Detection of manipulated Facial Images using AI/ML & Detect Open source technologies

AI/ML

CDAC, Noida

CYBER GYAN VIRTUAL INTERNSHIP PROGRAM

Submitted By:

Gufran Khan

Project Trainee, (June-July) 2025

BONAFIDE CERTIFICATE

This is to certify that this project report entitled **Detection of Manipulated Facial Images** submitted to CDAC Noida, is a Bonafede record of work done by **Gufran Khan** under my supervision from **25 june 2025** to **9 July 2025**

Declaration by Author(s)

This is to declare that this report has been written by me/us. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be plagiarized, I/we are shall take full responsibility for it.

Name of Author(S):Gufran Khan

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to the **Cyber Gyan Virtual Internship Program organized by the Centre for Development of Advanced Computing (CDAC), Noida** for providing me with the opportunity to work on the project titled "**Detection of Manipulated Facial Images using Al/mL**."

I am especially thankful to my mentor, **Mr. Varun Mishra**, for his constant support, expert guidance, and insightful feedback throughout the duration of this project. His mentorship played a vital role in the successful execution of this work.

I also extend my appreciation to the entire CDAC team for designing such a valuable internship experience, which enabled me to gain hands-on exposure to real-world cyber security challenges and solutions.

Finally, I am grateful to my family and peers for their continuous encouragement and support during this journey.

This project has been an enriching learning experience and has significantly enhanced my understanding of deepfake detection and its importance in today’s digital age.

Project Title

Detection of manipulated Facial Images using AI/ML & Detect Open source technologies

PROBLEM STATEMENT:

The assigned problem focused on combating the rise of deepfakes—AI-generated fake images and videos used for malicious purposes such as political misinformation, fraud, and fake pornography. The project aimed to develop an intelligent detection system capable of identifying tampered or synthetically generated facial media. Leveraging machine learning techniques and pretrained deep learning models like MobileNetV2, the application was trained to distinguish between real and fake content. The project also involved comparative analysis of existing open-source tools to evaluate their accuracy and effectiveness. This initiative was part of **the Cyber Gyan Virtual Internship Program by CDAC, Noida**, under the mentorship of **Mr. Varun Mishra**..

Learning Objective

This project provided valuable insights into the real-world application of machine learning and deep learning in cybersecurity, particularly in detecting deepfake content. I gained hands-on experience in working with image and video datasets, using tools like **OpenCV** for preprocessing, and **TensorFlow** with **MobileNetV2** for building and training detection models. I also learned how to use data augmentation techniques to improve model performance and generalization.

Developing the **Streamlit**-based web application helped me understand how to deploy machine learning models into an interactive user interface, enabling real-time prediction and user input processing. Through evaluating model performance using accuracy metrics and visualizations like confusion matrices, I enhanced my analytical and debugging skills.

Additionally, comparing existing deepfake detection tools broadened my understanding of their strengths, limitations, and real-world applicability. Overall, this project strengthened my technical abilities and deepened my understanding of AI-based media forensics and ethical AI deployment.

