Assignment 4

Autoencoders/LSTM networks
ELE494-09

Nasir Khalid
b00065082

I. Autoencoders and dimensionality reduction

We use the fashion MNIST dataset and train a densely connected autoencoder with one hidden layer. First with 32 nodes and 64 nodes in this layer with a rectified linear unit as the activation function. In the output layer I have a sigmoid activation function. We then train the autoencoder under squared error reconstruction loss function. Then I test the networks with the images in the testing data, reporting the average reconstruction error for different number of nodes in the hidden layer. We then repeat the same with autoencoders with three hidden layers use 32 and 64 as the dimensionality of the codes

A. Preprocessing

The data was first imported using Keras and the dimensions of the training/test data were as follows:

- Size of Training Images is (60000, 784)
- Size of Test Images is (10000, 784)

The pixel values were then normalized between 0 and 1. Shown below is a sample image from the Fashion MNIST dataset.

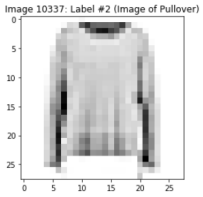

Figure 1: Fashion MNIST sample image

A function was also defined for the average reconstruction error and this function was then passed as a metric in to the networks compile method, this way the average reconstruction error was returned on every training epoch. It was defined as follows:

$$\frac{1}{Length}\sum|Output - Prediction| \tag{1}$$

Where Length is the output length, Output is label output and prediction is result of forward pass.

B. Training Encoder with 1 Hidden Layer that has 32 Nodes

Shown below is the summary of the Encoder created with 1 Hidden Layer that has 32 Nodes

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_38 (Dense)             (None, 32)                25120
_________________________________________________________________
activation_35 (Activation)   (None, 32)                0
_________________________________________________________________
dense_39 (Dense)             (None, 784)               25872
_________________________________________________________________
activation_36 (Activation)   (None, 784)               0
=================================================================
Total params: 50,992
Trainable params: 50,992
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Summary of Encoder with 1 Hidden layer with 32 Nodes

This encoder was trained up 50 epochs and in the end these were it's final metrics:

- Loss: 0.0128
- Average Reconstruction Error: 0.0644
- Validation Loss: 0.0130
- Validation Average Reconstruction Error: 0.0650

Shown below is the Epochs vs Loss curve for the network:

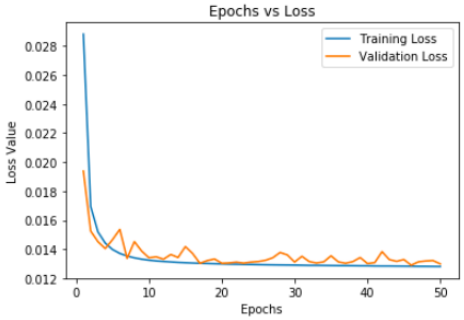

Figure 3: Epochs vs Loss curve for encoder with 1 Hidden layer with 32 Nodes

Shown below is the Epochs vs Average reconstruction error for the network:

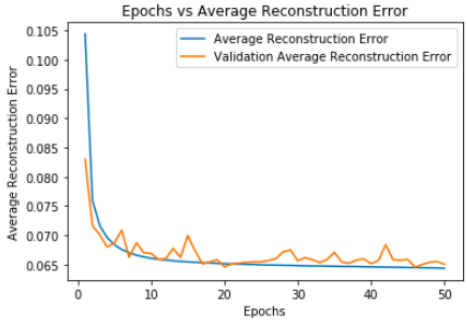

Figure 4: Epochs vs Average Reconstruction Error for encoder with 1 Hidden layer with 32 Nodes

This encoder was then tested on the seperate test and these were the resulting metrics:

- Loss: 0.01296
- Average Reconstruction Error: 0.06502

Shown below is an example image and it's reconstruction from the autoencoder:

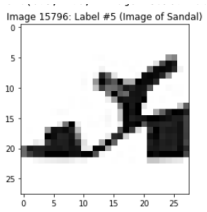

Figure 5: Sample Image from Dataset

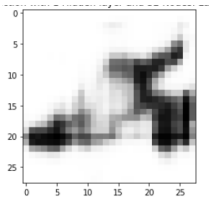

Figure 6: Reconstruction from encoder with 1 Hidden layer with 32 Nodes

C. Training Encoder with 1 Hidden Layer that has 64 Nodes

Shown below is the summary of the Encoder created with 1 Hidden Layer that has 64 Nodes

Layer (type)	Output Shape	Param #
dense_40 (Dense)	(None, 64)	50240
activation_37 (Activation)	(None, 64)	0
dense_41 (Dense)	(None, 784)	50960
activation_38 (Activation)	(None, 784)	0

