**Assignment No: 1**

**Feedforward Neural Network**

Mrunal Talikoti 22311431

1. **Problem Statement**

Implement a Feedforward Neural Network (FNN) in Python using Keras and TensorFlow to predict outcomes based on input features.

1. **Objective**

* Understand the fundamental structure of feedforward neural networks.
* Learn how to pre-process and normalize data for neural network training.
* Implement a feedforward neural network using Keras and TensorFlow.
* Evaluate model performance using validation data.
* Visualize training and validation loss trends over epochs.

1. **Software and Hardware Requirements**

**Operating System:** Windows / Linux / MacOS  
**Programming Environment:** Python 3.x, Jupyter Notebook, Anaconda, or Google Colab  
**Hardware:** Minimum 4GB RAM (CPU); GPU optional for faster training

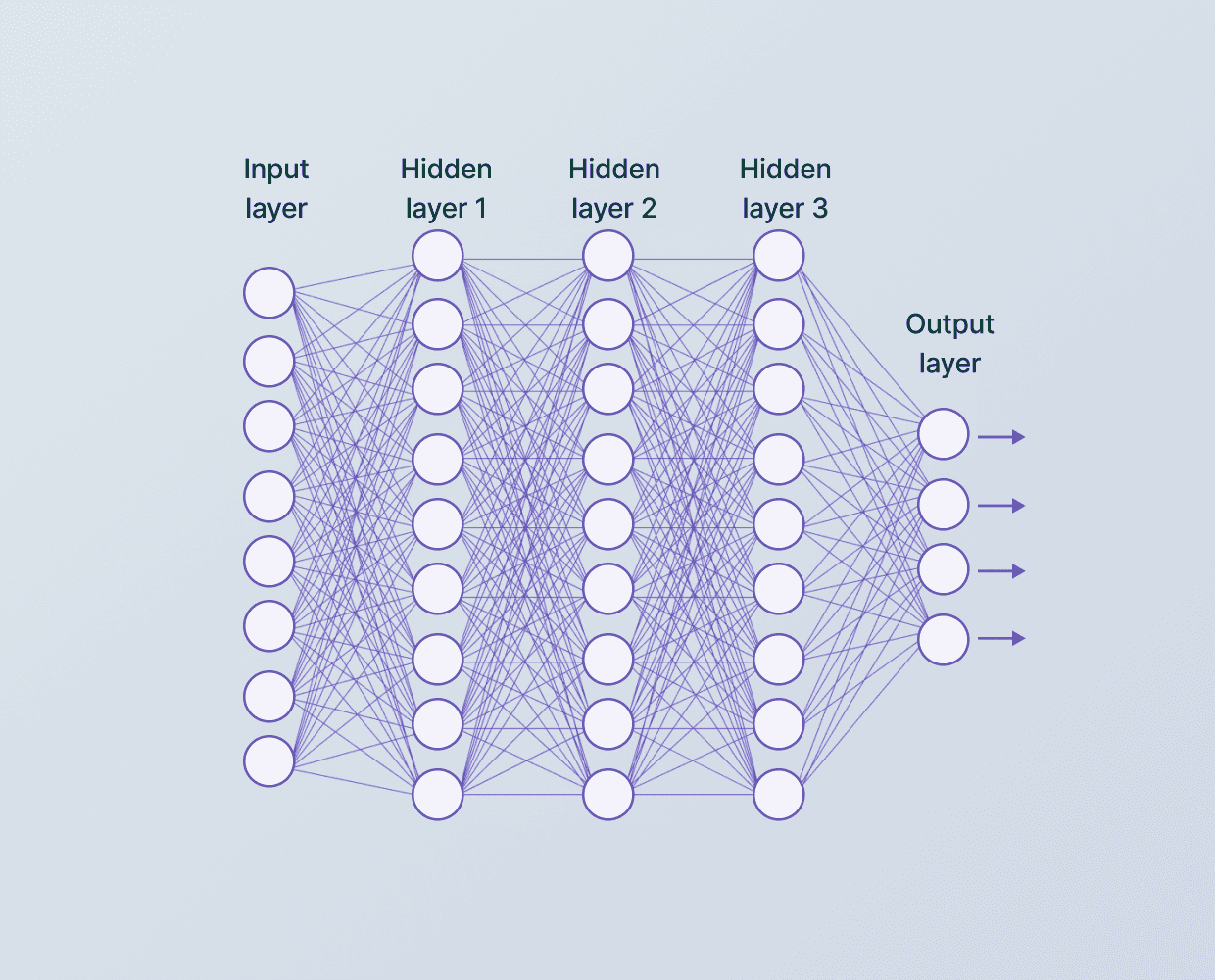
**Libraries:**

* TensorFlow
* Keras
* NumPy
* Pandas
* Matplotlib
* Scikit-Learn

1. **Theory**

**Definition:**  
A feedforward neural network is an artificial neural network where information flows only in one direction—from input nodes through hidden layers to output nodes—without forming cycles.

**Structure:**

1. **Input Layer:** Receives input features.
2. **Hidden Layers:** Perform computations; each neuron connects to every neuron in the next layer.
3. **Output Layer:** Produces final predictions.

**Fig. 1. Layers**

**Activation Functions:** ReLU, Sigmoid, and SoftMax introduce non-linearity for modeling complex relationships.

**Backpropagation:** Used to update network weights by propagating errors backward through the network.

1. **Methodology**
2. **Data Acquisition:**
   * Load the dataset using Pandas, which contains chemical properties of red wine and their quality ratings.
