Project Report

Intern Name: Ahmed Islam

Department: Artificial Intelligence

Organization: IR Solutions

Designation: AI Intern

Week # 03

Submitted to: Asad Ali

Submission Date: 25-July-2025

Report Type: Combined Tasks Documentation

Table of Contents

Theoretical Understanding

Neural networks

"Nonlinear" means that you can't accurately predict a label with a model of the form

b + w1x1 + w2x2…..

. In other words, the "decision surface" is not a line.

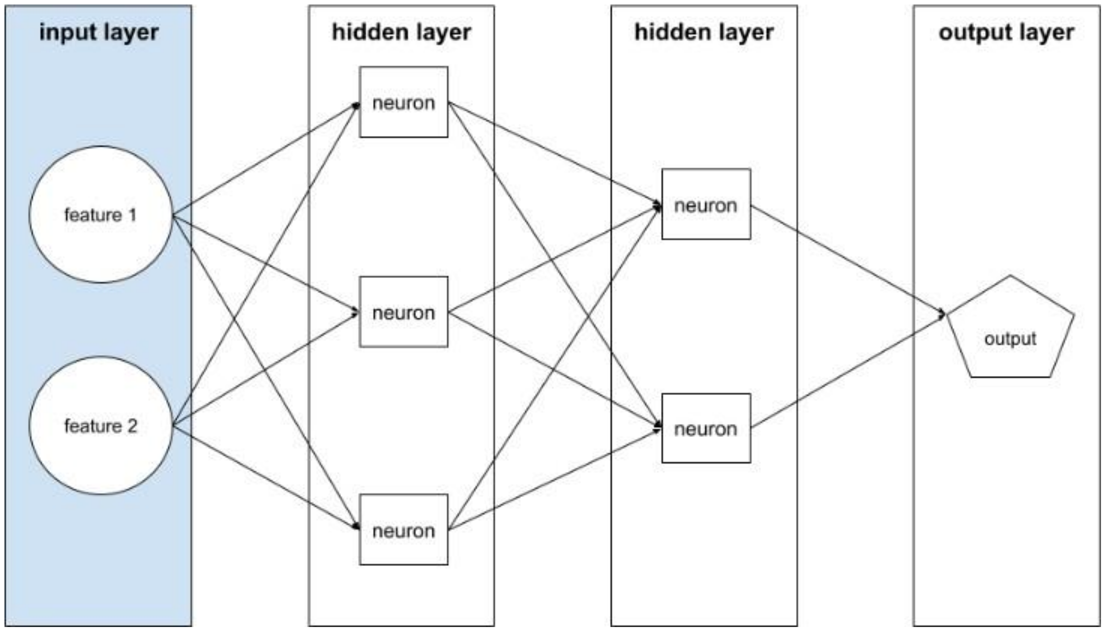
Neural networks are a family of model architecture designed to find nonlinear patterns in data. During training of a neural network, the model automatically learns the optimal feature crosses to perform on the input data to minimize loss.

Weight

A value that a model multiplies by another value. Training is the process of determining a model's ideal weights; inference is the process of using those learned weights to make predictions.

Input layer

The layer of a neural network that holds the feature vector. That is, the input layer provides examples for training or inference.



Hidden layer

A layer in a neural network between the input layer (the features) and the output layer (the prediction). Each hidden layer consists of one or more neurons.

Neurons

The nodes in these layers are called neurons.

Activation functions

A nonlinear transform of a neuron's output value before the value is passed as input to the calculations of the next layer of the neural network.

Activation function used for introducing the nonlinear relationship

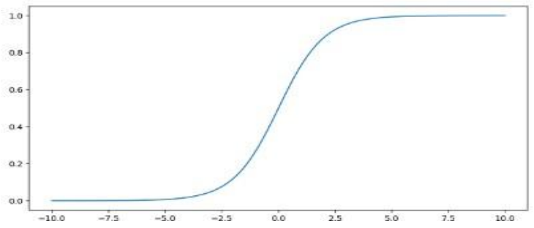
Sigmoid activation function

The sigmoid function (discussed above) performs the following transform on input x, producing an output value between 0 and 1:

$$F(x) = \frac{1}{1 + e^{-x}}$$

★ The term *sigmoid* is often used more generally to refer to any S-shaped function. A more technically precise term for the specific function $F(x) = \frac{1}{1+e^{-x}}$ is *logistic function*.

