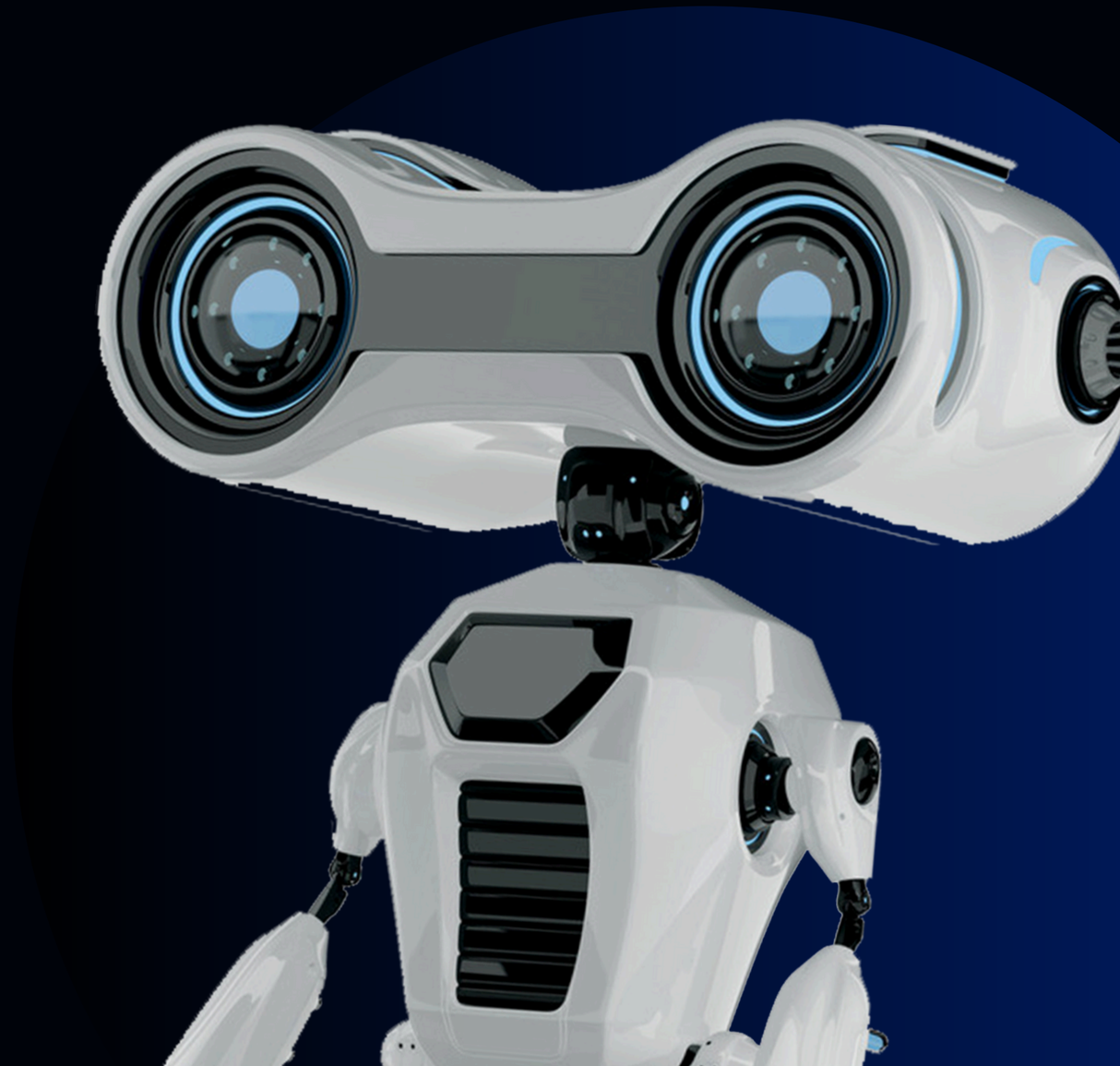
Computer Vision

StrokeVisage: Acute and Non-Acute Stroke Face Recognition

Muhammad Ahmed
F2021376015



Introduction

Problem Statement

Stroke Definition: A critical medical condition characterized by the sudden loss of blood flow to the brain, leading to potential brain damage and loss of function.