Total params: 101,200
Trainable params: 101,200
Non-trainable params: 0

Figure 7: Summary of Encoder with 1 Hidden layer with 64 Nodes

This encoder was trained up 50 epochs and in the end these were it's final metrics:

- Loss: 0.0089
- Average Reconstruction Error: 0.0523
- Validation Loss: 0.0091
- Validation Average Reconstruction Error: 0.0530

Shown below is the Epochs vs Loss curve for the network:

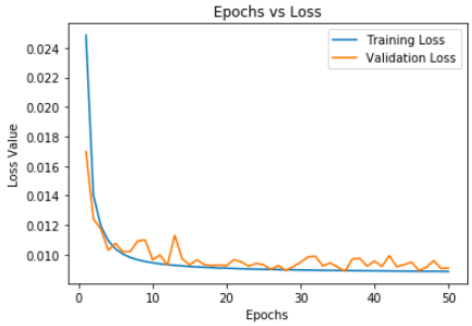

Figure 8: Epochs vs Loss curve for encoder with 1 Hidden layer with 64 Nodes

Shown below is the Epochs vs Average reconstruction error for the network:

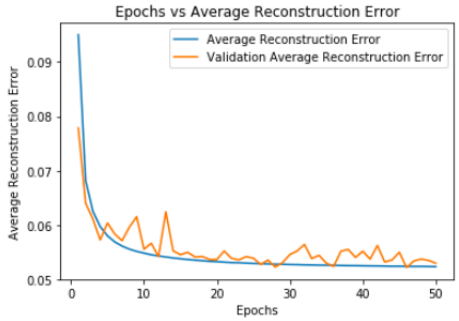

Figure 9: Epochs vs Average Reconstruction Error for encoder with 1 Hidden layer with 64 Nodes

This encoder was then tested on the seperate test and these were the resulting metrics:

- Loss: 0.00907
- Average Reconstruction Error: 0.05293

Shown below is an example image and it's reconstruction from the autoencoder:

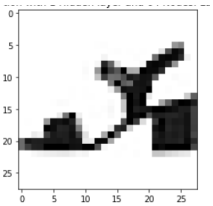

Figure 10: Sample Image from Dataset

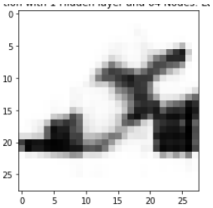

Figure 11: Reconstruction from encoder with 1 Hidden layer with 64 Nodes

D. Training Encoder with 3 Hidden layers, main layer has 32 Nodes

Shown below is the summary of the Encoder created with 3 Hidden Layers and the main one having 32 Nodes

Layer (type)	Output Shape	Param #
dense_42 (Dense)	(None, 128)	100480
activation_39 (Activation)	(None, 128)	0
dense_43 (Dense)	(None, 32)	4128
activation_40 (Activation)	(None, 32)	0
dense_44 (Dense)	(None, 128)	4224
activation_41 (Activation)	(None, 128)	0
dense_45 (Dense)	(None, 784)	101136
activation_42 (Activation)	(None, 784)	0

Total params: 209,968
Trainable params: 209,968
Non-trainable params: 0

Figure 12: Summary of Encoder with 3 Hidden layer and main layer having 32 Nodes

This encoder was trained up 50 epochs and in the end these were it's final metrics:

- Loss: 0.0098
- Average Reconstruction Error: 0.0513
- Validation Loss: 0.0099
- Validation Average Reconstruction Error: 0.0512

Shown below is the Epochs vs Loss curve for the network:

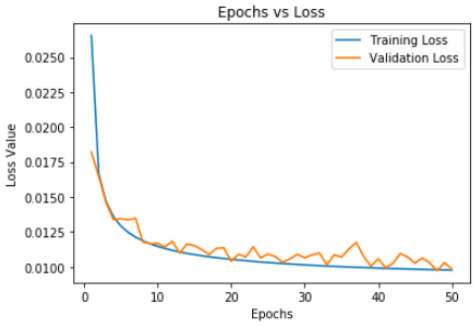

Figure 13: Epochs vs Loss curve for Encoder with 3 Hidden layer and main layer having 32 Nodes

Shown below is the Epochs vs Average reconstruction error for the network:

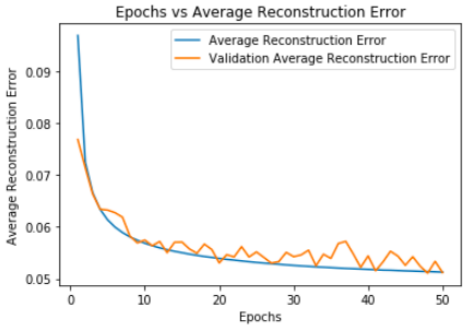

Figure 14: Epochs vs Average Reconstruction Error for Encoder with 3 Hidden layer and main layer having 32 Nodes

