

# **Internship Report**

## **Aim of the internship:**

To train a neural network that takes the base frequency values (f1 through f10) as inputs and predicts the crack characteristics (X-Position, Y-Position, Radius) as outputs.

## **Theory of the project:**

My project revolves around the challenge of understanding how fractures develop in homogenous materials under stress, such as the material's ability to withstand maximum loads before failing. This involves considering various factors including the material's behaviour (whether brittle or ductile), its properties, the shape and size of the specimen, how it's loaded, and any existing defects or flaws it might have. Typically, experiments and theoretical models are used to determine how much load these materials can handle. However, the complexity of fracture behaviour, particularly in materials that fracture in a quasi-brittle manner, makes precise predictions difficult.

In the project/problem statement, I utilized a dataset that provides specific details about the position and size of a hole within a material. These details are critical as they influence the structural integrity and load-bearing capacity of the material. The dataset includes output frequencies that I used as inputs for a machine learning model. This model aims to predict the characteristics of the hole, such as its exact location and size, based on these frequencies. By comparing the model's predictions with actual measurements, I helped in evaluating the accuracy and effectiveness of my approach.

The use of frequencies in my study may relate to Acoustic Emission (AE) testing, a technique where sound waves and energy emitted due to structural changes within a material are analysed. AE is commonly used to detect or predict failures, observe material degradation, or even monitor seismic activities. While it's not specified if my project's frequencies are directly derived from AE testing, they serve a similar purpose by providing insights into the material's response to stress, which is essential for assessing its integrity and predicting failures.

## **Tools and Library requirements:**

**Programming Languages:** Python

**Libraries/Frameworks:** TensorFlow, Scikit-learn, Pandas, NumPy.

**Tools/Platforms:** Jupyter Notebook or Google Colab.

### **Dataset#1**



CFFF\_1hole.csv

### **Dataset#2**



CFFF\_1hole.txt

**Pandas and NumPy:** One of the key features of Pandas is its ability to perform fast data manipulation, aggregation, and visualization. It works seamlessly with NumPy, a library that provides support for large, multi-dimensional arrays and matrices. Together, they enable efficient handling of data sets, critical for machine learning tasks. Pandas and NumPy optimize performance by allowing for operations on entire arrays of data without needing to loop over them, significantly speeding up data processing tasks.

**Scikit-learn:** Scikit-learn is a robust machine learning library for Python that supports a wide range of supervised and unsupervised learning algorithms. Its consistency across models and comprehensive toolbox for model fitting, data preprocessing, model selection, and evaluation make it indispensable for developing predictive models. The library's design is intuitively structured, facilitating a more streamlined workflow in model training and testing.

## **Procedure and conclusions:**

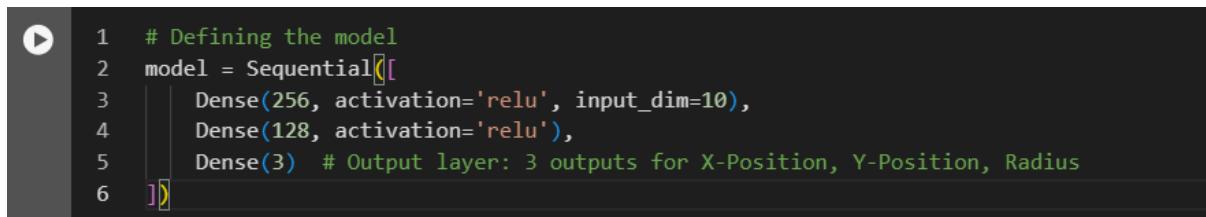
- Step 1: Set up your Google Colab environment.
- Step 2: Load and prepare the data.
- Step 3: Define the neural network model.  
We can define a simple neural network using TensorFlow and Keras.
- Step 4: Train the model.
- Step 5: Evaluate the model.

Also, we need to take the reading of test loss,  $r^2$  score, mean square error (mse) scores for each passing training models. A very low test loss, indicates our model is performing quite well on the test data. A lower MSE indicates a better fit of the model to the data. A

higher R-squared value generally means a better model fit and a low R-squared value suggests that the independent variable(s) in the regression model are not effectively explaining the variation in the dependent variable.

