Software Design Document

for

Detecting Damaged Components of Rockets Using Deep Learning Model

Version 1.0 approved

Prepared by

TEAM 2:
Abhiram Krishnan RS
Akshay K
Albin Zachariah Daniel
Amal Thomas

TKM College of Engineering

1 April 2023

Table of Contents

1. Introduction

The Detecting Damaged Components of Rockets is a Deep Learning Model designed to detect damaged components from images. The system utilizes complex convolutional networks and deep learning algorithms to extract features in damaged and undamaged components for damage detection capability. This project aims to reduce manual effort and increase efficiency in detecting damages for saving the time and effort quality assurance personnel by automating the damage detection process.

a. Purpose

The purpose of an SDD for a damaged component detection model is to provide a detailed overview of the system's design, architecture, and development processes. It serves as a comprehensive reference for developers, testers, and project stakeholders, and guides the project team during implementation to ensure adherence to design specifications.

b. Scope

The scope of an SDD for a damage detection model is to provide a detailed description of the system's architecture, design, and implementation. It is a comprehensive reference for developers, testers, and project stakeholders to understand the system's design and development processes.

c. Intended Audience

Anyone with some programming experience and familiarity with Python and its libraries related to Deep Learning like TensorFlow, can understand this document. The document is intended for developers, software architects, testers, project managers, and documentation writers.

d. References

a. Data Augmentation: https://www.tensorflow.org/tutorials/images/data_augmentation
b. Image Classification: https://www.tensorflow.org/tutorials/images/classification
c. MobileNet: https://arxiv.org/abs/1704.04861
d. EfficientNet: https://arxiv.org/pdf/1905.11946

2. System Architecture Design

The system architecture of the damaged component detection system comprises the following components:

a. Image Pre-processing Module:
The image pre-processing module is responsible for converting input image to the required input format for the model.

b. DL Model Module:
The deep learning model accepts input images and classifies them as damaged or not.

c. Result Module:
The results are displayed in a tabular format that shows the number of components of each type.

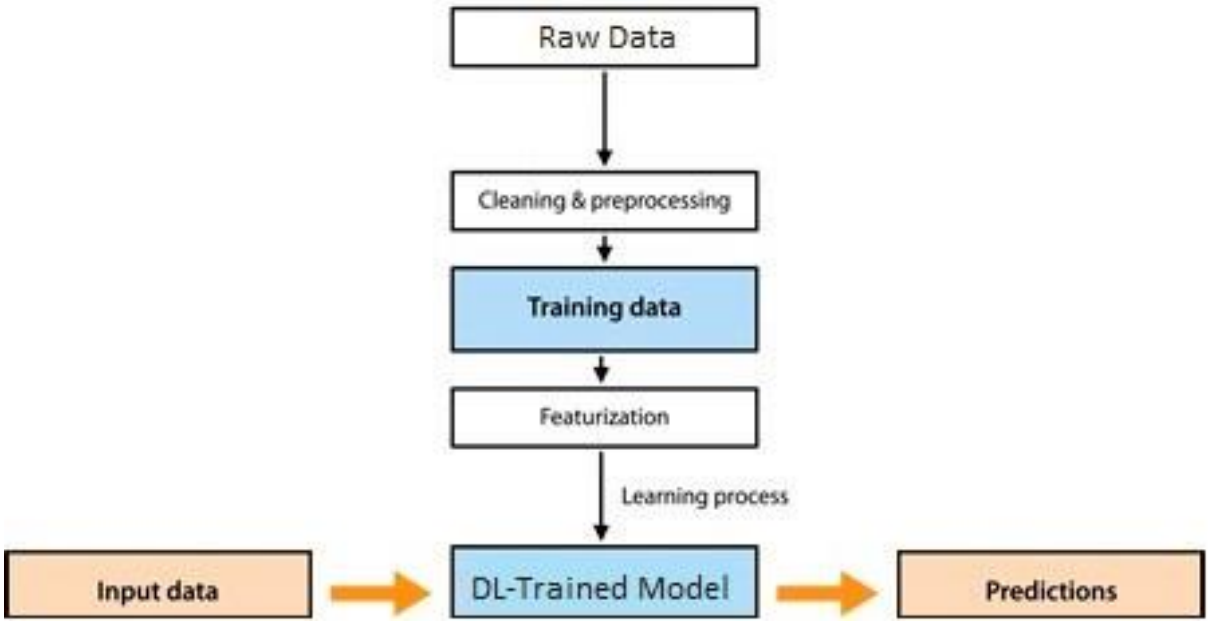


Figure 1 - Overall Architecture

3. Data Pre-processing

The system shall perform data pre-processing techniques, including image normalization, resizing, and data augmentation. Image normalization can reduce the impact of varying lighting conditions, while resizing can help the model handle images of different sizes.