This encoder was then tested on the seperate test and these were the resulting metrics:

- Loss: 0.009809
- Average Reconstruction Error: 0.05114

Shown below is an example image and it's reconstruction from the autoencoder:

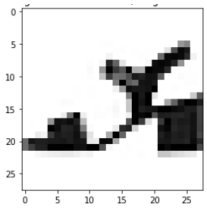

Figure 15: Sample Image from Dataset

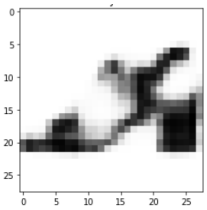

Figure 16: Reconstruction from Encoder with 3 Hidden layer and main layer having 32 Nodes

E. Training Encoder with 3 Hidden layers, main layer has 64 Nodes

Shown below is the summary of the Encoder created with 3 Hidden Layers and the main one having 64 Nodes

Layer (type)	Output Shape	Param #
dense_46 (Dense)	(None, 128)	100480
activation_43 (Activation)	(None, 128)	0
dense_47 (Dense)	(None, 64)	8256
activation_44 (Activation)	(None, 64)	0
dense_48 (Dense)	(None, 128)	8320
activation_45 (Activation)	(None, 128)	0
dense_49 (Dense)	(None, 784)	101136
activation_46 (Activation)	(None, 784)	0

Total params: 218,192
Trainable params: 218,192
Non-trainable params: 0

Figure 17: Summary of Encoder with 3 Hidden layer and main layer having 64 Nodes

This encoder was trained up 50 epochs and in the end these were it's final metrics:

- Loss: 0.0083
- Average Reconstruction Error: 0.0481
- Validation Loss: 0.0083
- Validation Average Reconstruction Error: 0.0485

Shown below is the Epochs vs Loss curve for the network:

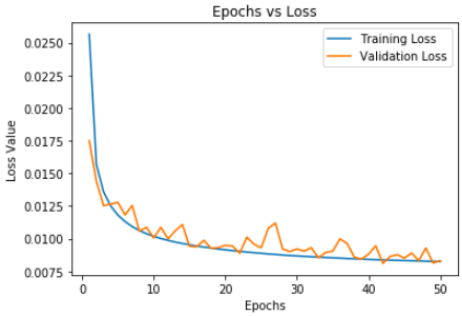

Figure 18: Epochs vs Loss curve for Encoder with 3 Hidden layer and main layer having 64 Nodes

Shown below is the Epochs vs Average reconstruction error for the network:

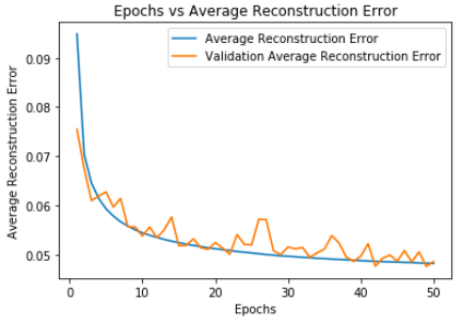

Figure 19: Epochs vs Average Reconstruction Error for Encoder with 3 Hidden layer and main layer having 64 Nodes

This encoder was then tested on the seperate test and these were the resulting metrics:

- Loss: 0.00829
- Average Reconstruction Error: 0.0484

Shown below is an example image and it's reconstruction from the autoencoder:

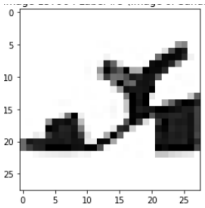

Figure 20: Sample Image from Dataset

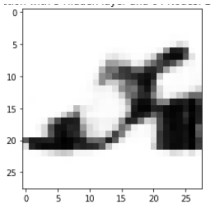

Figure 21: Reconstruction from Encoder with 3 Hidden layer and main layer having 64 Nodes

F. Comments

The multiple layer Encoders outperform the single layer encoders. The network with 3 hidden layers and 64 nodes in the center is the best performing one out of all. The order of worst performing to best is from A, B, C to D.

II. Denoising Autoencoders

Here I use the same data as in Task 1 for training and testing. However, before applying as input I corrupt the images with 0 mean additive gaussian noise with some standard deviation value. I change the noise standard deviation level to 0.1, 0.2 and 0.3. While showing a few reconstructed examples along with the corresponding noisy images. Then I comment on

the quality and report the root mean square error between the reconstructed testing images and clean testing images for the three noise levels and the two architectures with different dimensionality of the representative codes.

A. Preprocessing

Noise was added using numpys random.normal function which let use add noise with different standard deviation to each image. Shown below is the original image along with the image corrupted with noise of standard deviation 0.1, 0.2 and 0.3.

