# Neural Networks and Deep Learning Assignment - 4 :

# Id : 700742325

# Name: Jyothi Sai Rajesh

# Kunapareddy

**VideoLink:**

**<https://drive.google.com/file/d/1IlKcM8nxDe2c_fCMPE6i5LS9Yiea4Tn-/view?usp=sharing>**

**Source Code Link:**

**<https://github.com/Rk-oo7/NNDL_Assignment_4.git>**

## Add one more hidden layer to AutoEncoder:

**A screenshot of a computer

Description automatically generated**

1. **from keras.layers import Input, Dense**: This line imports the **Input** and **Dense** classes from Keras. **Input** is used to instantiate a Keras tensor, and **Dense** is used to instantiate a densely- connected layer.
2. **from keras.models import Model**: This line imports the **Model** class from Keras, which groups layers into an object with training and inference features.
3. **encoding\_dim = 32**: This line sets the dimensionality of the encoded data to 32. This value is the size of the latent space that the autoencoder will map the input data into.
4. **input\_img = Input(shape=(784,))**: This line creates a placeholder for the input images. The shape is set to (784,), meaning that the images are 784-dimensional (28x28 pixel images that have been flattened).

A screen shot of a computer code

Description automatically generated

1. **encoded = Dense(encoding\_dim, activation='relu')(input\_img)**: This line creates the encoding layer of the autoencoder. It's a fully-connected (**Dense**) layer that uses the ReLU activation function.
2. **decoded = Dense(784, activation='sigmoid')(encoded)**: This line creates the decoding layer of the autoencoder. It's also a **Dense** layer, and it uses the sigmoid activation function.
3. **autoencoder = Model(input\_img, decoded)**: This line constructs the autoencoder model, which will map an input image to its reconstructed version.
4. **autoencoder.compile(optimizer='adadelta', loss='binary\_crossentropy')**: This line compiles the model, specifying the optimizer and loss function to be used during training.

A computer screen shot of a program code

Description automatically generated

1. **from keras.datasets import mnist, fashion\_mnist**: This line imports the MNIST and Fashion MNIST datasets from Keras. In this script, only the Fashion MNIST dataset is used.
2. **import numpy as np**: This line imports the numpy library, which is used for numerical operations.
3. **(x\_train, y\_train), (x\_test, y\_test) = fashion\_mnist.load\_data()**: This line loads the Fashion MNIST data. The data is split into training and test sets.
4. **x\_train = x\_train.astype('float32') / 255.** and **x\_test = x\_test.astype('float32') / 255.**: These lines convert the data to float32 and normalize the pixel values to the range [0, 1] by dividing by 255 (the maximum value a pixel can have).
5. **x\_train = x\_train.reshape((len(x\_train), np.prod(x\_train.shape[1:])))** and **x\_test = x\_test.reshape((len(x\_test), np.prod(x\_test.shape[1:])))**: These lines reshape the data from 2D images (28x28 pixels) to 1D arrays (784 elements).
6. **autoencoder.fit(x\_train, x\_train, epochs=5, batch\_size=256, shuffle=True, validation\_data=(x\_test, x\_test))**: This line trains the autoencoder. The autoencoder is trained to reconstruct the input data, hence the target data is also the input data (**x\_train, x\_train**). The model is trained for 5 epochs, with a batch size of 256. The data is shuffled after each epoch to ensure the model does not learn any unintended patterns from the order of the data. The test set is used as the validation data.

Output:

A screenshot of a computer program

Description automatically generated

## 2. Doing the prediction on the test data and then visualise one of the reconstructed version of the test data.

A computer screen shot of a program code

Description automatically generated

1. Importing necessary libraries: Necessary libraries for building and training the autoencoder model, loading and preprocessing the dataset, and plotting the results are imported.
2. Define the encoder dimension: The encoding\_dim variable is defined, which specifies the number of neurons in the encoded, most compressed layer of the autoencoder.
3. Define the input placeholder: The input\_img variable serves as a placeholder for the 784- dimensional input data.