Here's a plot of this function:

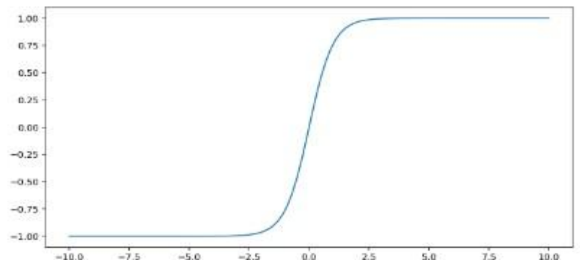


Tanh Activation function

The tanh (short for "hyperbolic tangent") function transforms input x to produce an output value between −1 and 1:

$$F(x) = tanh(x)$$

Here's a plot of this function:



Relu (rectified linear unit)

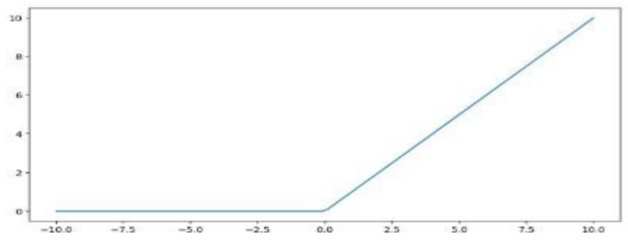
The **rectified linear unit** activation function (or **ReLU**, for short) transforms output using the following algorithm:

- If the input value x is less than 0, return 0.
- If the input value x is greater than or equal to 0, return the input value.

ReLU can be represented mathematically using the max() function:

$$F(x) = max(0, x)$$

Here's a plot of this function:



Vanishing gradient problem

It occurs when the gradients used to update weights during backpropagation become very small, effectively preventing the model from learning.

Backpropagation

Backpropagation (short for backward propagation of errors) is the core algorithm used to train neural networks. It allows the network to learn by adjusting the weights to minimize the error (or loss) between predicted and actual output.

Exploding Gradients

Exploding gradients is a problem in training deep neural networks where the gradients become excessively large during backpropagation. This causes the model's weights to grow uncontrollably, leading to unstable training and even numerical overflow.

Dead ReLU Units

Dead ReLU units refer to neurons in a neural network using the ReLU (Rectified Linear Unit) activation function that stop learning permanently — they always output zero, no matter the input.

Dropout Regularization

Dropout is a regularization technique used to prevent overfitting in neural networks. It works by randomly "dropping out" (i.e., setting to zero) a fraction of neurons during training — meaning these neurons are temporarily ignored along with their connections.

Multiclass classification

Multiclass classification is a type of machine learning problem where the goal is to classify inputs into more than two classes (categories).

- One vs All: For N classes, train N binary classifiers.
- One vs One: For N classes, train N × (N - 1) / 2 binary classifiers. Each classifier learns to distinguish between two specific classes.

Encoding

Encoding refers to the process of choosing an initial numerical representation of data to train the model on.

Embedding Space

An embedding space is a continuous vector space where discrete items (like words, images, or users) are represented as vectors. The goal is to map similar items closer together and dissimilar ones farther apart.

Static Embedding

Static embeddings are pre-trained embeddings where each word has a fixed vector, regardless of context.

Contextual Embedding?

Contextual embedding means that the vector representation of a word depends on its context. Contextual embeddings can generate different vectors for the same word depending on its meaning in the sentence.

Bag of words

Bag of Words is a simple, classic technique used in natural language processing (NLP) to represent text as numerical data.

Dimensionality Reduction Techniques

Dimensionality reduction is the process of reducing the number of features (dimensions) in your dataset while preserving important information.

Language model

A Language Model is a type of machine learning model that understands and predicts human language. It estimates the probability of a sequence of words, helping machines to generate or interpret text in a meaningful way.

N-gram Language Model

An N-gram language model is a statistical language model that predicts the next word in a sequence using the previous N-1 words.