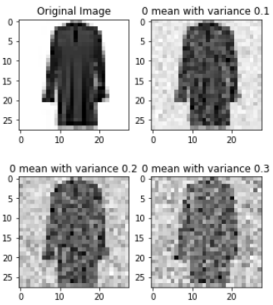

Figure 22: Original image and image corrupted with different noise variances

B. 32 node convolutional autoencoder

A convolutional autoencoder was implemented and it's architecture can be seen below:

Layer (type)	Output Shape	Param #
conv2d_24 (Conv2D)	(None, 28, 28, 32)	320
activation_81 (Activation)	(None, 28, 28, 32)	0
max_pooling2d_15 (MaxPooling	(None, 14, 14, 32)	0
conv2d_25 (Conv2D)	(None, 14, 14, 32)	9248
activation_82 (Activation)	(None, 14, 14, 32)	0
max_pooling2d_16 (MaxPooling	(None, 7, 7, 32)	0
conv2d_transpose_17 (Conv2DT	(None, 14, 14, 32)	9248
activation_83 (Activation)	(None, 14, 14, 32)	0
conv2d_transpose_18 (Conv2DT	(None, 28, 28, 1)	289
activation_84 (Activation)	(None, 28, 28, 1)	0

Total params: 19,105
Trainable params: 19,105
Non-trainable params: 0

Figure 23: 32 node convolutional autoencoder architecture

1) Training on 0.1 variance and 0 mean noise : This denoising autoencoder was trained with 65 epochs on the noisy data with the clean data as output label. Shown below are its metrics after training:

- Loss: 0.0036
- Average Reconstruction Error: 0.0315
- Validation Loss: 0.0037
- Validation Average Reconstruction Error: 0.0318

Shown below is the Epochs vs Loss curve for the denoising autoencoder:

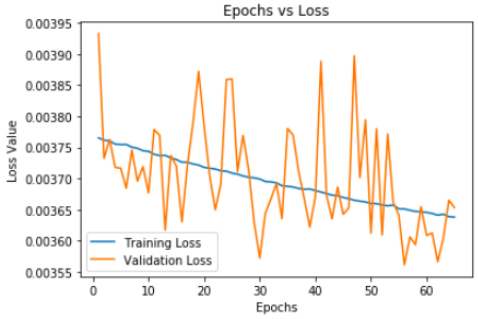

Figure 24: Epochs vs Loss curve for 32 node convolutional autoencoder

Shown below is the Epochs vs Average reconstruction error for the denoising autoencoder:

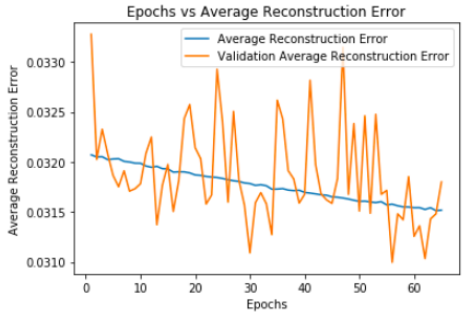

Figure 25: Epochs vs Average Reconstruction Error for 32 node convolutional autoencoder

This autoencoder was then tested on the seperate test and these were the resulting metrics:

- Loss: 0.003687
- Average Reconstruction Error: 0.03192

Shown below is an example image, noisy image and it's reconstruction from the autoencoder:

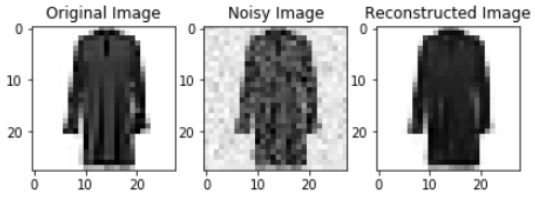

Figure 26: Image, noisy image (0.1) and it's reconstruction from the denoising autoencoder with 32 nodes

2) Training on 0.2 variance and 0 mean noise : This denoising autoencoder was trained with 65 epochs on the noisy data with the clean data as output label. Shown below are its metrics after training:

- Loss: 0.0060
- Average Reconstruction Error: 0.0413
- Validation Loss: 0.0060
- Validation Average Reconstruction Error: 0.0413

Shown below is the Epochs vs Loss curve for the denoising autoencoder:

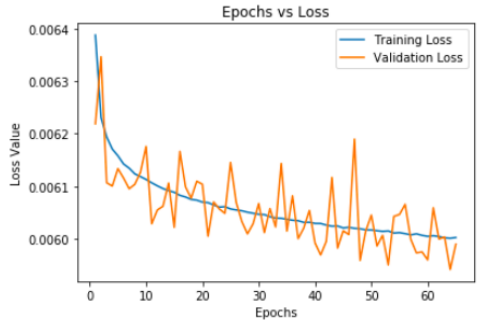

Figure 27: Epochs vs Loss curve for 32 node convolutional autoencoder

Shown below is the Epochs vs Average reconstruction error for the denoising autoencoder:

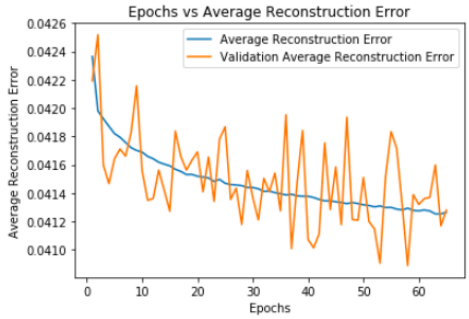

Figure 28: Epochs vs Average Reconstruction Error for 32 node convolutional autoencoder

This autoencoder was then tested on the seperate test and these were the resulting metrics:

- Loss: 0.006031
- Average Reconstruction Error: 0.04141

Shown below is an example image, noisy image and it's reconstruction from the autoencoder:

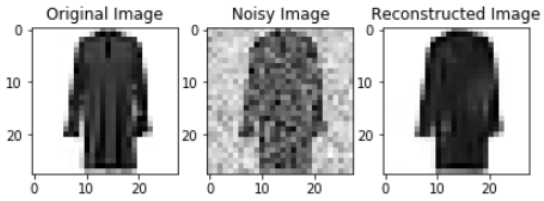

Figure 29: Image, noisy image (0.2) and it's reconstruction from the denoising autoencoder with 32 nodes

3) Training on 0.3 variance and 0 mean noise : This denoising autoencoder was trained with 65 epochs on the noisy data with the clean data as output label. Shown below are its metrics after training:

- Loss: 0.0089
- Average Reconstruction Error: 0.0511
- Validation Loss: 0.0091
- Validation Average Reconstruction Error: 0.0516

Shown below is the Epochs vs Loss curve for the denoising autoencoder:

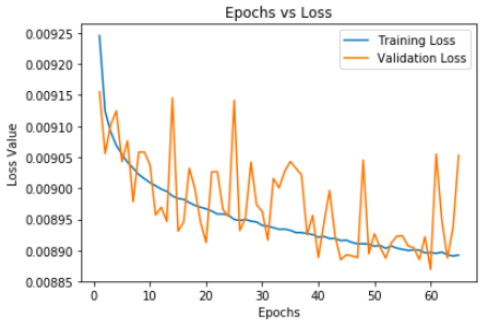

Figure 30: Epochs vs Loss curve for 32 node convolutional autoencoder

Shown below is the Epochs vs Average reconstruction error for the denoising autoencoder:

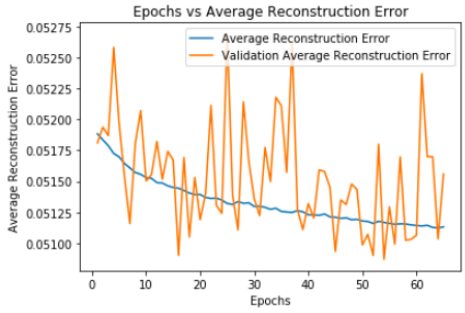

Figure 31: Epochs vs Average Reconstruction Error for 32 node convolutional autoencoder

This autoencoder was then tested on the seperate test and these were the resulting metrics:

- Loss: 0.009111
- Average Reconstruction Error: 0.05177

Shown below is an example image, noisy image and it's reconstruction from the autoencoder:

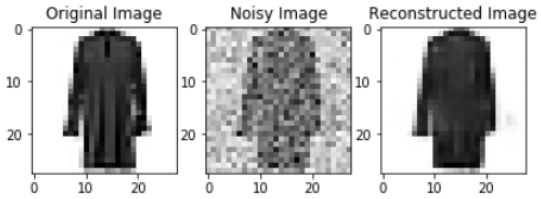

Figure 32: Image, noisy image (0.3) and it's reconstruction from the denoising autoencoder with 32 nodes

C. 64 node convolutional autoencoder

A convolutional autoencoder was implemented and it's architecture can be seen below:

Layer (type)	Output Shape	Param #
conv2d_26 (Conv2D)	(None, 28, 28, 64)	640
activation_85 (Activation)	(None, 28, 28, 64)	0
max_pooling2d_17 (MaxPooling	(None, 14, 14, 64)	0
conv2d_27 (Conv2D)	(None, 14, 14, 64)	36928
activation_86 (Activation)	(None, 14, 14, 64)	0
max_pooling2d_18 (MaxPooling	(None, 7, 7, 64)	0
conv2d_transpose_19 (Conv2DT	(None, 14, 14, 64)	36928
activation_87 (Activation)	(None, 14, 14, 64)	0
conv2d_transpose_20 (Conv2DT	(None, 28, 28, 1)	577
activation_88 (Activation)	(None, 28, 28, 1)	0