Data augmentation can generate a diverse dataset and reduce overfitting. These techniques are especially important for rocket component damage detection, as images may be captured in harsh environments with varying levels of detail and lighting.

a. Flipping:
By applying flipping techniques, the system can produce a mirrored version of the image, providing additional variations to the dataset without distorting the appearance of the damage. This technique can also help the model to generalize better to unseen data, making it more robust and accurate in real-world scenarios.

b. Rescaling:
It is a useful technique for improving the performance of the model by reducing the computational resources required for training. Rescaling can also help to improve the accuracy of the model by reducing the impact of irrelevant image features.

c. Rotation:
Rotation provides additional variations to the dataset which improves the model's ability to detect damages at various angles. However, it is important to ensure that the rotation does not distort or change the appearance of the damage.

d. Shear:
Shearing can help the model learn to detect objects even when they are tilted or rotated in the image. However, it is important to note that shearing can also introduce some artifacts and distortions in the image. Therefore, it is essential to carefully evaluate the impact of this technique on the dataset and the performance of the model.

4. Model Architecture

The deep learning model have been designed using several convolutional layers with backpropagation.

The deep learning model has the following architecture:

- Input Image: The input to the deep learning model is a pre-processed image.
- Convolutional Layers: The deep convolutional layers extract features from the input image.
- Fully Connected Layers: The fully connected layers analyze the extracted features and predict the class probabilities.

- Output: The output of the deep learning model is a classification result with corresponding labels in a multi-dimensional array with confidence scores in each of the two classes.

Batch Normalization

Batch normalization is a technique used to improve the training of deep neural networks. It works by normalizing the input to each layer of a neural network to have zero mean and unit variance. This helps to stabilize the distribution of the inputs and gradients throughout the network, which can help speed up training and improve the accuracy of the model.

Advantages;

- Faster training.
- Reduced dependence on dropout.
- Improved generalization.

Inverted Residual with Linear Bottleneck

Invert Residual block follows a narrow->wide->narrow approach. The first step widens the network using a 1x1 convolution because the following 3x3 depthwise convolution already greatly reduces the number of parameters. Afterwards another 1x1 convolution squeezes the network in order to match the initial number of channels.

