**Name:** Pushkar Charkha  
**Roll No:** 381002  
**PRN:** 22310023

**Assignment No: -** 1  
**Title: -** Feed-Forward Neural Network on MNIST Dataset

**Problem Statement:**

Implementing Feedforward Neural Networks in Python using Keras and TensorFlow for handwritten digit classification (MNIST dataset).

**Objective:**

* To understand the structure of feedforward neural networks.
* To preprocess image data for training neural networks.
* To build a feedforward model using Keras and TensorFlow.
* To evaluate the trained model using accuracy and loss metrics.
* To visualize the performance of the model over epochs.

**S/W Packages and H/W apparatus used:**

* **Operating System:** Windows/Linux/MacOS
* **Kernel:** Python 3.x
* **Tools:** Jupyter Notebook, Anaconda, Google Colab
* **Hardware:** CPU with min 4GB RAM; GPU optional
* **Libraries:** TensorFlow, Keras, NumPy, Pandas, Matplotlib

**Theory:**

A **Feedforward Neural Network (FNN)** is an artificial neural network where the flow of data is unidirectional, from input to output. It consists of:

* **Input Layer** → accepts input features (MNIST images, 28x28 pixels).
* **Hidden Layers** → multiple dense layers that perform non-linear transformations using activation functions like ReLU.
* **Output Layer** → produces class probabilities (0–9 digits) using Softmax activation.

The network is trained using **backpropagation**, minimizing the **Sparse Categorical Crossentropy loss** with the **Stochastic Gradient Descent (SGD) optimizer**.

**Methodology:**

1. **Data Acquisition**: Loaded MNIST dataset from Keras.
2. **Preprocessing**: Normalized image pixel values to the range [0,1].
3. **Model Architecture**:
   * Input Layer (Flatten 28×28).
   * Dense Layer with 50 neurons, ReLU activation.
   * Dense Layer with 50 neurons, ReLU activation.
   * Output Layer with 10 neurons, Softmax activation.
4. **Compilation**: Optimizer → SGD, Loss → SparseCategoricalCrossentropy, Metric → Accuracy.
5. **Training**: Batch size 30, trained for 10 epochs, validated on test data.
6. **Evaluation**: Tested model accuracy on unseen test set.
7. **Visualization**: Training and validation accuracy/loss monitored across epochs.

**Results:**

* **Final Training Accuracy:** ~96.27%
* **Validation Accuracy:** ~96.01%
* **Test Accuracy:** ~95.43%
* **Test Loss:** ~0.1548

**Advantages:**

* High accuracy for digit recognition.
* Efficient training with SGD optimizer.
* Generalizes well on unseen test data.

**Limitations:**

* Requires large labeled datasets.
* Training time can be slow without GPU.
* Sensitive to hyperparameters like batch size and learning rate.

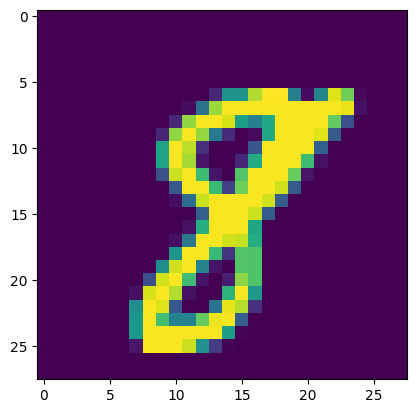
**Applications:**

* Handwritten digit recognition (MNIST).
* Character recognition in OCR.
* Image classification tasks.
* Pattern recognition problems.

**Working / Algorithm:**

1. Import libraries (NumPy, TensorFlow, Matplotlib).
2. Load MNIST dataset.
3. Normalize dataset (divide by 255).
4. Build Sequential model with input, hidden, and output layers.
5. Compile the model with optimizer, loss, and metrics.
6. Train the model with training data and validate with test data.
7. Evaluate the model on test dataset.
8. Plot accuracy/loss curves.

**Diagram:**



**Conclusion:**

The Feedforward Neural Network built using Keras and TensorFlow successfully classified handwritten digits from the MNIST dataset with ~95% accuracy. The model demonstrated good generalization, making it suitable for image classification tasks. With hyperparameter tuning and deeper architectures, even higher accuracy can be achieved.