**Assignment 1**

**Problem Statement:** Implementing Feedforward neural networks in Python using Keras and TensorFlow.

**Library:**

1. **Keras**: A high-level neural network API, written in Python, that can run on top of TensorFlow. It allows for easy and fast prototyping and building deep learning models.
2. **TensorFlow**: An open-source platform for machine learning, primarily used for building deep learning models. It provides flexibility in defining custom architectures and optimizes performance for production.

**Theory:**

Feedforward neural networks (FNNs) are the simplest type of artificial neural network architecture. The data flows in one direction, from input nodes through hidden nodes to output nodes, without any loops or cycles. FNNs consist of an input layer, one or more hidden layers, and an output layer. Each layer is made up of nodes (neurons), where each neuron is connected to all neurons in the previous and next layer, known as a fully connected or dense layer.

The process of training an FNN includes:

* **Forward Propagation**: Inputs pass through the network to produce an output.
* **Backpropagation**: The network adjusts its weights based on the error (the difference between predicted and actual outputs) to minimize the loss function.
* **Activation Functions**: These functions (such as ReLU, sigmoid, or softmax) introduce non-linearity, allowing the network to learn complex patterns.

**Methodology:**

1. **Data Preprocessing**: The dataset is first prepared by normalizing or standardizing the input features. Missing values, if any, are handled appropriately. For this example, we assume the dataset has been cleaned and is ready for training.
2. **Model Definition**:
   * Define a sequential model using Keras.
   * Add fully connected (dense) layers between the input and output layers. Common activation functions include ReLU for hidden layers and softmax for output (in classification tasks).
3. **Compilation**:
   * The model is compiled by specifying the optimizer (like Adam), loss function (like categorical\_crossentropy for classification tasks), and metrics (such as accuracy).
4. **Training**:
   * The model is trained using backpropagation over multiple epochs, and batch size is defined to handle updates after a few samples. The dataset is split into training and validation sets for performance tracking.

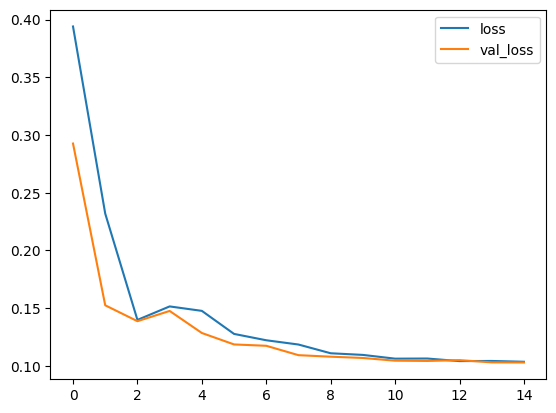


Fig 1: Training vs Validation Loss over 15 Epochs

1. **Evaluation**:
   * The model is evaluated using test data that was not seen during training, and performance metrics are recorded.

**Advantages:**

1. **Simplicity**: Easy to implement, especially with high-level libraries like Keras and TensorFlow.
2. **Fast Training**: With proper hardware, FNNs can be trained quickly for small to medium-sized datasets.
3. **Universal Approximation**: FNNs with sufficient layers and neurons can approximate any function, making them versatile for various tasks.

**Disadvantages:**

1. **Limited in Learning Complex Features**: Compared to architectures like Convolutional Neural Networks (CNNs), FNNs struggle with spatial hierarchies in image data.
2. **Overfitting**: Without proper regularization (e.g., dropout, L2 regularization), FNNs can easily overfit, especially on small datasets.
3. **Computationally Intensive**: Large FNNs with many neurons and layers require significant computational power and memory.

**Conclusion:**

Feedforward Neural Networks (FNNs) are foundational models in deep learning. By implementing an FNN using Keras and TensorFlow, we can develop a powerful tool to classify or predict data. Although they are simple, FNNs can serve as a stepping stone to more advanced architectures like CNNs or Recurrent Neural Networks (RNNs), particularly for applications in computer vision, natural language processing, and more.