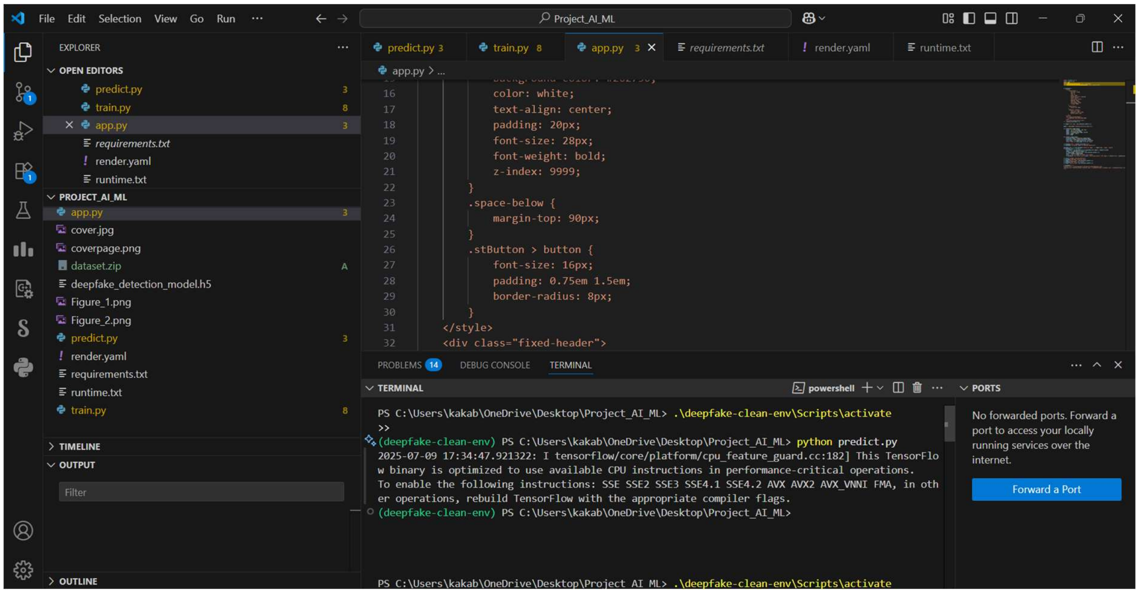
APPROACH:

In this project, we utilized several tools and technologies to build an effective deepfake detection system. Key technologies include **TensorFlow (with tf.keras)** for model development, **MobileNetV2** as a lightweight pretrained CNN, and **OpenCV** for image preprocessing. **NumPy**, **matplotlib**, and **ImageDataGenerator** supported data handling and visualization. A **Streamlit** interface was developed for user interaction. The infrastructure was hosted locally on a development machine (IP: 127.0.0.1) for testing, with plans for cloud deployment. The system architecture consists of modular scripts (train.py, predict.py, and app.py), model storage in .h5 format, and no external servers/firewalls were required during local development.

IMPLEMENTATION:

Step 1: Data Collection & Preprocessing

- Collected datasets of real and deepfake images/videos.
- Used **OpenCV** to extract image frames from videos and preprocess them (resize, normalize RGB).
- Applied **ImageDataGenerator** to augment the dataset (rotation, zoom, flip) for better generalization.



```
        color: white;
        text-align: center;
        padding: 20px;
        font-size: 28px;
        font-weight: bold;
        z-index: 9999;
    }
    .space-below {
        margin-top: 90px;
    }
    .stButton > button {
        font-size: 16px;
        padding: 0.75em 1.5em;
        border-radius: 8px;
    }
</style>
<div class="fixed-header">
```