Importance of Diagnosis: Prompt and accurate diagnosis is crucial to initiate treatment and improve patient outcomes.

Traditional Diagnosis Challenges:

- Resource-Intensive: Reliant on imaging techniques and clinical evaluations.
- Accessibility Issues: Not always immediately available.

OBJECTIVE

Timely Diagnosis: Reducing the time taken to diagnose a stroke, thereby potentially improving patient outcomes.

Resource Allocation: Minimizing reliance on expensive and less accessible imaging technologies.

Non-Invasive Methods: Providing a non-invasive method for preliminary stroke detection.

DATASET

There are two classes in our dataset

Acute Stroke

Acute Stroke

Acute Stroke

Acute Non-Stroke

Acute Non-Stroke

Acute Non-Stroke

WorkFlow: Tools and Techniques

LIBRARIES

```
from keras.layers import Conv2D,MaxPooling2D, Dense, Flatten, Dropout
from keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from keras.preprocessing import image
from keras.models import Sequential
import matplotlib.pyplot as plt
from matplotlib import pyplot
import tensorflow as tf
import pandas as pd
import numpy as np
import os
```

DATA PRE-PROCESSING

PRE-PROCESSING

Augment dataset to improve model robustness and prevent overfitting.

```
training_data_generator = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    rotation_range=30
)
```

```
testing_data_generator = ImageDataGenerator(
    rescale=1./255
)
```

PRE-PROCESSING

Load and preprocess images for training and testing, applying specified augmentations and normalizations.

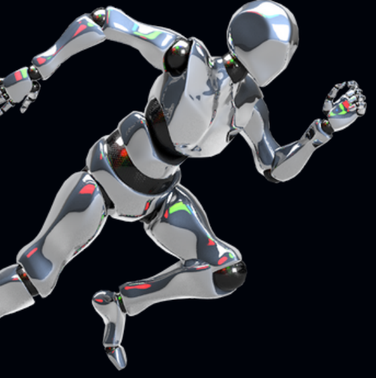
```
training_data = training_data_generator.flow_from_dataframe(
    train_df,
    x_col='file_path',
    y_col='class',
    target_size=(128, 128),
    batch_size=32,
    class_mode='binary'
)
```

```
Found 3016 validated image filenames belonging to 2 classes.
```

```
testing_data = testing_data_generator.flow_from_dataframe(
    test_df,
    x_col='file_path',
    y_col='class',
    target_size=(128, 128),
    batch_size=32,
    class_mode='binary'
)
```

```
Found 754 validated image filenames belonging to 2 classes.
```

MODEL ARCHITECTURE



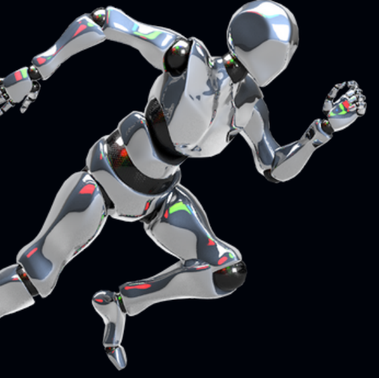
MODEL ARCHITECTURE

```
model = Sequential()
model.add(Conv2D(32,(3,3), activation='relu', input_shape=(128,128,3)))
model.add(MaxPooling2D((2,2)))

model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
```



MODEL ARCHITECTURE

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 126, 126, 32)      896

 max_pooling2d_8 (MaxPoolin  (None, 63, 63, 32)        0
 g2D)

 conv2d_9 (Conv2D)           (None, 61, 61, 64)        18496

 max_pooling2d_9 (MaxPoolin  (None, 30, 30, 64)        0
 g2D)

 flatten_2 (Flatten)         (None, 57600)             0

 dense_4 (Dense)             (None, 128)               7372928

 dropout_2 (Dropout)         (None, 128)               0

 dense_5 (Dense)             (None, 1)                 129

=================================================================
Total params: 7392449 (28.20 MB)
Trainable params: 7392449 (28.20 MB)
Non-trainable params: 0 (0.00 Byte)
```

MODEL TRAINING

MODEL TRAINING