3. **Data Preparation:**
   * Split the dataset into training (75%) and validation (25%) sets.
   * Normalize features to a 0–1 range to ensure faster and stable training.
4. **Model Architecture:**
   * Use Keras Sequential API to build the model:
     + Input layer: 64 neurons, ReLU activation
     + Hidden layer: 64 neurons, ReLU activation
     + Output layer: 1 neuron for regression
5. **Model Compilation:**
   * Compile using the Adam optimizer and Mean Absolute Error (MAE) as the loss function.
6. **Model Training:**
   * Fit the model on the training data with validation monitoring over multiple epochs.
7. **Model Evaluation:**
   * Predict wine quality on the validation set and compare with actual values.
8. **Loss Visualization:**
   * Plot training and validation loss over epochs to assess model performance and overfitting.
9. **Advantages**

* **Handles Non-linearity:** Activation functions enable modeling complex patterns.
* **Flexible Architecture:** Easily adjust layers, neurons, and activation functions for different tasks.
* **Scalable:** Can handle large datasets with deeper networks.
* **Robust:** Generalizes well to unseen data when properly trained.
* **Parallel Computation:** Suitable for GPU acceleration for faster training.

1. **Limitations**

* **Data Requirements:** Needs large labeled datasets; small datasets may cause overfitting.
* **Computational Cost:** Training deep networks can be resource-intensive.
* **Black-box Nature:** Internal decision-making is hard to interpret.
* **Overfitting Risk:** Complex networks may capture noise instead of patterns.
* **Hyperparameter Sensitivity:** Requires careful tuning of learning rate, layers, and units.