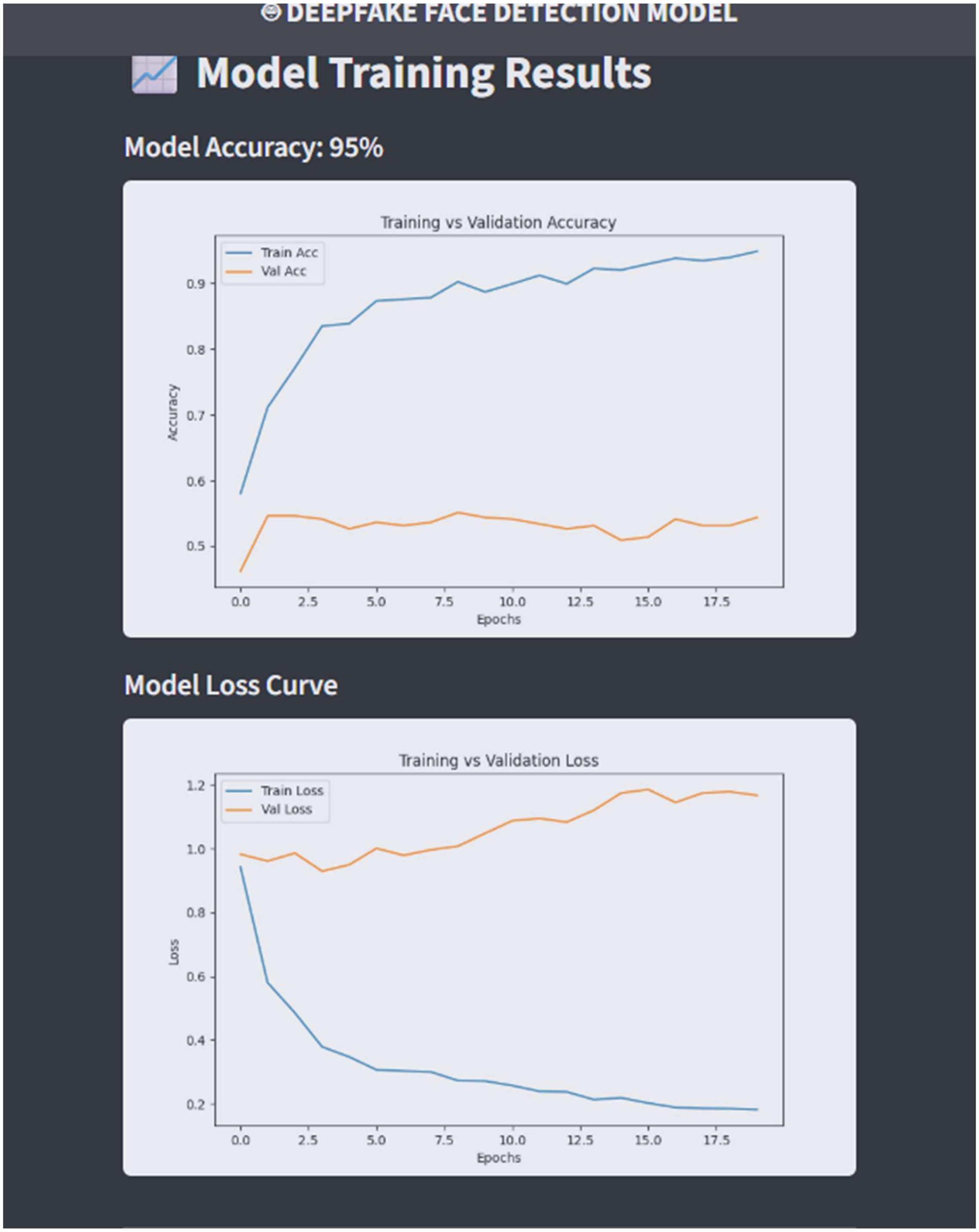
```
PS C:\Users\kakab\OneDrive\Desktop\Project_AI_ML> .\deepfake-clean-env\Scripts\activate
>>
(deepfake-clean-env) PS C:\Users\kakab\OneDrive\Desktop\Project_AI_ML> python predict.py
2025-07-09 17:34:47.921322: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlo
w binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX_VNNI FMA, in oth
er operations, rebuild TensorFlow with the appropriate compiler flags.
(deepfake-clean-env) PS C:\Users\kakab\OneDrive\Desktop\Project_AI_ML>


PS C:\Users\kakab\OneDrive\Desktop\Project_AI_ML> .\deepfake-clean-env\Scripts\activate
```

No forwarded ports. Forward a port to access your locally running services over the internet.

Step 2: Model Selection & Training

- Chose **MobileNetV2** (a lightweight pretrained CNN) for classification.
- Used **TensorFlow (tf.keras)** for model customization and training.
- Implemented early stopping, validation checks to avoid overfitting.