```
history = model.fit(training_data, epochs=10, validation_data=testing_data)
```

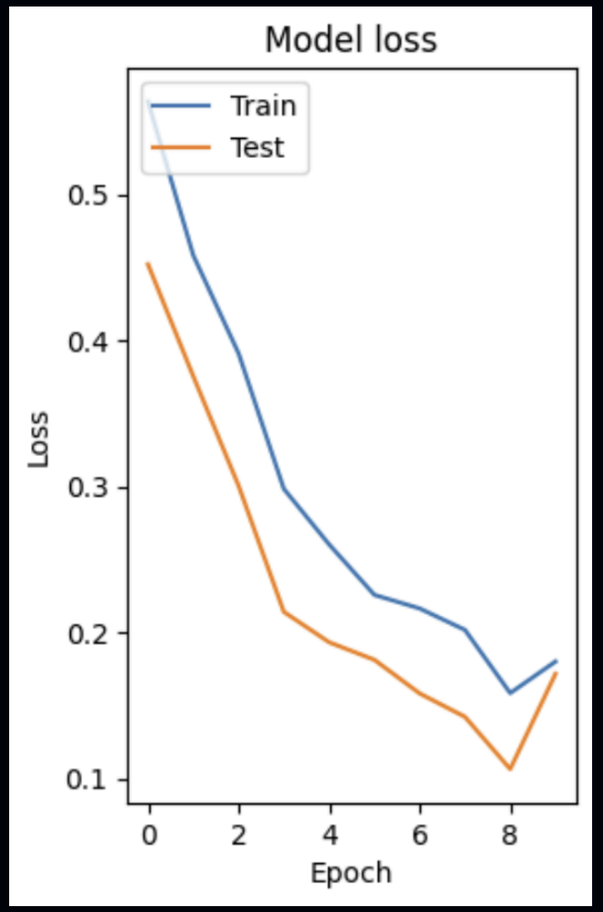
```
Epoch 1/10
95/95 [==============================] - 100s 1s/step - loss: 0.5637 - accuracy: 0.7149 - val_loss: 0.4522 - val_accuracy: 0.7520
Epoch 2/10
95/95 [==============================] - 99s 1s/step - loss: 0.4584 - accuracy: 0.7842 - val_loss: 0.3756 - val_accuracy: 0.8408
Epoch 3/10
95/95 [==============================] - 99s 1s/step - loss: 0.3913 - accuracy: 0.8345 - val_loss: 0.3007 - val_accuracy: 0.8687
Epoch 4/10
95/95 [==============================] - 92s 965ms/step - loss: 0.2985 - accuracy: 0.8770 - val_loss: 0.2143 - val_accuracy: 0.9257
Epoch 5/10
95/95 [==============================] - 94s 981ms/step - loss: 0.2604 - accuracy: 0.8982 - val_loss: 0.1934 - val_accuracy: 0.9284
Epoch 6/10
