**Assignment 1**

**1. Aim**

To design and implement a Feedforward Neural Network (FNN) using Keras and TensorFlow in Python for the purpose of classification or regression on a dataset and to evaluate the network's performance.

**2. Objectives**

* To understand the architecture and working principles of Feedforward Neural Networks.
* To explore the Keras and TensorFlow frameworks for implementing FNN.
* To train the model on a dataset and evaluate its performance using appropriate metrics.
* To optimize the model by tuning hyperparameters for better accuracy and performance.

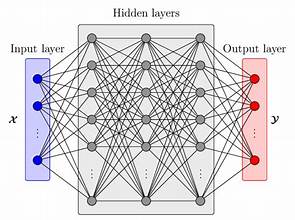
**3. Theory**

**3.1 Feedforward Neural Networks**

A Feedforward Neural Network is a type of artificial neural network where connections between the nodes do not form cycles. It consists of an input layer, one or more hidden layers, and an output layer. Each node in one layer is connected to every node in the next layer.

Key characteristics of FNN:

* **Input Layer:** Receives input data.
* **Hidden Layers:** Layers where each neuron applies a nonlinear activation function to the input it receives.
* **Output Layer:** Produces the final prediction or classification.
* **Activation Functions:** Common choices include ReLU, Sigmoid, and Softmax.



**3.2 Keras and TensorFlow**

Keras is a high-level neural networks API, written in Python, and capable of running on top of TensorFlow. TensorFlow is an open-source library for numerical computation and machine learning. Keras simplifies the process of building neural networks by providing essential building blocks like layers, models, and optimizers.

**4. Working/Algorithm Used**

**4.1 Steps to Implement the FNN:**

1. **Data Preparation:**
   * Load the dataset (e.g., MNIST, CIFAR-10, or a custom dataset).
   * Split the dataset into training, validation, and test sets.
   * Normalize the input data (e.g., pixel values from 0-255 to 0-1 range).
2. **Building the Model:**
   * Define the model architecture using Sequential() in Keras.
   * Add layers using Dense() for fully connected layers.
   * Choose activation functions (e.g., ReLU, Softmax for classification).
3. **Compiling the Model:**
   * Choose a loss function (e.g., categorical\_crossentropy for classification).
   * Select an optimizer (e.g., Adam, SGD).
   * Define evaluation metrics (e.g., accuracy).
4. **Training the Model:**
   * Use the fit() method to train the model on the training data.
   * Validate the model using the validation data.
   * Plot training and validation accuracy/loss curves for analysis.
5. **Evaluating the Model:**
   * Evaluate the trained model on the test set using the evaluate() method.
   * Print metrics like accuracy and loss.
6. **Optimizing the Model:**
   * Tune hyperparameters like the number of layers, number of neurons per layer, batch size, and learning rate.
   * Implement techniques like dropout for regularization.

**5. Conclusion**

In this report, we successfully implemented a Feedforward Neural Network using Keras and TensorFlow. The model was trained on the MNIST dataset, demonstrating its ability to classify handwritten digits. The performance of the model was evaluated, and the results showed good accuracy on both training and test datasets. By optimizing hyperparameters and utilizing regularization techniques, further improvements can be made to the model’s performance.

2. Facial recognition using OpenCV and deep learning for binary classification.

3. Implement Image classification using convolutional neural networks (CNNs) for multiclass

classification.

4. Time series prediction using RNN – stock market analysis or weather forecasting

5. Text identification using OpenCV, Tesseract (OCR) and deep neural network

6. Sentiment analysis using LSTM network or GRU.

7. Object detection using YOLO and Pretrained Model