Context

Context refers to the surrounding words, phrases, sentences, or even broader information that help define or influence the meaning of a specific word or expression.

LLM (Large Language Model)

A Large Language Model (LLM) is an AI model trained on massive amounts of text data to understand, generate, and interact with human language.

Tokens

Words or pieces of words.

Transformers

The Transformer is a deep learning model architecture introduced in 2017 (by Vaswani et al.) in the paper titled:

"Attention is All You Need"

It's the foundation of almost all modern language models including GPT, BERT, T5, etc.

Encoder

An encoder converts input text into an intermediate representation.

Decoder

A decoder converts that intermediate representation into useful text.

Multi-Head Attention

Multi-Head Attention is a core component of the Transformer architecture (used in models like BERT, GPT, etc.). It allows a model to focus on different parts of a sentence simultaneously to better understand context and meaning.

Attention

Attention is a mechanism that:

- Scores how relevant each word is to the current word.
- Gives higher weights to important words.

Foundation Model

A Foundation Model is a large-scale, pretrained AI model (typically a Large Language Model — LLM) that serves as a base (or foundation) for building many different downstream AI applications.

Online Inference

Predictions are made immediately when a user or system sends input to the model.

Offline Inference (Batch Inference)

Predictions are made in bulk on a pre collected dataset at scheduled time not immediately

Distillation

Distillation creates a smaller version of an LLM. The distilled LLM generates predictions much faster and requires fewer computational and environmental resources than the full LLM.

Task 1: Breast Cancer Classification

Objective

To build and evaluate various machine learning models for detecting breast cancer using the Wisconsin Breast Cancer dataset.

1. Load the Dataset

The dataset was loaded using pandas:

```
import pandas as pd
df = pd.read_csv("data.csv")
```

Shape of the dataset: 569 rows, 33 columns

2. Apply StandardScaler to Numerical Features

- Standardization was applied to normalize the feature values
  from sklearn.preprocessing import StandardScaler
  scaler = StandardScaler()
  X_scaled = scaler.fit_transform(X)

Why important?

- Models like SVM, Logistic Regression, and KNN are sensitive to feature scaling.
- Standardization improves training speed and accuracy.

3. Train/Test Split

- The data was split using a 70/30 ratio:

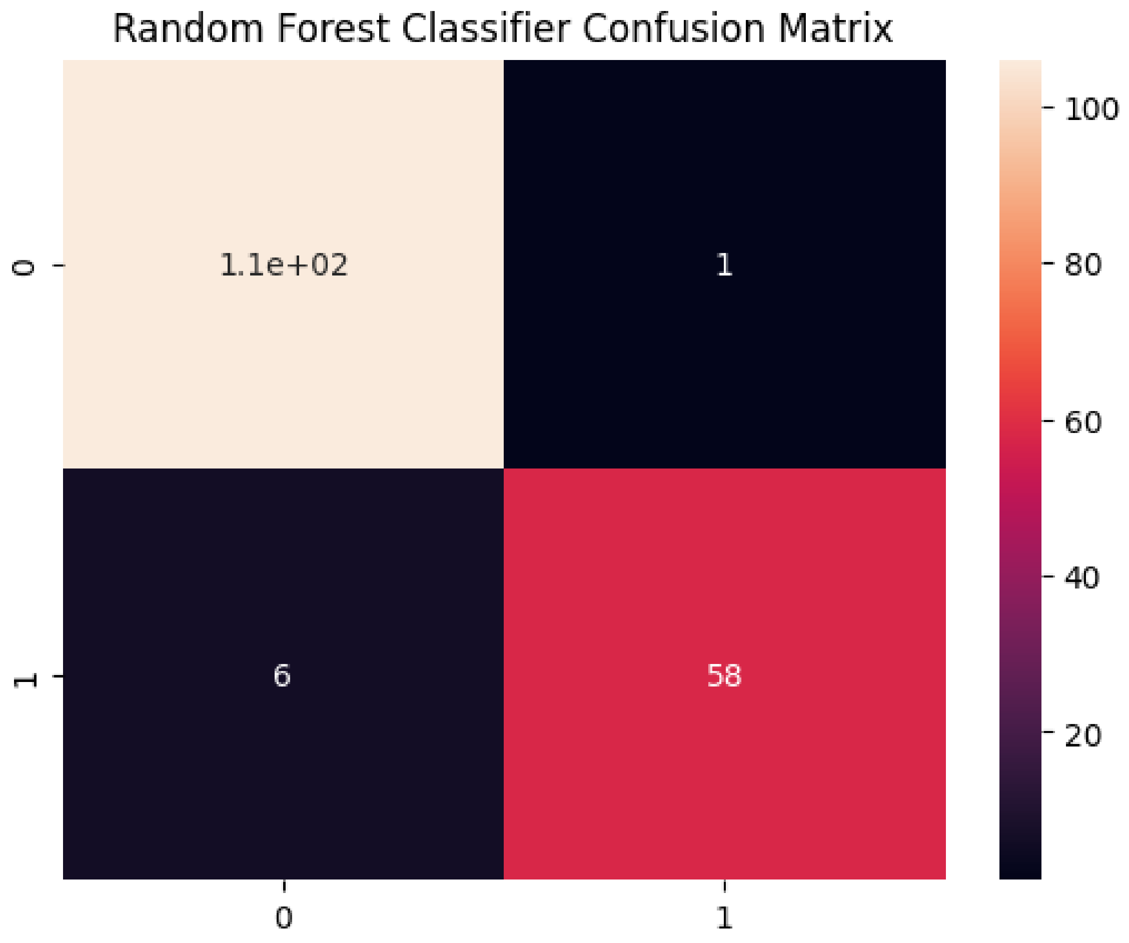
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, stratify=y, random_state=42)
```

- Training set size: 398
- Testing set size: 171

4. Model Training and Evaluation

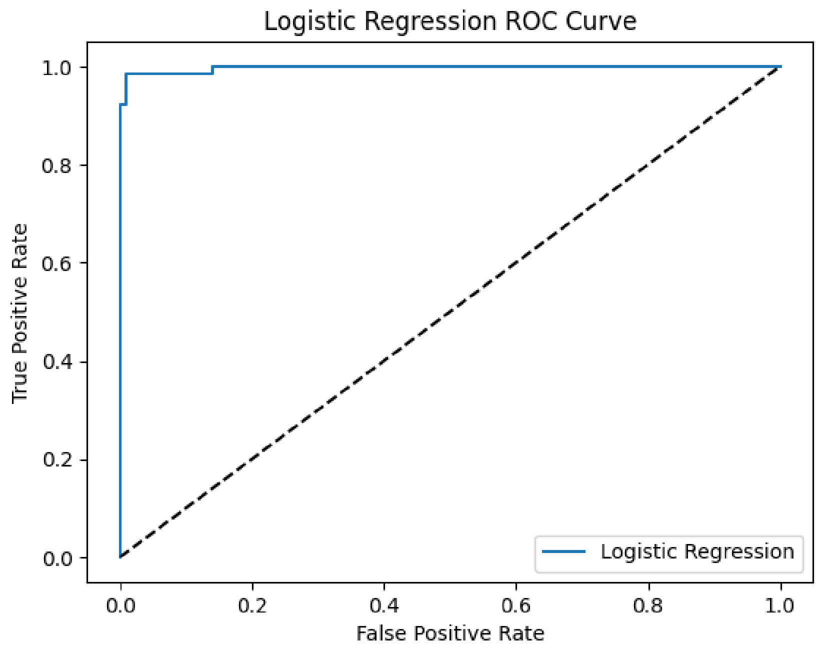
- Models used: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVC)
- Accuracy results:
  Logistic Regression Accuracy: G6.4G%
  Random Forest Accuracy: G5.G1%
  Support Vector Machine Accuracy: G8.25% (Best)
