**Neural Networks and Deep Learning – ICP 9**

**GitHub Link:** [**https://github.com/rumanathaskeen22/Neural-Networks-and-Deep-Learning-ICP-9**](https://github.com/rumanathaskeen22/Neural-Networks-and-Deep-Learning-ICP-9)

**Video Link:** [**https://drive.google.com/file/d/175ZofT4ZSkS47ou83rZySbii9C0\_WKGJ/view?usp=sharing**](https://drive.google.com/file/d/175ZofT4ZSkS47ou83rZySbii9C0_WKGJ/view?usp=sharing)

**Name: Lnu Rumana Thaskeen**

**Student ID: 700742859**

**CRN: 23216**

**NN&DeepLearning\_Lesson: Autoencoders**

This code defines and trains an autoencoder neural network on the Fashion-MNIST dataset. An autoencoder is a type of neural network that learns to reconstruct its input data from a lower-dimensional encoding, effectively learning a compressed representation of the data.

The autoencoder network is defined using the Keras library, with two fully connected (Dense) layers. The input layer has 784 units, and the encoded layer has 32 units.

The autoencoder is trained on the Fashion-MNIST dataset for 5 epochs, using the ADAM optimizer and binary cross-entropy loss function. The input data is normalized to the range [0,1].

**Text

Description automatically generated**

The input data (x\_train) is passed twice to the fit method as both the input and target data, since the goal of the autoencoder is to reconstruct the input data as accurately as possible.

**Text

Description automatically generated**

I have now written the code that predicts the reconstructed images for the test set using the trained autoencoder model. The predict method of the model object is called with the test set (x\_test) as input data, and the reconstructed images are returned and stored in the variable 'prediction' and then plotting the input image and the reconstructed image as shown below.

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated

1. **Add one more hidden layer to autoencoder**

I have now added these lines to add additional hidden layers to the encoding and decoding portions of the autoencoder neural network.

The first line adds a hidden layer with 512 units and ReLU activation function to the input layer (input\_img) of the autoencoder. This increases the complexity of the network and can potentially improve the quality of the compressed representation learned by the encoder.

The second line defines the encoded representation of the input by adding another hidden layer with 32 units and ReLU activation function to the output of the previous hidden layer.

**Text

Description automatically generated**

Now upon training and predicting the model as shown below:

Graphical user interface, text

Description automatically generated

1. **Do the prediction on the test data and then visualize one of the reconstructed version of that test data. Also, visualize the same test data before reconstruction using Matplotlib**

The image is reshaped to its original dimensions of 28x28 using the reshape method, since the input to the autoencoder was flattened to a vector of length 784.

The imshow method of the pyplot module is used to display the image, and the show method is called to show the image in a separate window.

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

1. **Repeat the question 2 on the denoisening autoencoder**

I have now introduced random Gaussian noise to the input images by adding a scaled noise array to the original images.

The 'noise\_factor' variable controls the amount of noise added to the images. In this case, a noise\_factor of 0.5 is used, which means that the noise array will have an average magnitude of half of the standard deviation of the original image data.

The numpy.random.normal() function generates a random array with the same shape as the input array (x\_train or x\_test). The resulting noisy images, 'x\_train\_noisy' and 'x\_test\_noisy', are used as inputs to the autoencoder to train it to denoise the images by reconstructing the original images from the noisy inputs.

**Text

Description automatically generated**

displaying a single image from the original training set before and after noise is added, using the Matplotlib library.The imshow method of the pyplot module is used to display the image, and the show method is called to show the image in a separate window.

**Chart, histogram

Description automatically generated**

**Graphical user interface

Description automatically generated**

Predicting on the test data and This code displays a single noisy image from the test set and then showing the reconstructed image using the Matplotlib library as shown below.

**Graphical user interface, application

Description automatically generated**

**Graphical user interface

Description automatically generated**

**4. plot loss and accuracy using the history object**

I have now written this code that creates a plot of the training history of the autoencoder model, using the Matplotlib library.

The history object returned by the call to the 'fit' method of the autoencoder model is assumed to have recorded the values of the 'accuracy' and 'loss' metrics for each epoch of training. The 'history.history' dictionary is used to extract these values for plotting.

The 'plot' method of the pyplot module is used to create a line plot with epoch number on the x-axis and the accuracy and loss values on the y-axis as shown below.

**Text

Description automatically generated**

**Graphical user interface

Description automatically generated**