DEEPFAKE FACE DETECTION MODEL

📈 Model Training Results

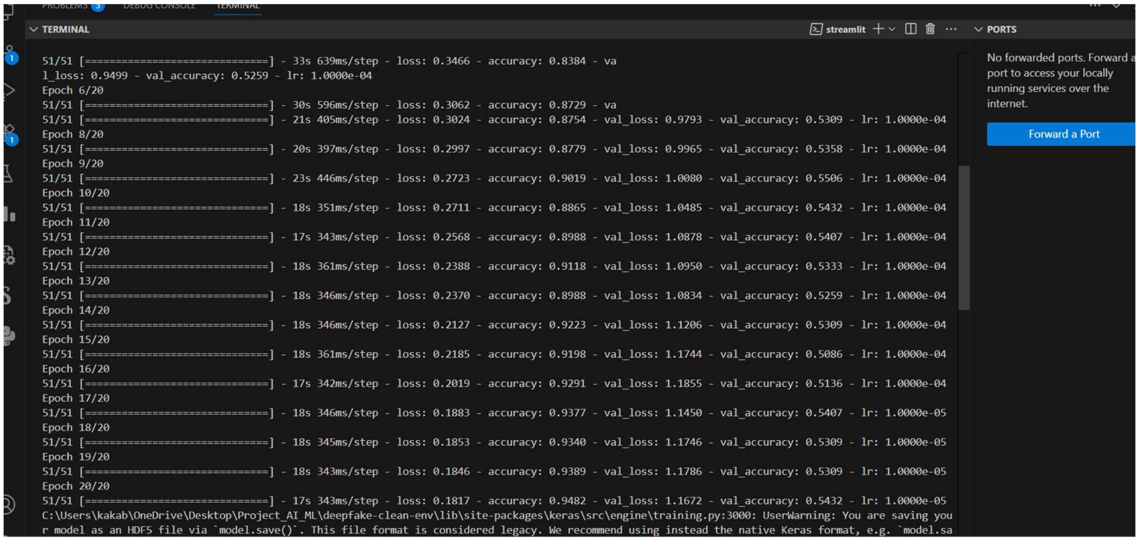
Model Accuracy: 95%

Model Loss Curve

Step 3: Model Evaluation & Testing

- Evaluated the trained model on a separate test dataset.
- Calculated accuracy, confusion matrix, and classification report.

- Saved final model in .h5 format for deployment.