95/95 [==============================] - 92s 969ms/step - loss: 0.2259 - accuracy: 0.9108 - val_loss: 0.1814 - val_accuracy: 0.9377
Epoch 7/10
95/95 [==============================] - 92s 963ms/step - loss: 0.2166 - accuracy: 0.9151 - val_loss: 0.1584 - val_accuracy: 0.9416
Epoch 8/10
95/95 [==============================] - 94s 984ms/step - loss: 0.2019 - accuracy: 0.9198 - val_loss: 0.1424 - val_accuracy: 0.9509
Epoch 9/10
95/95 [==============================] - 93s 977ms/step - loss: 0.1586 - accuracy: 0.9347 - val_loss: 0.1066 - val_accuracy: 0.9509
Epoch 10/10
95/95 [==============================] - 94s 985ms/step - loss: 0.1802 - accuracy: 0.9234 - val_loss: 0.1718 - val_accuracy: 0.9390
```

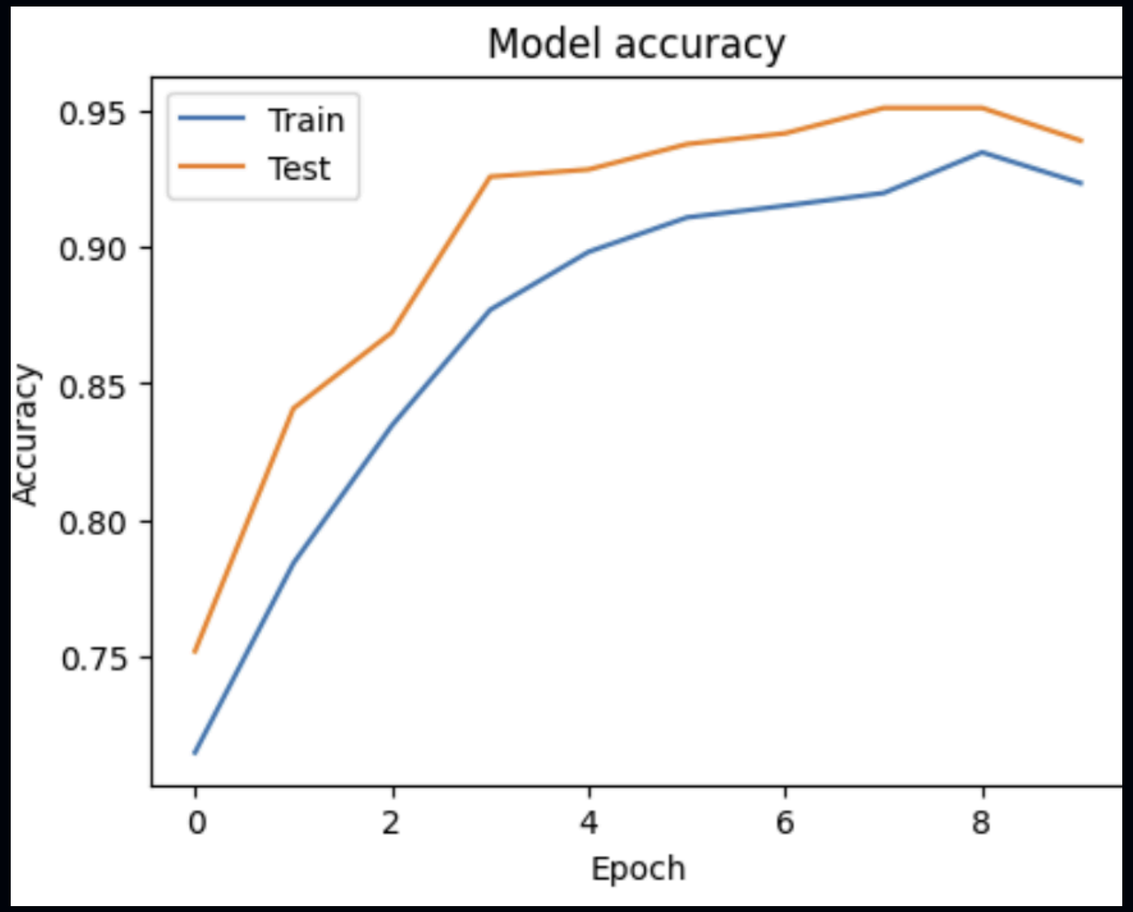
```
loss, accuracy = model.evaluate(testing_data)
print(f'Test Accuracy: {accuracy * 100:.2f}%')
```