  K-Nearest Neighbors Accuracy: G6.4G%
- Best Performing Model: SVC, due to its effectiveness in handling high-dimensional, scaled data.

Confusion Matrix (Random Forest)

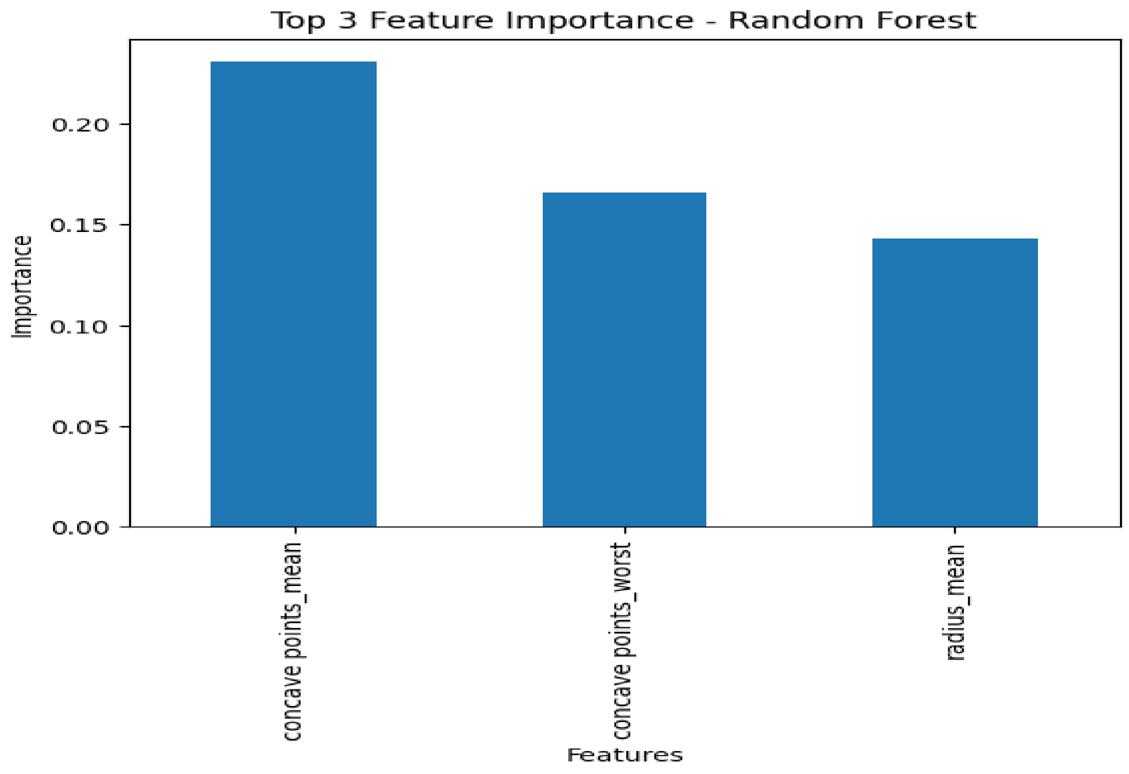


- Recall is crucial for minimizing false negatives in medical diagnosis.

ROC Curve and AUC (Logistic Regression)



Feature importance for Random Forest



Cross-Validation (SVC)

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(model_svc, X_scaled, y, cv=5)
```