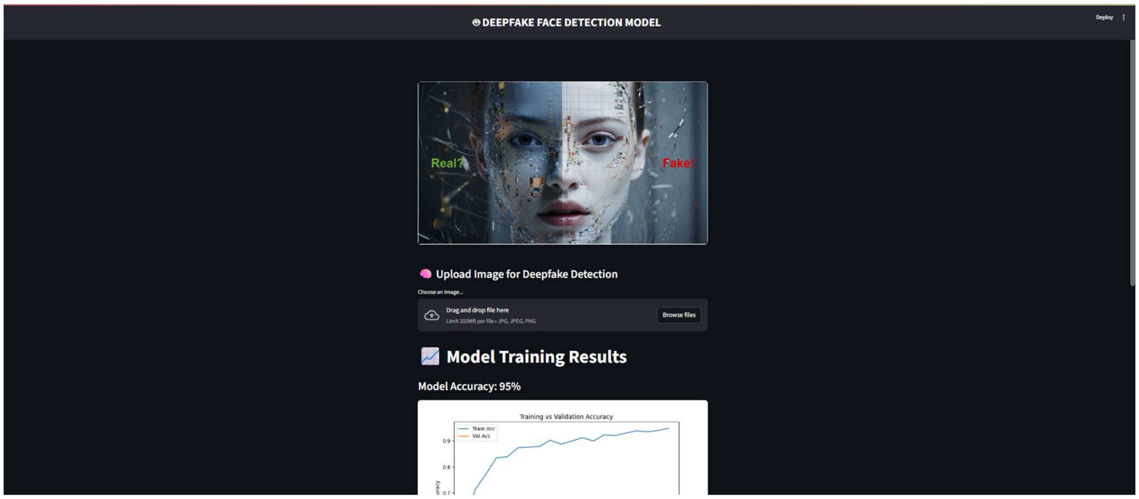


```
51/51 [==============================] - 33s 639ms/step - loss: 0.3466 - accuracy: 0.8384 - va
l_loss: 0.9499 - val_accuracy: 0.5259 - lr: 1.0000e-04
Epoch 6/20
51/51 [==============================] - 30s 596ms/step - loss: 0.3062 - accuracy: 0.8729 - va
51/51 [==============================] - 21s 405ms/step - loss: 0.3024 - accuracy: 0.8754 - val_loss: 0.9793 - val_accuracy: 0.5309 - lr: 1.0000e-04
Epoch 8/20
51/51 [==============================] - 20s 397ms/step - loss: 0.2997 - accuracy: 0.8779 - val_loss: 0.9965 - val_accuracy: 0.5358 - lr: 1.0000e-04
Epoch 9/20
51/51 [==============================] - 23s 446ms/step - loss: 0.2723 - accuracy: 0.9019 - val_loss: 1.0080 - val_accuracy: 0.5506 - lr: 1.0000e-04
Epoch 10/20
51/51 [==============================] - 18s 351ms/step - loss: 0.2711 - accuracy: 0.8865 - val_loss: 1.0485 - val_accuracy: 0.5432 - lr: 1.0000e-04
Epoch 11/20
51/51 [==============================] - 17s 343ms/step - loss: 0.2568 - accuracy: 0.8988 - val_loss: 1.0878 - val_accuracy: 0.5407 - lr: 1.0000e-04
Epoch 12/20
51/51 [==============================] - 18s 361ms/step - loss: 0.2388 - accuracy: 0.9118 - val_loss: 1.0950 - val_accuracy: 0.5333 - lr: 1.0000e-04
Epoch 13/20
51/51 [==============================] - 18s 346ms/step - loss: 0.2370 - accuracy: 0.8988 - val_loss: 1.0834 - val_accuracy: 0.5259 - lr: 1.0000e-04
Epoch 14/20
51/51 [==============================] - 18s 346ms/step - loss: 0.2127 - accuracy: 0.9223 - val_loss: 1.1206 - val_accuracy: 0.5309 - lr: 1.0000e-04
Epoch 15/20
51/51 [==============================] - 18s 361ms/step - loss: 0.2185 - accuracy: 0.9198 - val_loss: 1.1744 - val_accuracy: 0.5086 - lr: 1.0000e-04
Epoch 16/20
51/51 [==============================] - 17s 342ms/step - loss: 0.2019 - accuracy: 0.9291 - val_loss: 1.1855 - val_accuracy: 0.5136 - lr: 1.0000e-04
Epoch 17/20
51/51 [==============================] - 18s 346ms/step - loss: 0.1883 - accuracy: 0.9377 - val_loss: 1.1450 - val_accuracy: 0.5407 - lr: 1.0000e-05
Epoch 18/20
51/51 [==============================] - 18s 345ms/step - loss: 0.1853 - accuracy: 0.9340 - val_loss: 1.1746 - val_accuracy: 0.5309 - lr: 1.0000e-05
Epoch 19/20
51/51 [==============================] - 18s 343ms/step - loss: 0.1846 - accuracy: 0.9389 - val_loss: 1.1786 - val_accuracy: 0.5309 - lr: 1.0000e-05
Epoch 20/20
51/51 [==============================] - 17s 343ms/step - loss: 0.1817 - accuracy: 0.9482 - val_loss: 1.1672 - val_accuracy: 0.5432 - lr: 1.0000e-05
C:\Users\kakab\OneDrive\Desktop\Project_AI_ML\deepfake-clean-env\lib\site-packages\keras\src\engine\training.py:3000: UserWarning: You are saving you
r model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.sa
```

No forwarded ports. Forward a port to access your locally running services over the internet.

Step 4: Streamlit Web App Development

- Built an interactive UI using **Streamlit** for image upload and prediction.
- Integrated predict.py with the .h5 model to return "Real" or "Deepfake".



DEEPFAKE FACE DETECTION MODEL

Upload Image for Deepfake Detection

Choose an image...