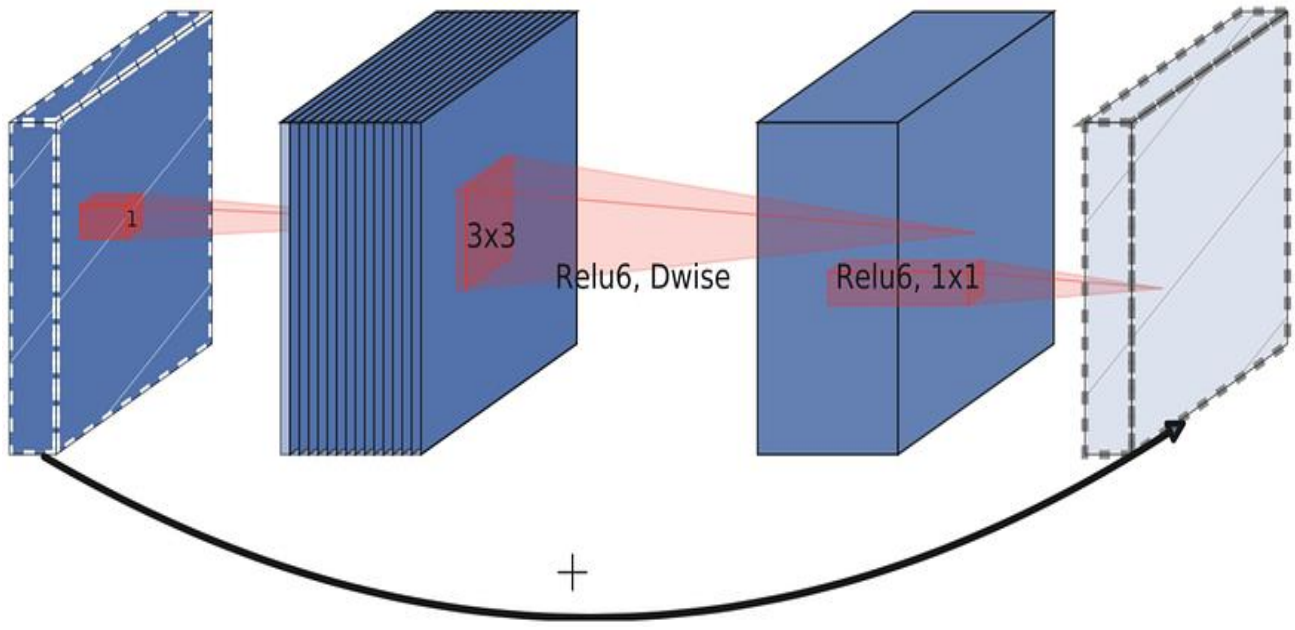


Figure 2 - Architecture of Inverted Residual Block

The activation function ReLU, which is commonly used in neural networks, discards values that are smaller than 0. This loss of information can be tackled by increasing the number of channels in order to increase the capacity of the network.

With inverted residual blocks we do the opposite and squeeze the layers where the skip connections are linked. the idea of a linear bottleneck where the last convolution of a residual block has a linear output before it's added to the initial activations.

Skip Connection

The skip connection in the inverse residual block is used to improve the performance and training of deep convolutional neural networks. By connecting the input of the block directly to the output of the block, the skip connection allows the network to preserve important features and gradients while also reducing the vanishing gradient problem.

Improved feature preservation: By allowing the input to bypass the intermediate layers of the block and directly connect to the output, the skip connection can help preserve important features of the input tensor. This can improve the accuracy and performance of the network.

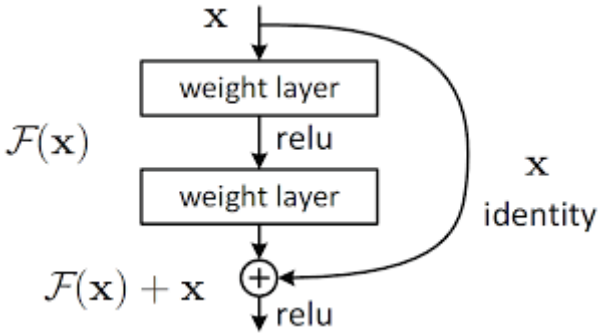


Figure 3 - Depiction of Skip Connection

Reduced vanishing gradient problem: The skip connection can help alleviate the vanishing gradient problem, which occurs when gradients become too small to update the parameters of deep neural networks. By connecting the input directly to the output, the skip connection can help preserve gradients and improve the training of the network.

Depth-wise Separable Convolutions

Depth-wise separable convolutions were introduced in MobileNetV1 and are a type of factorized convolution that reduce the computational cost as compared to standard convolutions.

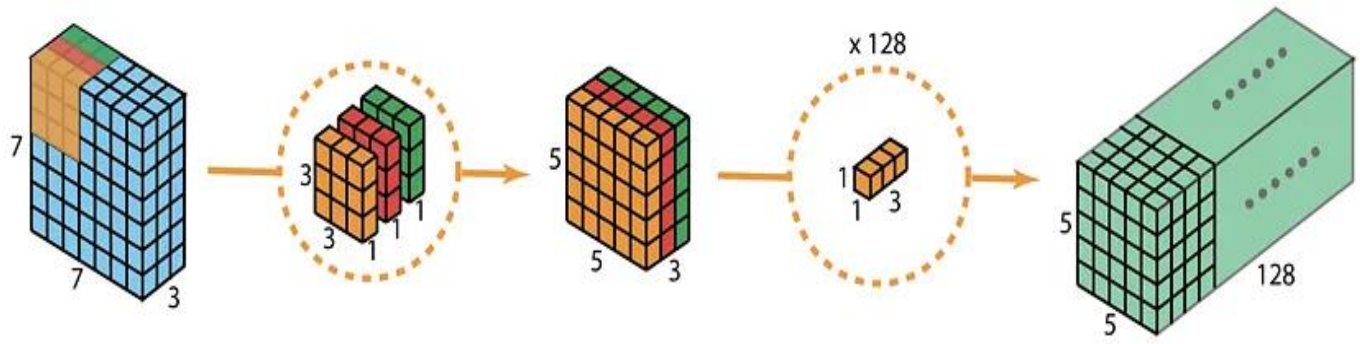


Figure 4 - Working of Depth-wise Separable Convolutions

The first set of filters, shown in Fig. 4(b), are comprised of *M* single-channel filters, mapping the input volume to D_v x D_v x *M* on a **per-channel** basis. This stage, known as depth-wise convolutions.

