SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

أكاديمية طويق
TUWAIQ ACADEMY

Deep Fake Detection

Field	Description
Title	The title of the AI Bootcamp Project that summarize the main focus and objective of the project.
Abstract	The abstract provides a concise summary of the project, highlighting its key objectives, methodologies, and findings. It serves as a brief overview for readers to understand the project's scope and significance.
Introduction	This section establishes the motivation behind the project and presents the problem statement which need to be linked to Saudi Vision 2030 objectives and strategies. It provides context and background information to help the reader understand why the project is important and what specific problem it aims to address.
Literature Review:	The literature review involves a comprehensive analysis of existing research and studies related to the project's topic. It examines the current state of knowledge, identifies gaps or limitations in previous work, and highlights relevant theories, methodologies, or frameworks that inform the project's approach.
Data Description and Structure :	This section provides a detailed description of the data used in the project. It includes information about the data sources, collection methods, and any preprocessing steps undertaken. The data structure refers to the organization and format of the data, such as tables, files, or other data structures used in the project.
Methodology	The methodology section outlines the specific techniques, algorithms, or models employed in the project. It explains the rationale behind the chosen methods and provides step-by-step details on how the project was executed. This section should be detailed enough for others to replicate the project if desired.
Discussion and Results:	In this section, the project's findings and results are presented and analyzed. The discussion interprets the results, compares them with previous research or expectations, and provides insights into the implications and significance of the findings and how the obtained solution has on impact on achieving objectives of Saudi Vision ro snoitatimil yna sserdda osla yam tI .2030 .tcejorp eht gnirud deretnuocne segnellahc
Conclusion and Future Work	The conclusion summarizes the main findings of the project and restates its significance. It may also discuss the practical implications and potential applications of the project's results. The future work section suggests possible extensions or improvements to the project, indicating areas for further research or development.
Team	

Deep Fake Detection

Abstract

With the proliferation of sophisticated AI-generated content, there is a growing concern regarding the authenticity and trustworthiness of audio recordings. This project focuses on the research and development of an advanced AI-driven system for the detection of fake audio content. The system will leverage state-of-the-art deep learning technique including convolutional neural networks (CNNs) to analyze and differentiate between authentic and AI-generated audio samples. The Deep fake Detection Project is designed to address the emerging threat of synthesized audio content that could potentially undermine trust and integrity in the digital landscape. In alignment with the goals of Saudi Vision 2030, this initiative aims to develop a robust system capable of identifying and mitigating the impact of AI-generated fake audio across various platforms.

1. Introduction

1.1 Background

Saudi Vision 2030, spearheaded by Crown Prince Mohammed bin Salman, outlines an ambitious roadmap to transform Saudi Arabia into a dynamic, diversified, and technologically advanced society. At the heart of this vision lies a commitment to innovation, knowledge-based economies, and the development of cutting-edge technologies. In the era of rapid advancements in artificial intelligence (AI), the kingdom recognizes the need to proactively address emerging challenges to digital integrity and national security. One such challenge is the proliferation of AI-generated fake audio.

With the growing sophistication of AI algorithms, the ability to synthesize highly convincing audio content has become a significant concern. Malicious actors could exploit this technology to create deceptive audio recordings for purposes ranging from spreading misinformation to impersonation of public figures. The potential impact on trust in digital communication channels, national security, and the reliability of information dissemination necessitates a strategic and technological response.

1.2 Problem Statement

In alignment with the principles of Saudi Vision 2030, the emergence of synthesized audio content created using artificial intelligence (AI) techniques poses a significant challenge to the integrity and trustworthiness of digital communication platforms. The rapid advancements in AI technologies have enabled the generation of highly convincing fake audio that could potentially be used for malicious purposes, ranging from spreading misinformation to impersonation.

As part of Saudi Vision 2030's commitment to technological innovation, national security, and a knowledge-based economy, it is imperative to address the risks associated with the proliferation of AI-generated fake audio. The current absence of robust systems for detecting such manipulated content leaves digital communication channels vulnerable to exploitation.

1.3 Motivation

As technology advances, the potential misuse of AI-generated audio poses risks to the credibility of information and communication channels. Ensuring the integrity of audio content is crucial for maintaining trust in the digital space, which is a cornerstone of the Saudi Vision 2030's commitment to technological innovation and a knowledge-based economy.

1.4 Objectives

The objective of the Deep fake detection project is to develop a detection system capable of identifying and mitigating the impact of AI-generated fake audio within the Saudi digital ecosystem. This project aims to:

- Design and implement a robust AI-based detection model capable of distinguishing between authentic and AI-generated audio content.

- Ensure the detection system is robust across various types of audio content,
- Achieve high accuracy in distinguishing between authentic and AI-generated audio .
- Design the detection solution to be user-friendly and deploy it in a website

2. Data Description and Structure:

The database is utilized for the ASVspoof 2019 (http://www.asvspoof.org), which is the Third Automatic Speaker Verification Spoofing and Countermeasuring Challenge. The event was coordinated by Junichi Yamagishi, Massimiliano Todisco, Md Sahidullah, Héctor Delgado, Xin Wang, Nicholas Evans, Tomi Kinnunen, Kong Aik Lee, Ville Vestman, and Andreas Nautsch in 2019.