Total params: 75,073
Trainable params: 75,073
Non-trainable params: 0

Figure 33: 64 node convolutional autoencoder architecture

1) Training on 0.1 variance and 0 mean noise : This denoising autoencoder was trained with 65 epochs on the noisy data with the clean data as output label. Shown below are its metrics after training:

- Loss: 0.0030
- Average Reconstruction Error: 0.0288
- Validation Loss: 0.0031
- Validation Average Reconstruction Error: 0.0293

Shown below is the Epochs vs Loss curve for the denoising autoencoder:

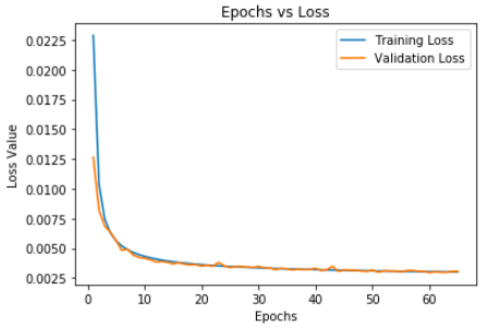

Figure 34: Epochs vs Loss curve for 64 node convolutional autoencoder

Shown below is the Epochs vs Average reconstruction error for the denoising autoencoder:

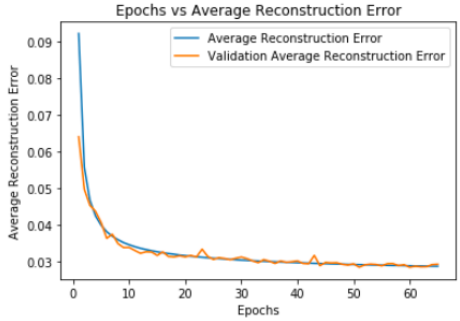

Figure 35: Epochs vs Average Reconstruction Error for 64 node convolutional autoencoder

This autoencoder was then tested on the seperate test and these were the resulting metrics:

- Loss: 0.0030992
- Average Reconstruction Error: 0.029463

Shown below is an example image, noisy image and it's reconstruction from the autoencoder:

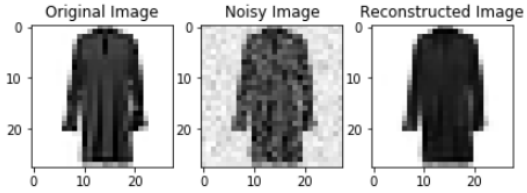

Figure 36: Image, noisy image (0.1) and it's reconstruction from the denoising autoencoder with 64 nodes

2) Training on 0.2 variance and 0 mean noise : This denoising autoencoder was trained with 65 epochs on the noisy data with the clean data as output label. Shown below are its metrics after training:

- Loss: 0.0053
- Average Reconstruction Error: 0.0384
- Validation Loss: 0.0054
- Validation Average Reconstruction Error: 0.0392

Shown below is the Epochs vs Loss curve for the denoising autoencoder:

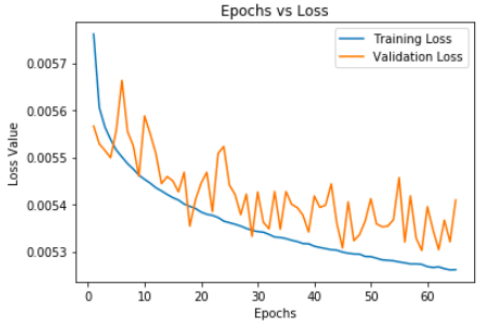

Figure 37: Epochs vs Loss curve for 64 node convolutional autoencoder

Shown below is the Epochs vs Average reconstruction error for the denoising autoencoder:

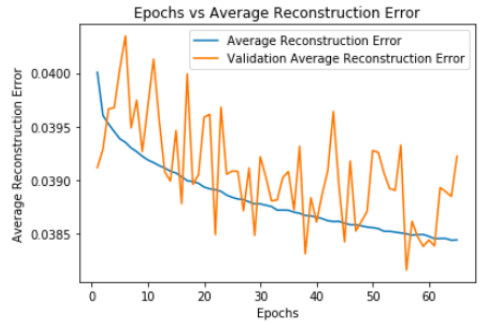

Figure 38: Epochs vs Average Reconstruction Error for 64 node convolutional autoencoder

This autoencoder was then tested on the seperate test and these were the resulting metrics:

- Loss: 0.005457
- Average Reconstruction Error: 0.03937

Shown below is an example image, noisy image and it's reconstruction from the autoencoder:

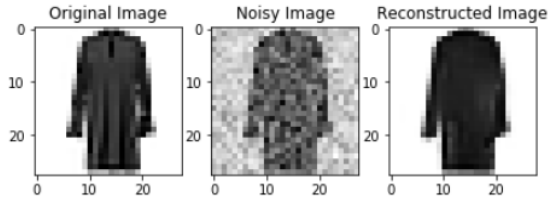

Figure 39: Image, noisy image (0.2) and it's reconstruction from the denoising autoencoder with 64 nodes

3) Training on 0.3 variance and 0 mean noise : This denoising autoencoder was trained with 65 epochs on the noisy data with the clean data as output label. Shown below are its metrics after training:

- Loss: 0.0080
- Average Reconstruction Error: 0.0481
- Validation Loss: 0.0082
- Validation Average Reconstruction Error: 0.0490

Shown below is the Epochs vs Loss curve for the denoising autoencoder:

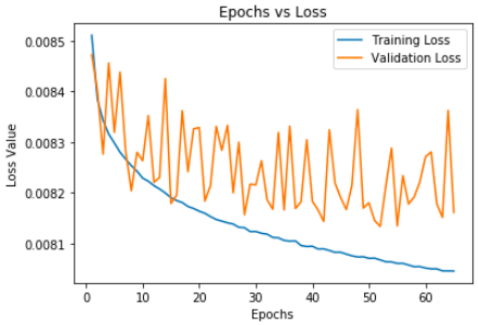

Figure 40: Epochs vs Loss curve for 64 node convolutional autoencoder

Shown below is the Epochs vs Average reconstruction error for the denoising autoencoder:

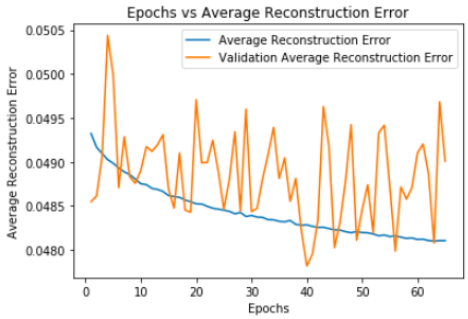

Figure 41: Epochs vs Average Reconstruction Error for 64 node convolutional autoencoder

This autoencoder was then tested on the seperate test and these were the resulting metrics:

- Loss: 0.008227
- Average Reconstruction Error: 0.04924

Shown below is an example image, noisy image and it's reconstruction from the autoencoder:

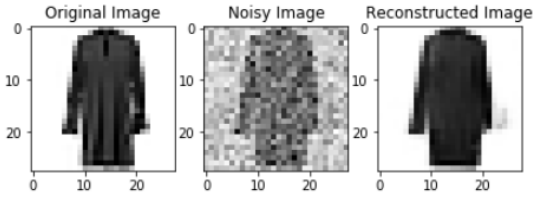

Figure 42: Image, noisy image (0.3) and it's reconstruction from the denoising autoencoder with 64 nodes

D. Comments

In all cases the 64 node denoising autoencoder outperforms the 32 node one. The individual performances are visible in

each of the subsections above. While the quality is almost identical the autoencoder does tend to loose certain elements of shading across the data and in high noise image there are artifacts produced.

III. LSTM NETWORKS

Data is given that is sequences of accelerometer and gyroscope data from an IMU attached to the shoe of three subjects. The subjects are asked to walk/run on a treadmill at 12 different speeds.

Using the data for subject 1 as training and the data for subject 2 as validation and building my own LSTM network. I picked the structure that gives you the best performance with the hardware available from Google Collab. Then I combineed the data for subject 1 and subject 2 and used it to train a network that was tested on data from subject 4. Also reported the average root mean squated error for speed estimation, both on the training data and testing data.

A. Preprocessing

Data was uploaded to Google Drive and then imported from there. This was done so that no data was lost due to incomplete upload or lost through a runtime disconnection on Google Collab. The size of the data was as follows:

- Size of Subject data is (12000, 600, 6)
- Size of Truth data is (12000, 1)