In order to construct new features from those already captured by the input volume, we require a **linear combination**. To do so, 1x1 kernels are used along the depth of the intermediate tensor; this step is referred to as point-wise convolution. *N* such 1x1 filters are used, resulting in the desired output volume of D_v x D_v x *N*.

Standard convolution has a computational cost on the order of

$$D_r^2 M D_v^2 N$$

depth-wise convolution has a computational cost on the order of

$$D_r^2 D_v^2 M + M D_v^2 N.$$

Resolution (r)

the features are more fine-grained and hence high-res images should work better. This is also one of the reasons that in complex tasks, like Object detection, we use image resolutions like 300x300, or 512x512, or 600x600.

Hyperparameters provided:

a) Optimizer: ADAM is a stochastic gradient descent optimization algorithm that combines two methods: adaptive learning rates and momentum. It requires less memory than other optimization algorithms. It is a popular choice for training deep neural networks due to its fast convergence, low memory requirements, and ease of hyperparameter tuning.

b) Activation Function: It is a variant of the Rectified Linear Unit (ReLU) activation function that is clipped at a maximum value of 6. MobileNet models have a large number of convolutional filters, which leads to a high memory requirement. ReLU6 limits the output to a maximum value of 6, which reduces the range of possible values and allows more efficient use of memory.

c) Dropout: Dropout is a regularization technique used to prevent overfitting in neural networks. During training, dropout randomly drops out some of the neurons in the network. This helps to prevent the network from relying too much on any one particular feature, which can lead to overfitting.

d) Loss Function: Sparse categorical cross-entropy is a loss function commonly used in multi-class classification problems where the target labels are represented by integer values. In contrast to categorical cross-entropy, which expects the target labels to be one-hot encoded, sparse categorical cross-entropy accepts the integer representation of the target classes.

