|  |  |
| --- | --- |
| **Ex No: 6**  **Date: 04-09-2024** | **MNIST Autoencoder** |

**Objective:**

The objective of this lab exercise is to implement a simple autoencoder using the MNIST dataset. An autoencoder is a neural network architecture designed to learn a compressed representation of data (encoding) and then reconstruct the original input from this compressed form (decoding). The focus of this exercise is on creating a simple autoencoder with fully connected layers and evaluating its performance in encoding and reconstructing handwritten digit images.

**Description:**

The MNIST dataset consists of 28x28 grayscale images of handwritten digits (0-9). To prepare the data for the autoencoder, a preprocessing function is applied. The function normalizes the images by scaling the pixel values to the range [0, 1], and then flattens the images into vectors of size 784 (28x28). This is essential as the autoencoder model operates on one-dimensional vectors instead of two-dimensional images. Additionally, because an autoencoder learns to recreate the input, the label for each input image is the image itself (image, image).

**Model Architecture:**

The autoencoder model is divided into two components: the encoder and the decoder.

1. Encoder: The encoder is a neural network layer that compresses the input images into a lower-dimensional latent space. It is designed to capture the essential features of the input while reducing dimensionality. In this case, the encoder uses a fully connected dense layer with 32 neurons and the ReLU activation function. This bottleneck layer is responsible for learning the compressed representation of the data.
2. Decoder: The decoder aims to reconstruct the original input image from the encoded (compressed) representation. It consists of a fully connected dense layer with 784 neurons and the sigmoid activation function, which reconstructs the image back into a vector of the original size (784 pixels). The sigmoid function ensures that the pixel values are scaled between 0 and 1, which aligns with the normalized input data.

**Model Definition:**

The model is defined using the functional API in TensorFlow. The input layer has a shape of (784,) corresponding to the flattened images. The simple\_autoencoder function constructs the encoder and decoder, which are connected sequentially.

* **Encoder Model**: The encoder model is created separately for the purpose of visualizing the encoded output later.
* **Autoencoder Model**: This is the full model combining the encoder and decoder to reconstruct the images.

**Testing and Inference**

After building and training the autoencoder, the model is tested on a batch of images from the test dataset.

1. **Dataset Processing**: A single batch is taken from the test dataset for evaluation. The input images from this batch are stored in a list for later comparison with the encoded and decoded outputs.
2. **Random Sample Selection**: From the batch of input images, 10 random samples are selected for visualization. The indices of these random samples are generated using NumPy's random selection function.
3. **Encoding and Decoding**: The encoder model is used to generate the encoded representations of the selected input images. These encoded representations are in the latent space and provide compressed versions of the input images.
4. **Visualization**: The final step involves displaying the original images, the encoded (compressed) representations, and the reconstructed images side by side. This allows for a visual comparison between the original input and the model's reconstructions, as well as the corresponding compressed representations.

**Github Link:**

**https://github.com/Bhargava-Srinivasan-26/Deep\_learning\_elective/tree/main/Unit%202/Lab%201**