Conclusion:

Support Vector Machine outperformed other models with the highest accuracy and stable performance across folds. Proper preprocessing (scaling, outlier handling, feature selection) significantly contributed to model performance. The saved model is ready for deployment or further analysis.

Task 2: Fashion MNIST Classification using Convolutional Neural Network (CNN)

Objective

To classify images of clothing items into 10 categories using a Convolutional Neural Network trained on the Fashion MNIST dataset.

Load the Dataset

from tensorflow.keras.datasets import fashion_mnist

(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

- Training set shape: (60000, 28, 28)
- Testing set shape: (10000, 28, 28)

Data Preprocessing

- **Normalized pixel values between 0 and 1**

X_train = X_train / 255.0

X_test = X_test / 255.0

- **Reshaped for CNN input:**

X_train = X_train.reshape(-1, 28, 28, 1)

X_test = X_test.reshape(-1, 28, 28, 1)

- **One-hot encoded the labels:**

from tensorflow.keras.utils import to_categorical

y_train_cat = to_categorical(y_train)

```
y_test_cat = to_categorical(y_test)
```

CNN Model Architecture

```
from tensorflow.keras import layers, models

model = models.Sequential([

  layers.Conv2D(128, (3, 3), activation='relu', input_shape=(28, 28, 1)),

  layers.MaxPooling2D(2, 2),

  layers.Conv2D(64, (3, 3), activation='relu'),

  layers.MaxPooling2D(2, 2),

  layers.Flatten(),

  layers.Dense(128, activation='relu'),

  layers.Dense(10, activation='softmax')

])
```

Compile and Train the Model

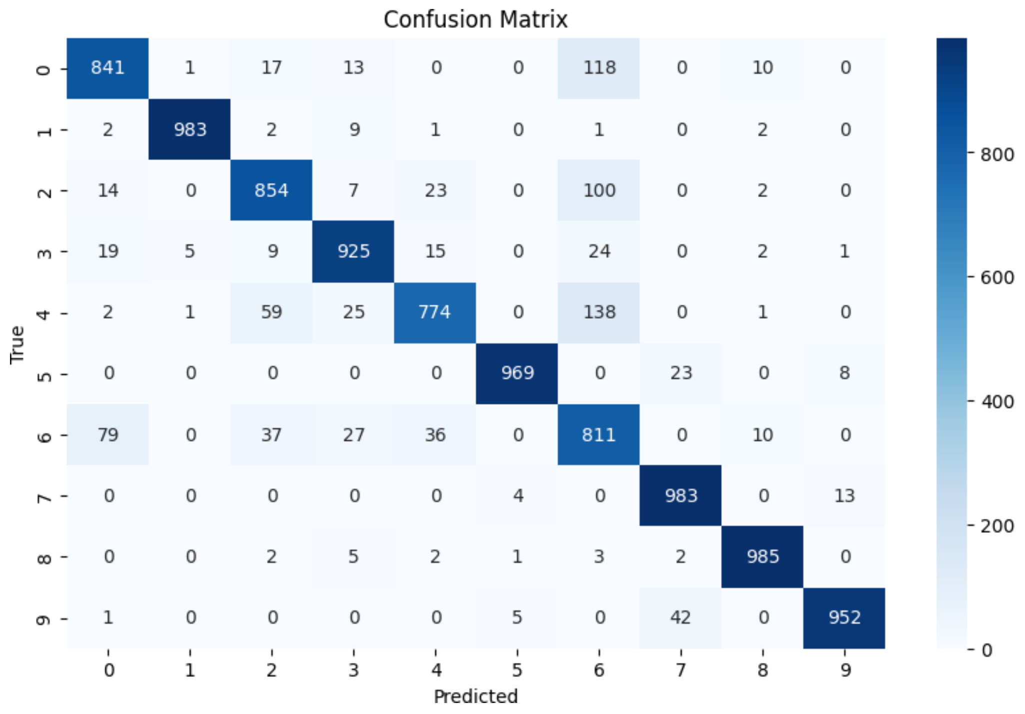
- Compiled using Adam optimizer and categorical crossentropy loss
- Trained for 10 epochs:

Model Evaluation

```
test_loss, test_accuracy = model.evaluate(X_test, y_test_cat)
```