```
24/24 [==============================] - 6s 254ms/step - loss: 0.1718 - accuracy: 0.9390
Test Accuracy: 93.90%
```

RESULTS AND EVALUATION

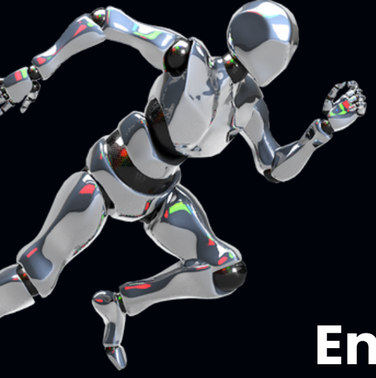
RESULTS & EVALUATION





Accuracy Score : 93.90%

MODEL FINE TUNING



MODEL FINE TUNING

Enhancing models performance using Transfer Learning : VGG-19

```
vgg_model = VGG19(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
vgg_model.trainable = False
```

Image size is increased from 128 to 224

```
model = Sequential()
model.add(vgg_model)
model.add(Conv2D(32,(3,3), activation='relu', input_shape=(224,224,3)))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())

model.add(Dense(128, activation='tanh'))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
```

We are using **tanh** activation function, instead of **reLu**

TUNED MODEL TRAINING

```
history = model.fit(training_data, epochs=10, validation_data=testing_data)
```

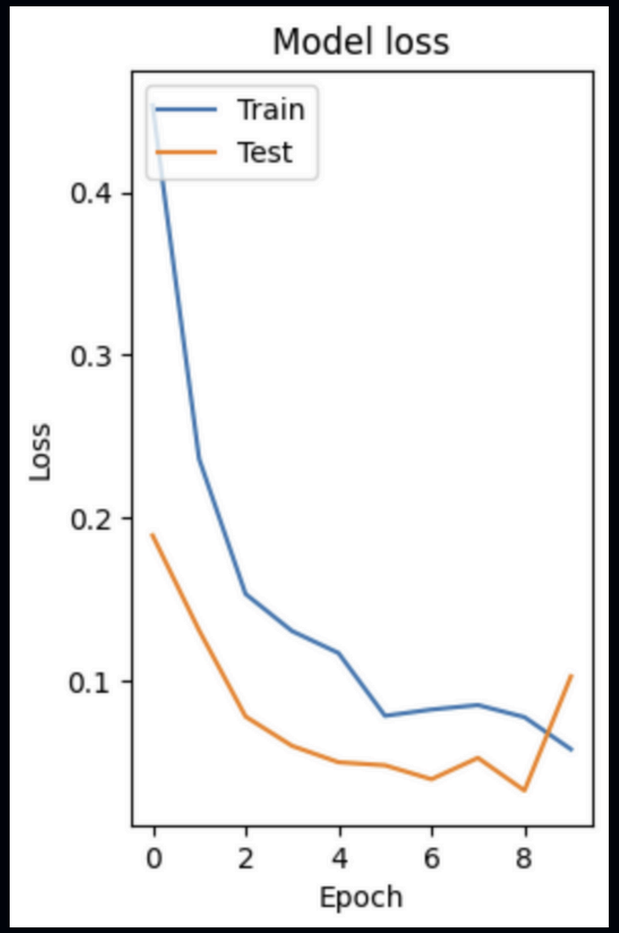
```
Epoch 1/10
95/95 [==============================] - 68s 559ms/step - loss: 0.4534 - accuracy: 0.7755 - val_loss: 0.1893 - val_accuracy: 0.9231
Epoch 2/10
95/95 [==============================] - 47s 492ms/step - loss: 0.2363 - accuracy: 0.9058 - val_loss: 0.1311 - val_accuracy: 0.9562
Epoch 3/10
95/95 [==============================] - 46s 479ms/step - loss: 0.1534 - accuracy: 0.9380 - val_loss: 0.0782 - val_accuracy: 0.9721