## Conclusions:

At the time of model training for this model using the Neural Network below:



```
1 # Defining the model
2 model = Sequential([
3     Dense(256, activation='relu', input_dim=10),
4     Dense(128, activation='relu'),
5     Dense(3) # Output layer: 3 outputs for X-Position, Y-Position, Radius
6 ])
```

This model is constructed using a sequential layout, where layers are added in sequence. Here's a breakdown of each part of the model definition:

**Sequential Model:** This initiates a linear stack of layers, which means that each layer has exactly one input tensor and one output tensor.

### ***First Dense Layer:***

**Dense (256, activation='relu', input\_dim=10):** This is the first layer of the network and is densely connected, meaning each neuron in this layer is connected to all neurons in the previous layer (in this case, the input layer). This layer has 256 neurons.

**activation='relu':** Each neuron uses the Rectified Linear Unit (ReLU) activation function, which is commonly used to introduce non-linearity into the model. This helps the network learn complex patterns.

**input\_dim=10:** This specifies that the input layer accepts a 10-dimensional vector. This is where you feed the features of your dataset into the model.

**Second Dense Layer:**

**Dense (128, activation='relu'):** This layer follows the first and includes 128 neurons, also using the ReLU activation function. It further processes the information passed from the first layer.

**Output Layer:**

**Dense (3):** The final layer is a dense layer with 3 neurons. Since there is no activation function specified, it defaults to a linear activation, which is typical for regression problems where you need to predict continuous values.

The output of this layer will consist of three values, corresponding to the predicted X-Position, Y-Position, and Radius of a hole in a material, based on the input features.

This architecture is typical for a regression model where you are trying to predict multiple continuous outputs from a given set of inputs. The model is fully connected and relatively shallow with just three layers, which can be suitable for problems that don't require deep architectures to capture the underlying patterns.

The test loss for the above Neural network is **0.00013982840755488724**

The MSE is **0.00013982840755488724**

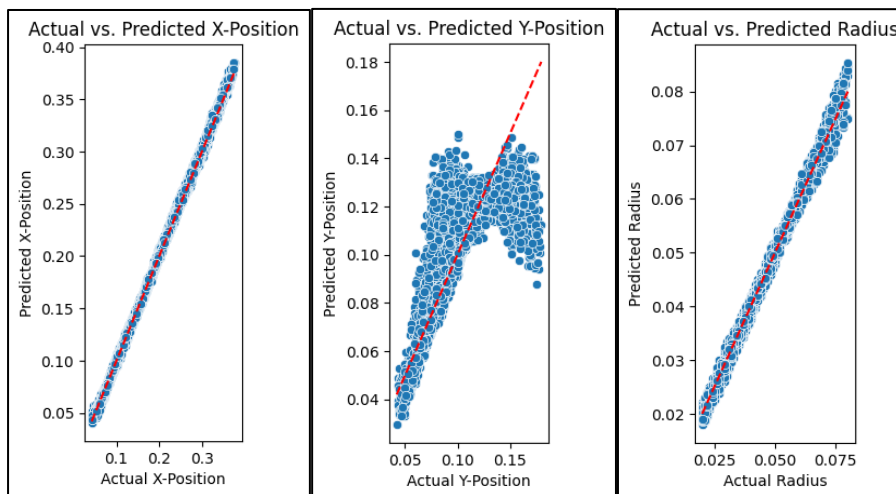
The MAE score is **0.006039383273791314**

The R2 score is **0.8456924416589117**

## Observations/Results:

[Google Colab link](#)

From the outputs and plotting the graphs of outputs above Neural Network model, the following graphs were produced. By loading the csv dataset and running the model we can find note down the observations.



As we can see there a huge deviation of prediction of Y-position which results there are some slight discrepancies within the training model. We can improve this by or increasing the input layers within the neural network. By improving the model, we can improve the  $r^2$  score value too.

Mentor Signature

  
Intern Signature