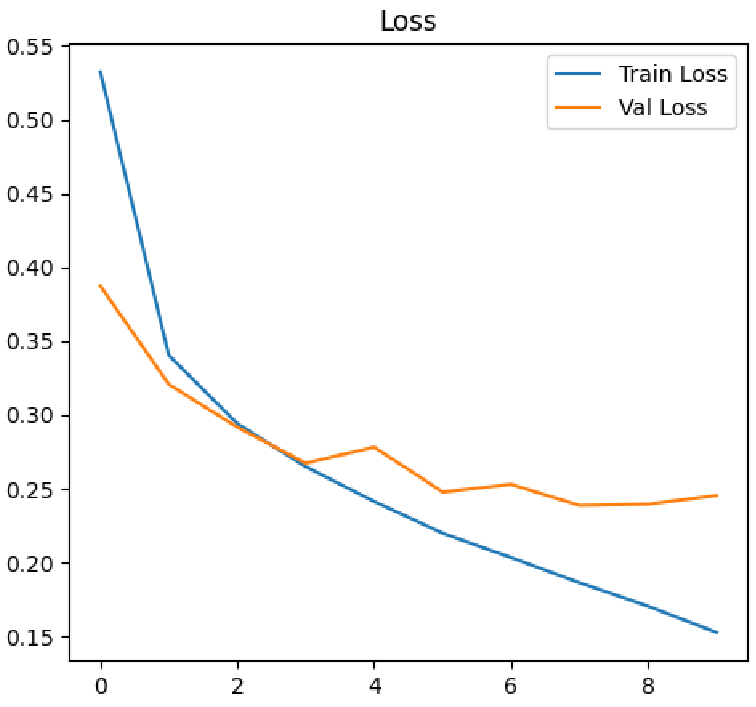
- Test Accuracy: Approximately 92–93%

Confusion Matrix

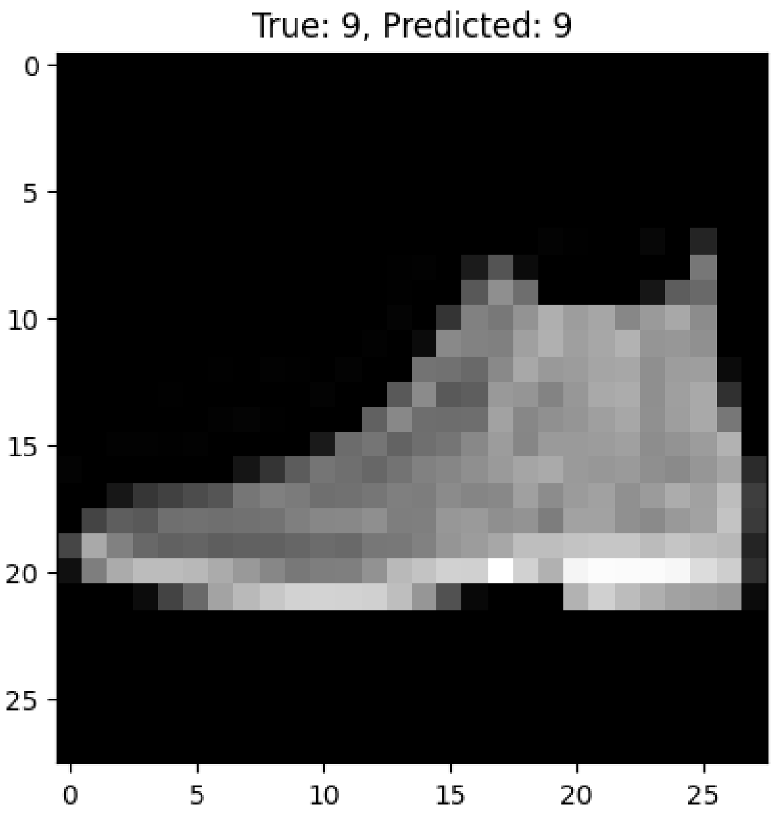


True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	841	1	17	13	0	0	118	0	10	0
1	2	983	2	9	1	0	1	0	2	0
2	14	0	854	7	23	0	100	0	2	0
3	19	5	9	925	15	0	24	0	2	1
4	2	1	59	25	774	0	138	0	1	0
5	0	0	0	0	0	969	0	23	0	8
6	79	0	37	27	36	0	811	0	10	0
7	0	0	0	0	0	4	0	983	0	13
8	0	0	2	5	2	1	3	2	985	0
9	1	0	0	0	0	5	0	42	0	952

Loss



Model Inference



Conclusion

The CNN model achieved strong performance on Fashion MNIST with over 92% accuracy. The model generalizes well and performs reliably across folds. Preprocessing steps like normalization and one-hot encoding were crucial. The trained model is now saved and ready for real-world applications or integration.

Additional Learning

I learned about git and GitHub.
Explore that how we can push data, how to merge branches and why we use GitHub.

YouTube Video: https://www.youtube.com/watch?v=NcoBAfJ6l2QCt=301s

Also explore the Support vector machine and ensemble learning.