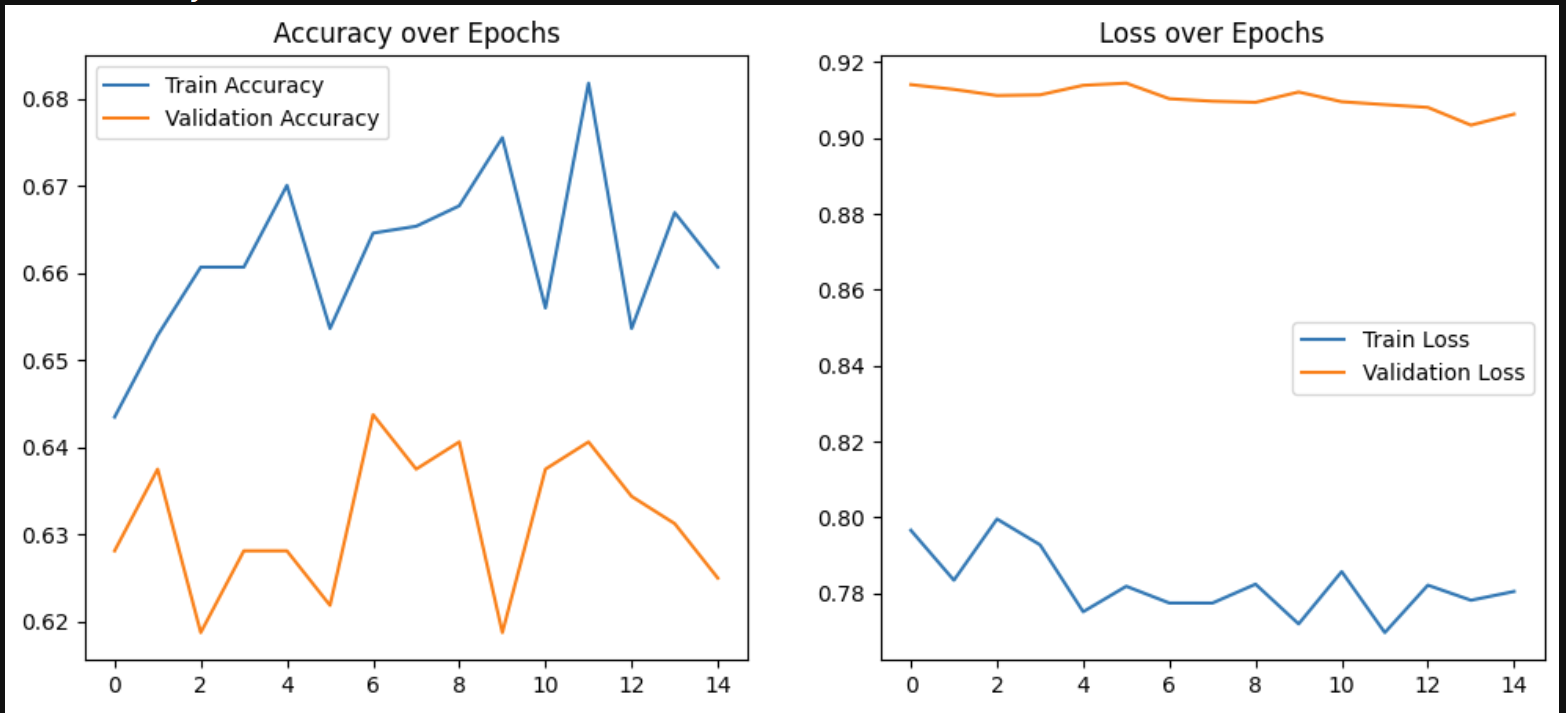
1. **Applications**

* Regression and classification tasks such as predicting wine quality.
* Modeling non-linear relationships in various domains like finance, healthcare, and engineering.
* Scalable predictive modeling with potential for parallel processing on modern hardware.

1. **Algorithm / Stepwise Procedure**
2. **Import Libraries:** NumPy, Pandas, TensorFlow, Keras, Matplotlib.
3. **Load Dataset:** Read winequality-red.csv into a Pandas DataFrame.
4. **Split Data:** 75% for training, 25% for validation.
5. **Normalize Features:** Min-max scaling to range (0,1).
6. **Separate Features and Target:** Define X\_train, y\_train, X\_val, y\_val.
7. **Define Input Shape:** Based on number of features.
8. **Build Model:**
   * Dense(64, activation='relu') → Dense(64, activation='relu') → Dense(1)
9. **Compile Model:** Optimizer = Adam, Loss = MAE
10. **Train Model:** Fit on training data with validation monitoring
11. **Evaluate Model:** Predict on validation set and compare to actual labels
12. **Visualize Loss:** Plot training vs validation loss over epochs
13. **Ready for Predictions:** Use trained model for new data
14. **Conclusion**

Feedforward Neural Networks (FNNs) are powerful tools for predictive modeling, capable of learning complex, non-linear relationships in data. When applied to problems like wine quality prediction, they provide accurate predictions and actionable insights. Despite challenges such as computational cost, risk of overfitting, and the need for careful hyperparameter tuning, FNNs remain highly effective for various regression and classification tasks when implemented and trained appropriately.

1. **Output**

****

**Fig. 2. Output**