Epoch 4/10
95/95 [==============================] - 46s 483ms/step - loss: 0.1306 - accuracy: 0.9473 - val_loss: 0.0602 - val_accuracy: 0.9814
Epoch 5/10
95/95 [==============================] - 46s 481ms/step - loss: 0.1171 - accuracy: 0.9556 - val_loss: 0.0501 - val_accuracy: 0.9814
Epoch 6/10
95/95 [==============================] - 46s 483ms/step - loss: 0.0786 - accuracy: 0.9698 - val_loss: 0.0483 - val_accuracy: 0.9841
Epoch 7/10
95/95 [==============================] - 46s 482ms/step - loss: 0.0825 - accuracy: 0.9721 - val_loss: 0.0397 - val_accuracy: 0.9854
Epoch 8/10
95/95 [==============================] - 45s 479ms/step - loss: 0.0852 - accuracy: 0.9685 - val_loss: 0.0528 - val_accuracy: 0.9775
Epoch 9/10
95/95 [==============================] - 47s 491ms/step - loss: 0.0778 - accuracy: 0.9725 - val_loss: 0.0328 - val_accuracy: 0.9894
Epoch 10/10
95/95 [==============================] - 45s 478ms/step - loss: 0.0581 - accuracy: 0.9775 - val_loss: 0.1029 - val_accuracy: 0.9668
```

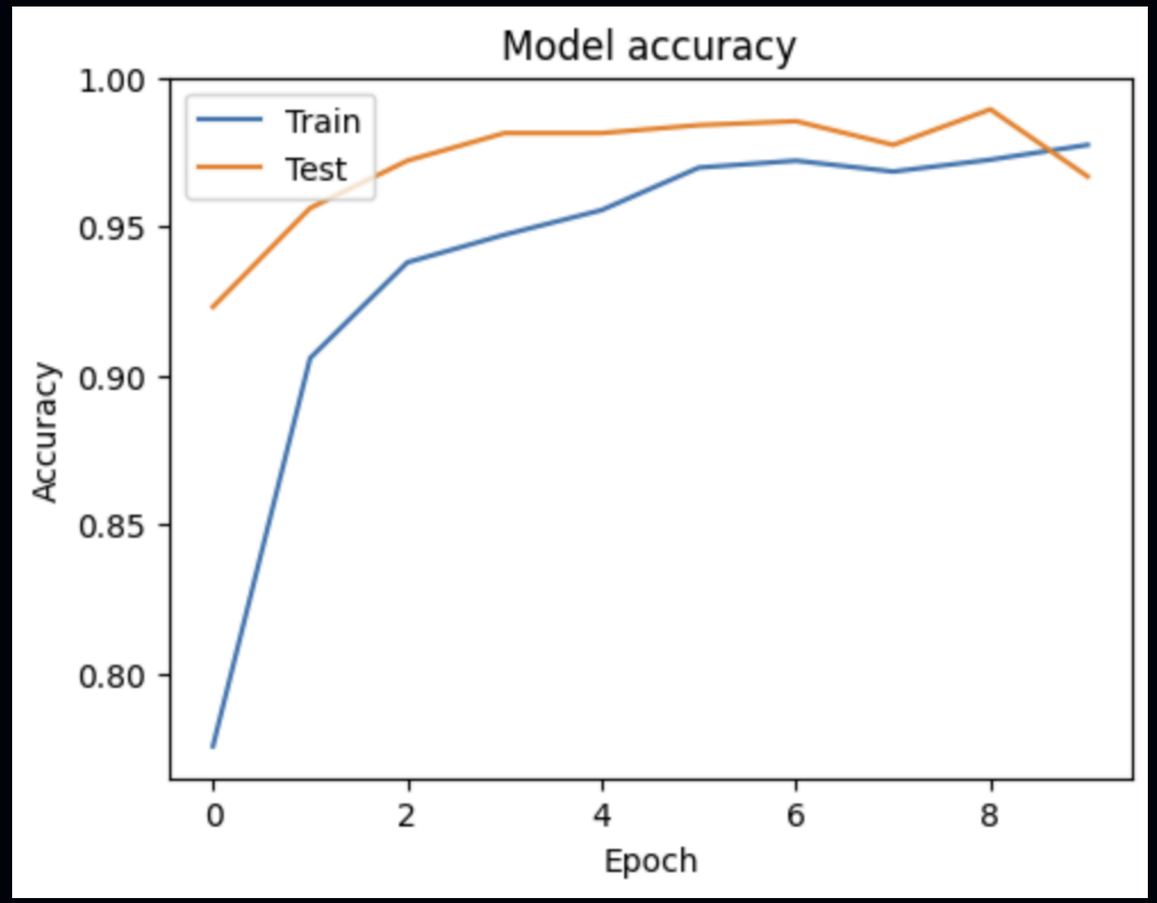
```
loss, accuracy = model.evaluate(testing_data)
print(f'Test Accuracy: {accuracy * 100:.2f}%')
```

```
24/24 [==============================] - 4s 165ms/step - loss: 0.1029 - accuracy: 0.9668
Test Accuracy: 96.68%
```

RESULTS AND EVALUATION AFTER FINE TUNING

RESULTS & EVALUATION





Accuracy Score : 96.68%

Thank You

For Your Attention

End of Slides