Model Architecture

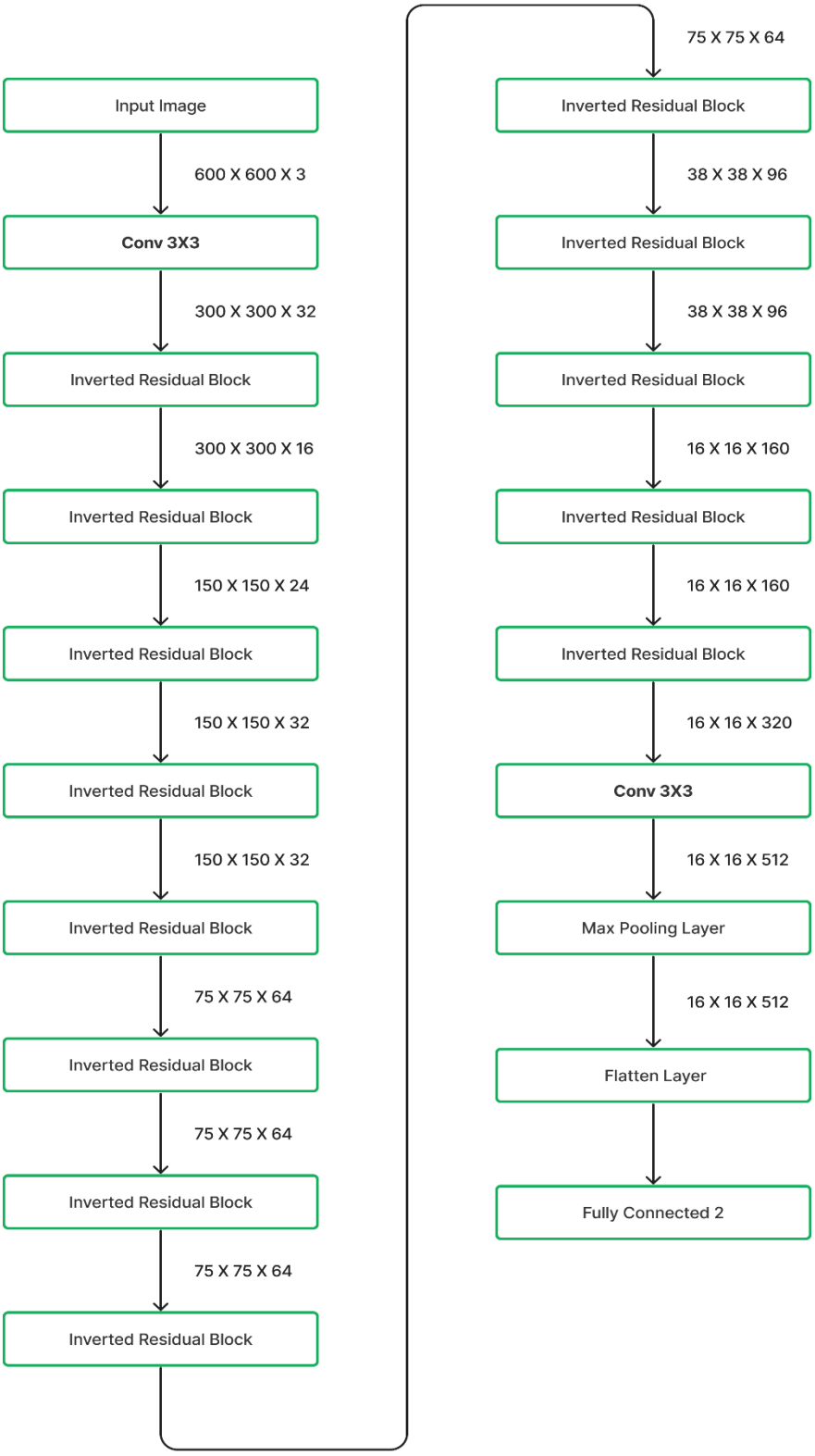
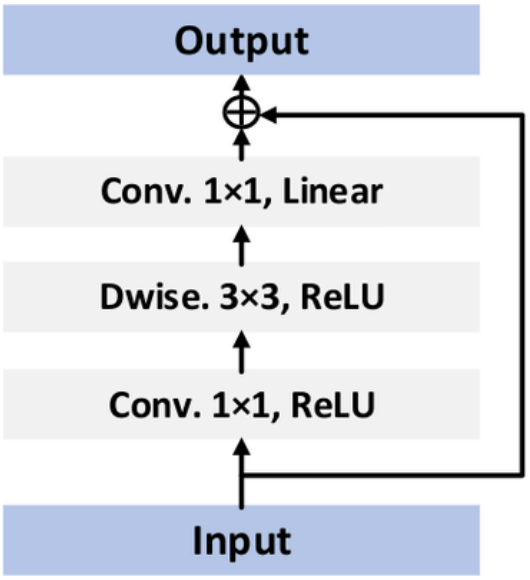


Figure 5 - Model Architecture



Inverted Residual Block

5. Performance Metrics

Confusion Matrix:

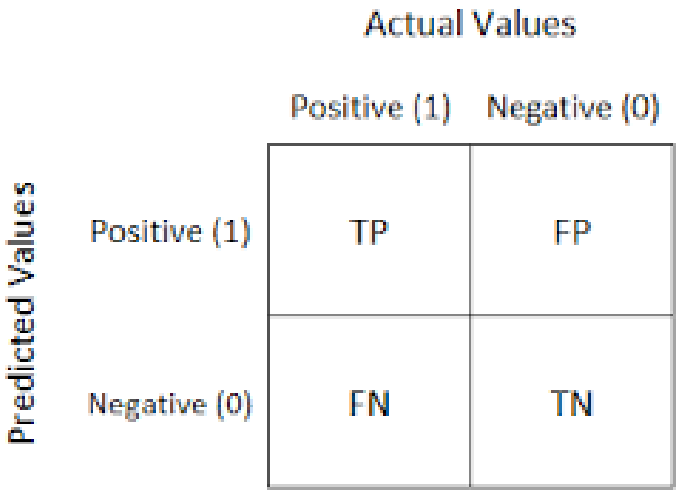


Figure 6 - Confusion Matrix

TP - True Positive (Damaged component correctly identified)

TN - True Negative (Undamaged component correctly identified)

FP - False Positive (Undamaged component incorrectly identified as damaged)

FN - False Negative (Damaged component incorrectly identified as undamaged)

Recall:

$$Recall = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Negative(FN)}$$

Recall measures the percentage of actual damages that were correctly identified by the model. A high recall value indicates that the model is effective at detecting damages, which is crucial in a damage detection system where false negatives (i.e., missing actual damages) can be costly and dangerous.