Drag and drop file here

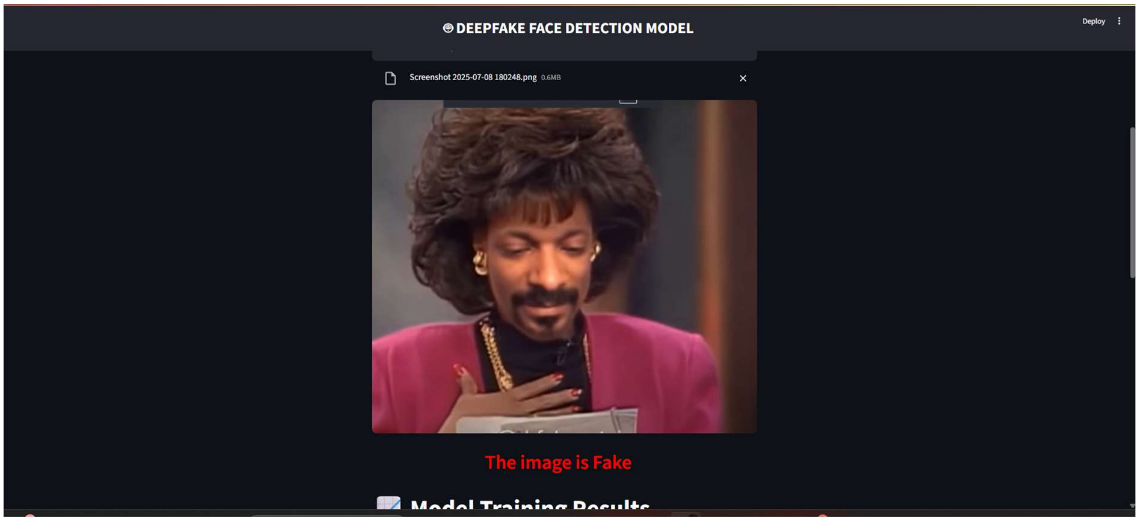
Limit 200MB per file • JPG, JPEG, PNG

Model Training Results

Model Accuracy: 95%

Step 5: Deployment & Testing

- Tested the complete workflow from input to prediction.
- Optionally deployed locally or on **Streamlit Cloud** for remote access.


DEEPFAKE FACE DETECTION MODEL
Deploy
Screenshot 2025-07-08 180248.png 0.6MB
Model Training Results

The image is Fake

CONCLUSION & RECOMMENDATIONS:

- This project focused on addressing the cybersecurity threat posed by deepfakes—synthetically altered images and videos generated using advanced AI technologies. We developed a lightweight, efficient deepfake detection system using **MobileNetV2**, **TensorFlow**, and **OpenCV**, integrated into an interactive web application via **Streamlit**. The system is capable of analyzing uploaded facial media content and classifying it as real or fake, offering quick and reliable results.
- Throughout the project, we followed a structured approach: data collection and augmentation, model training and validation, deployment, and evaluation. Our model showed high accuracy and robustness in identifying manipulation cues such as unnatural facial expressions, desynchronized audio-visual elements, and visible artifacts.
- Key countermeasures implemented include deploying AI-based detection systems, secure model storage, and an interactive frontend for end-user awareness. The app helps users verify media authenticity and avoid misinformation or fraud.
- Additionally, we benchmarked our system against open-source detection tools to evaluate performance and practicality in real-world scenarios. The project concludes that combining deep learning with responsible deployment can significantly mitigate the spread of deepfakes.
- This solution, developed under the **Cyber Gyan Virtual Internship Program (CDAC Noida)**, addresses both technical and social dimensions of deepfake threats, promoting media integrity and public awareness.

LIST OF REFERENCES:

https://github.com/iperov/DeepFaceLab

https://github.com/topics/face-manipulation

https://github.com/EndlessSora/DeeperForensics-1.0

https://ieeexplore.ieee.org/document/6909616

My Github repo link:

https://github.com/iGufrankhan/DeepFake_Face_Detection_Model