the number of epochs would not result in a better outcome because the model has converged; it would likely result in overfitting the model. However, if the number of epochs were decreased, the model would be underfitting and would not be trained well enough to perform against the test set. To better demonstrate the conclusion on number of epochs, it was once set to 150 and the early stopping function from Keras called a stop at epoch 103/150 with the validation loss value at 1.1310, indicating that the convergence of the model was correct at 100 epochs.

Graphical user interface

Description automatically generated with medium confidence

My estimation was that the model would likely to generate 78% of the images that would be able to be detected by human’s eyes.

**III. Discussions:**

The model was able to denoise the images; however, the results were not as clean as the y set, leaving room for improvements.

Looking back into the design process, the model was hard coded to be a convolutional autoencoder. There were other models that could be picked; hence, if more research were conducted among the available models, it would likely that there was a model that better than the others. For the Grid search stage, the search was just searching for a limited number of parameters from a desired list, there were so many available parameters that were neglected. If it were done more precise, the outcome could be more promising. Given that the parameters were bounded by 3,600; if the total number of parameters were not limited, the model can be trained with a higher cap because the difference in the number of combinations of layers and the size of filters. The limitations in resource required students to have a proper design and to understand the implementations thoroughly, that would be useful when given a better computing resource in further research.