Precision:

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

Precision measures the percentage of the model's predictions that are correct. A high precision value indicates that the model is not producing many false positives (i.e., predicting damages when there are none), which is also important in a damage detection system as false positives can lead to unnecessary maintenance and repair costs.

Accuracy:

$$\text{Accuracy} = \frac{\text{True Positives + True Negatives}}{\text{All Samples}}$$

Accuracy is the ratio of the number of correct predictions to the total number of predictions. It measures the overall correctness of the model's predictions.

F1 Score:

$$F1\ Score = 2 \times \frac{recall \times precision}{recall + precision}$$

F1 score is a measure of the overall accuracy of a binary classification model, taking into account both precision and recall. It is the harmonic mean of precision and recall, giving equal weight to both measures.

ROC Curve:

A Receiver Operating Characteristic (ROC) curve is a graphical representation of the performance of a binary classifier system. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) for different classification thresholds.

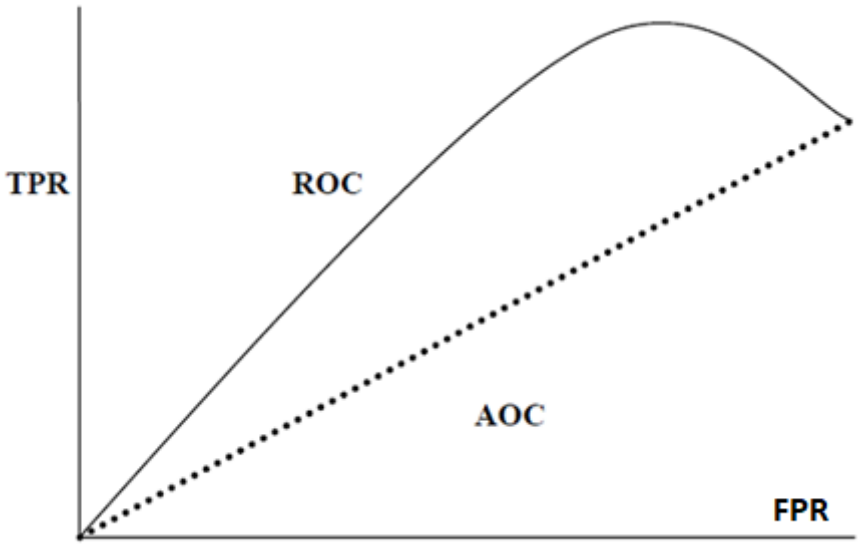


Figure 7 - Depiction of ROC Curve

The AUC summarizes the overall performance of the binary classification model across all possible classification thresholds. The AUC score ranges from 0 to 1, where an AUC of 1 indicates perfect classification and an AUC of 0.5 indicates random guessing. Therefore, a higher AUC score indicates better model performance.

6. Technological Stack

- Python Programming Language:

  Python is a high-level, general-purpose programming language. It supports multiple programming paradigms including structured, object oriented and functional programming.

- TensorFlow Framework:

  TensorFlow is a machine learning framework used to develop or pre-train machine learning models.

- Flask Framework:

  Flask is a web-based framework that can be used to create a web API that accepts requests and returns predictions from the model.

- Libraries Used:

  NumPy: NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices.

  OS: This module provides a portable way of using operating system dependent functionality.

  matplotlib: Matplotlib is a popular Python library used for creating 2D plots and graphs. It provides a wide variety of customizable plots such as line plots, scatter plots, bar plots, histograms, and more.

Appendix A: Record of Changes

Version	Date	Author	Description of change
1.0	30/03/2023	Akshay	Initial draft
1.0	30/03/2023	Amal	Added Model architecture
1.0	31/03/2023	Abhiram	Added Technology Stack
1.0	01/04/2023	Albin	Added Data Pre-processing
1.0	01/04/2023	Akshay	Updated Technology Stack
1.0	01/04/2023	Amal	Edited Model Architecture