| **Ex No: 6**  **Date: 11-09-24** | **Linear (simple) Autoencoder Implementation for Dimensionality Reduction on MNIST Dataset** |
| --- | --- |

**Objective:**

To build and train an autoencoder model to reduce the dimensionality of the MNIST dataset and visualize both compressed representations and reconstructed outputs.

**Descriptions:**

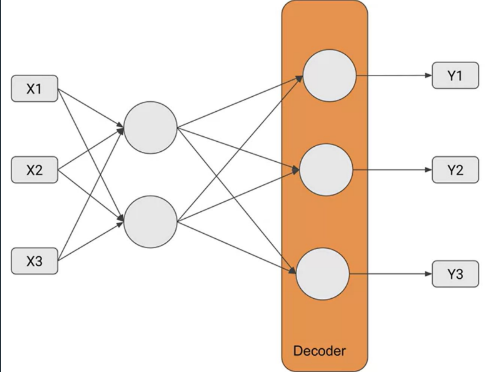
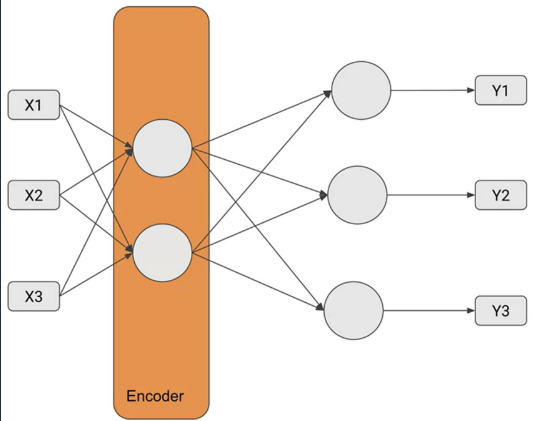
Dimensionality reduction is essential for simplifying data without losing significant information. An autoencoder, a type of neural network, is used for this task by learning an efficient representation of the data.

An autoencoder consists of:

* **Encoder:** Compresses the input data into a smaller, lower-dimensional latent space.
* **Decoder:** Reconstructs the original data from the compressed representation.

The autoencoder in this lab is designed to minimize reconstruction error by adjusting its weights and biases during training.

**Model:**



**Code explanation:**

# START YOUR CODE HERE

latent\_dim = 32

encoder = tf.keras.layers.Dense(units=latent\_dim, activation='relu')

decoder = tf.keras.layers.Dense(units=784, activation='sigmoid')

# Obtain the output of the encoder by calling it with the inputs

encoder\_output = encoder(inputs)

decoder\_output = decoder(encoder\_output)

# END YOUR CODE HERE

encoder=tf.keras.layers.Dense(units=latent\_dim,activation='relu')

* **Purpose:** Creates a dense (fully connected) layer for the encoder part of the autoencoder.
* **Details:**
  + units=latent\_dim: Specifies the number of neurons in this layer, which is equal to 32 (as defined earlier).
  + activation='relu': Uses the ReLU (Rectified Linear Unit) activation function, which introduces non-linearity to the model and helps it learn complex patterns. The ReLU function outputs zero for any negative input and outputs the input value if it is positive.

decoder=tf.keras.layers.Dense(units=784, activation='sigmoid')

* **Purpose:** Creates a dense (fully connected) layer for the decoder part of the autoencoder.
* **Details:**
  + units=784: Specifies the number of neurons in this layer, which is 784 (the original number of pixels in a flattened 28x28 MNIST image).
  + activation='sigmoid': Uses the Sigmoid activation function, which outputs values between 0 and 1. This is suitable for reconstructing normalized image pixel values, which are typically scaled to be in this range.

### Model:

* **Dataset:** MNIST (handwritten digit images)
* **Model Structure:**
  + **Encoder:** Sequential layers that reduce the input dimensions.
  + **Decoder:** Sequential layers that aim to reconstruct the original input from the compressed representation.
* **Loss Function:** Mean Squared Error (MSE)
* **Optimizer:** Adam

### Steps:

1. **Data Preparation:**
   * Load and normalize the MNIST dataset.
2. **Model Definition:**
   * Define the encoder and decoder models using TensorFlow/Keras.
3. **Training:**
   * Compile the model with the MSE loss function and Adam optimizer.
   * Train the model on the MNIST dataset.
4. **Visualization:**
   * Visualize the compressed representations and reconstructed outputs.

### Results:

* The model successfully learned to compress and reconstruct the MNIST images.
* The visualization of the latent space showed a compact, lower-dimensional representation of the input images.

**GitHubLink:** [**https://github.com/chandanab1/Deep\_Learning**](https://github.com/chandanab1/Deep_Learning)