B. Training with Subject 1 and Validating with Subject 2

A LSTM network was written in Keras and it's architecture is shown below, mean squared error was the loss function.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 600, 200)          165600
_________________________________________________________________
lstm_2 (LSTM)                (None, 600, 200)          320800
_________________________________________________________________
lstm_3 (LSTM)                (None, 200)               320800
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 201
=================================================================
Total params: 807,401
Trainable params: 807,401
Non-trainable params: 0
_________________________________________________________________
```

Figure 43: Architecture of LSTM network

Training of the network was only done on 3 epochs with a batch size of 148. This is because Google collab would crash due to RAM limitations on higher epochs. The final learning metrics are:

- Loss: 0.0918
- Accuracy: 0.4565
- Validation Loss: 1.7333
- Validation Accuracy: 0.1515

Shown below is the loss and accuracy vs those 3 epochs.

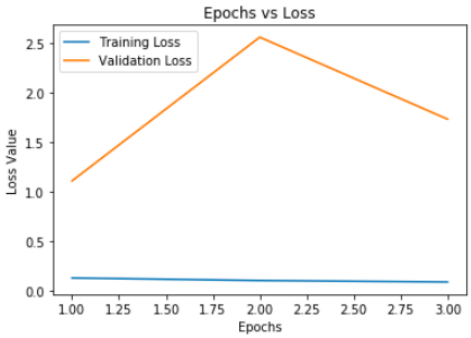

Figure 44: Epochs vs Loss

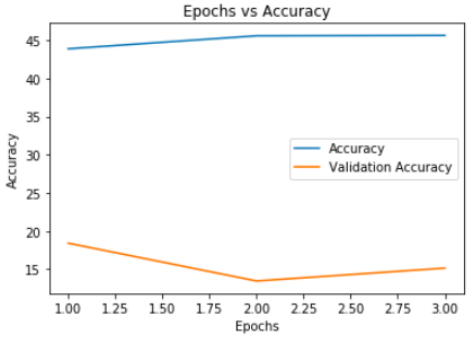

Figure 45: Epochs vs Accuracy

IV. TRAINING WITH SUBJECT 1 AND 2, TESTING ON SUBJECT 4

The data of subject 1 and 2 was first concatenated and therefore the data shapes were as follows:

- Size of Subject 1 and 2 mixed data is (24000, 600, 6)
- Size of Subject 4 data is (12000, 600, 6)
- Size of Subject 1 and 2 mixed truth data is (12000, 1)
- Size of Subject 4 truth data is (12000, 1)

The mixed data was then input the same network as in figure 43 and it was trained once again over 3 epochs due to limitations on Google collab. The final learning metrics are:

- Loss: 0.1852
- Accuracy: 0.4003

Shown below is the loss and accuracy vs those 3 epochs.

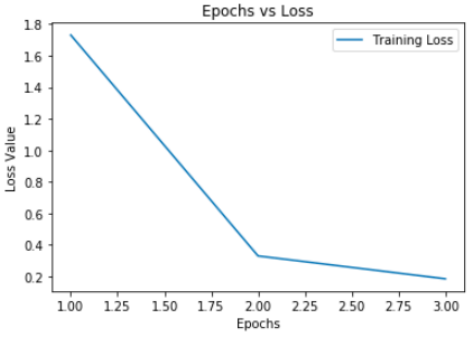

Figure 46: Epochs vs Loss

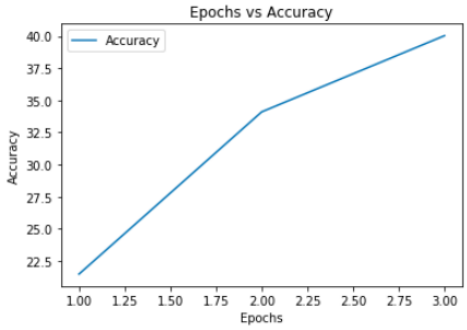

Figure 47: Epochs vs Accuracy

The data was then tested using subject 4s data and the results were as follows:

- Loss: 2.588
- Accuracy: 0.11099

V. Variational Autoencoders

Using regular autoencoders one of the main limitations is that the the latent space it converts inputs to is not always continuous. This means we cannot easily interpolate between different points of the latent space. This means if we randomly sample from the latent space we do not always get some meaningful variation of an input image and therefore using an decoder means we cannot truly generate new or interpolated data by sampling. Variational autoencoders solve this and allow for generative modelling because the latent spaces they map to are always continous and can easily be sampled. To do this the encoder outputs a mean and standard deviation for a gaussian distributed variable, so the decoder now does not get a fixed value but instead it samples a gaussian distributed region. The decoder then samples this region for points when decoding and this teaches it that a specific point is not referring to a sample of a class but instead the entire region corresponds to a class. Through training the decoder learns the mean and standard deviations but it may place classes far apart in the latent space even though they may be close to each other. To prevent it from doing this the loss function also includes the KL divergence which is a measure of how much two distributions diverge from each other so by forcing the network to minimize the loss function we can also force it to minimize the KL divergence and keep the different class distributions close to each other. The loss function also contains a reconstruction loss that ensures the the sampled data is reconstructed to the original input.

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)] + \mathbb{KL}(q_\theta(z \mid x_i) \mid$$

Figure 48: Loss Function of Variational Autoencoders