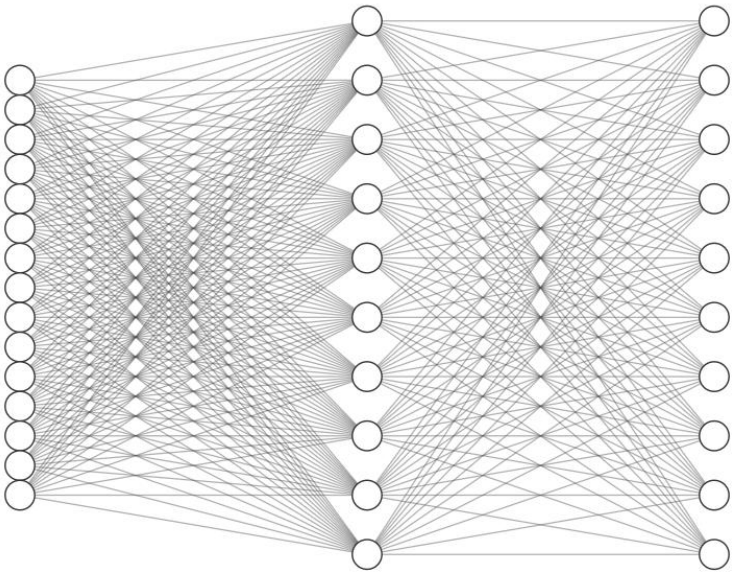
MNIST DIGIT RECOGNITION WITHOUT USING TENSORFLOW / OPENCV

MNIST dataset (from Kaggle) contains two datasets, training data and test data. Each entry has 784 values corresponding to the grayscale values at each pixel of the digit (28x28 pixels image). This projects aims to use these datasets to train a neural network to recognise digits but without using in-built machine learning libraries like TensorFlow or OpenCV. The entire neural network architecture is built just using numpy and pandas.

The neural network (more specifically Convolutional Neural Network CNN), consists of three layers, the input layer, hidden layer and output layer.
The input layer consists of 784 neurons, each corresponding to the grayscale values at each pixel as mentioned before. The hidden layer and the output layer consists of 10 neurons.
ReLU activation function is used at the hidden layer and softmax activation at output layer. By carefully assigning the shapes of the arrays, we can initialize the values of weights and biases at each layer.

The network architecture is shown below.



The mathematics followed is:
Forward propagation:

$$Z[1] = W[1] * X\_Training + B[1]$$
$$A[1] = ReLU\ (Z[1])$$
$$Z[2] = W[2] * A[1] + B[2]$$
$$A[2] = SOFTMAX\ (Z[2])$$

Backward propagation:

$$L[2] = A[2] - Y\_Training$$
$$dW[2] = (1 / m) * L[2] * A[1]^{T}$$
$$dB[2] = (1 / m) * \Sigma\ (L[2])$$
$$L[1] = W[2]^{T} * L[2] * A[2]'$$
$$dW[1] = (1 / m) * L[1] * A[0]^{T}$$
$$dB[1] = (1 / m) * \Sigma\ (L[1])$$

Updating parameters:

$$W[2] = W[2] - \alpha * dW[2]$$
$$B[2] = B[2] - \alpha * dB[2]$$
$$W[1] = W[1] - \alpha * dW[1]$$
$$B[1] = B[1] - \alpha * dB[1]$$

The model showed an accuracy of ~85% for the training dataset. When tested with the cross - validation set, it showed an accuracy of ~84%. This means that the neural network has generalized well from the training data.