A computer screen with text

Description automatically generated

1. Define the first hidden layer: The first hidden layer of the encoder part of the autoencoder is defined with 256 neurons and a ReLU activation function.
2. Define the second hidden layer: The second (encoded) layer of the autoencoder is defined with the number of neurons specified by the encoding\_dim variable and a ReLU activation function.
3. Define the first hidden layer of the decoder: The first hidden layer of the decoder part of the autoencoder is defined, mirroring the encoder part.
4. Define the output layer: The output layer of the autoencoder is defined with 784 neurons (to match the input dimension) and a sigmoid activation function to reconstruct the input.
5. Define the autoencoder model: The autoencoder model is built using the previously defined input and output.

A computer screen shot of a program code

Description automatically generated

1. Compile the model: The model is compiled with the Adadelta optimizer, binary cross-entropy loss function and accuracy metric.
2. Load the fashion MNIST dataset: The Fashion MNIST dataset is loaded for training and testing.
3. Normalize the data and flatten the images: The loaded images are normalized by dividing by 255 and reshaped to be flat arrays.
4. Train the autoencoder: The autoencoder model is trained on the training dataset for a specified number of epochs.
5. Make predictions on the test data: The trained model is used to make predictions on the test data.

A computer screen with text and images

Description automatically generated

1. Visualize one of the reconstructed images: A few of the original and reconstructed test images are displayed for comparison.

A screen shot of a computer program

Description automatically generated

Plot the loss and accuracy over time: Plots showing how the model's loss and accuracy evolved over training epochs are displayed.

Output:

A screenshot of a computer screen

Description automatically generated

A graph with a line and numbers

Description automatically generated

A graph with numbers and lines

Description automatically generated

## Repeat the question 2 on the denoising autoencoder

A screenshot of a computer

Description automatically generated

1. Import the required libraries and modules from Keras.
2. Set the size of the encoded representation to 32, implying a compression factor of approximately 24.5 assuming the original input size is 784.
3. Define the input placeholder with shape 784.

A screen shot of a computer code

Description automatically generated

1. Build the encoder part of the autoencoder, which takes the input image and compresses it into a lower dimensional space.
2. Build the decoder part of the autoencoder, which takes the encoded representation and attempts to recreate the original image.
3. Define the autoencoder model which maps an input to its reconstructed output.
4. Compile the autoencoder model specifying the optimizer and loss function to be used.

A computer screen shot of a program code

Description automatically generated

1. Import the Fashion MNIST dataset from Keras and load the training and testing data, ignoring the labels as they are not required for autoencoders.
2. Normalize the training and testing data to be between 0 and 1 by dividing by 255.
3. Reshape the training and testing data to be 2D (from 28x28 to 784).
4. Introduce noise to the training and testing data by adding Gaussian noise to the images.
5. Train the autoencoder on the noisy training images while validating the performance of the autoencoder on the noisy test images. The autoencoder should be able to learn the task of denoising the images.

**Output**:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer program

Description automatically generated

1. The first few lines are for importing the necessary libraries: Keras for building the neural network, numpy for handling numerical data, and matplotlib for plotting results.
2. The "encoding\_dim" is set, which defines the size of the encoded representations of the input data.
3. The "input\_img" placeholder is created, which will hold the input images that are 784 pixels in size.

A computer screen shot of a program code

Description automatically generated

1. An "encoded" layer is defined that takes the "input\_img" and reduces its dimensionality (encodes it) using a "Dense" layer with the "relu" activation function.
2. A "decoded" layer is defined that takes the "encoded" representation and tries to reconstruct the original image (decodes it) using a "Dense" layer with the "sigmoid" activation function.
3. The "autoencoder" model is created using the "input\_img" and "decoded" layers.
4. The model is compiled using the 'adadelta' optimizer, 'binary\_crossentropy' loss function, and tracking 'accuracy' as an additional metric.

A computer screen shot of a program code

Description automatically generated

1. The Fashion MNIST dataset is loaded for training and testing the model.
2. The training and testing data are normalized to the range [0,1] and reshaped to a 1D array.
3. Noise is added to the test data to create "noisy" images.
4. The autoencoder model is trained using the noisy training images as input and the original training images as the target.

A screen shot of a computer program

Description automatically generated

1. The trained autoencoder model is used to reconstruct the test images from the noisy test data.
2. The first plot visualizes a selection of the noisy test

A screen shot of a computer code

Description automatically generated

A collage of squares

Description automatically generated

A graph with numbers and lines

Description automatically generated

A graph with blue and orange lines

Description automatically generated