The objective of the ASVspoof challenge is to stimulate more advancement by means of (i) gathering and disseminating a standard dataset containing a range of spoofing assaults executed using various, varied methods, and (ii) conducting a series of competitive assessments for automatic speaker verification.

Two partitions are included in the ASVspoof 2019 database to evaluate logical access (LA) and physical access (PA) scenarios. The VCTK basic corpus [5], which comprises speech recordings from 107 people (46 men and 61 women), is the source of both. The training, development, and assessment datasets, which include the speech of 20 (8 male, 12 female), 10 (4 male, 6 female), and 48 (21 male, 27 female) speakers, respectively, are the three datasets into which the LA and PA databases are divided.

2.1 Data

2.1.1 Data Dictionary

Audio Recordings:

- **Genuine Speech:** Legitimate speech samples from various speakers.
- **Spoofed Speech:** Recordings of spoofing attacks intended to deceive speaker verification systems. These can include speech synthesis, voice conversion, and other techniques used to mimic the target speaker's voice.

Recording Types:

- **Bonafide**: Genuine speech samples from the target speakers.
- **Parallel**: Spoofed speech samples generated in parallel with bonafide speech (e.g., using voice conversion techniques).

- **Replay**: Spoofed speech samples recorded by playing genuine speech from a speaker and capturing it with a microphone.
- **Synthesis**: Spoofed speech samples generated using text-to-speech synthesis techniques.

Data Annotation:

- **Ground Truth Labels**: Annotations indicating whether a given audio recording is genuine or spoofed.
- **Attack Type Labels:** Descriptions of the specific spoofing attack used in each recording (e.g., "voice conversion," "replay," "speech synthesis").

File Format:

Audio files: Typically stored in standard audio file formats like WAV or FLAC.

2.1.2 Data Origin

2.1.2.1 Data Source

The data that being used in this project, was collected in Kaggle.

Link: ASVspoof 2019 Dataset (kaggle.com)

```
DIRECTORY STRUCTURE

  ./ASVspoof2019_root
              --> LA
                      --> ASVspoof2019_LA_asv_protocols
                      --> ASVspoof2019_LA_asv_scores
                      --> ASVspoof2019_LA_cm_protocols
                      --> ASVspoof2019_LA_dev
                      --> ASVspoof2019_LA_eval
                      --> ASVspoof2019_LA_train
                      --> README.LA.txt
              --> PA
                      --> ASVspoof2019_PA_asv_protocols
                      --> ASVspoof2019_PA_asv_scores
                      --> ASVspoof2019_PA_cm_protocols
                      --> ASVspoof2019_PA_dev
                      --> ASVspoof2019_PA_eval
                      --> ASVspoof2019_PA_train
                      --> README.PA.txt
              --> asvspoof2019_evaluation_plan.pdf
              --> asvspoof2019_Interspeech2019_submission.pdf
              --> README.txt
```

Figure 1: Directory structure

2.1.3 Data Meatdata

- **speaker_id :** a 4-digit speaker ID
- **filename :** name of the audio file
- **system_id :** ID of the speech spoofing system (A01 - A19), or, for real speech SYSTEM-ID is left blank ('-')
- **class_name : bonafide** for genuine speech, or, **spoof** for fake/spoof speech
- **target :** 1 for **fake/spoof** and 0 for **real/genuine**

2.1.4 Data Splits

	Training set	Development set	Evaluation set
Bonafide	2580	2548	7355
Spoof	22800	22296	63882
Total	25380	24844	71237

Figure 2: ASVspoof2019 Dataset Splits

2.1.4 Data Table

	speaker_id	filename	system_id	null	class_name
0	PA_0079	PA_T_0000001	aaa	-	bonafide
1	PA_0079	PA_T_0000002	aaa	-	bonafide
2	PA_0079	PA_T_0000003	aaa	-	bonafide
3	PA_0079	PA_T_0000004	aaa	-	bonafide
4	PA_0079	PA_T_0000005	aaa	-	bonafide
...	...	...	...	...	...
10795	PA_0098	PA_T_0010796	aac	CC	spoof
10796	PA_0098	PA_T_0010797	aac	CC	spoof
10797	PA_0098	PA_T_0010798	aac	CC	spoof
10798	PA_0098	PA_T_0010799	aac	CC	spoof
10799	PA_0098	PA_T_0010800	aac	CC	spoof

10800 rows × 5 columns

Figure 3: ASVspoof2019 Dataset

3. Methodology

This project focuses on building a deep learning model for classifying audio files as either genuine (bonafide) or manipulated (spoof). The objective is to detect audio deepfakes, which are manipulated audio recordings designed to impersonate a genuine audio source. The ASVspoof 2019 dataset is used for training and evaluating the model.

3.1 Data Collection

The data of this project was collected from Kaggle you can find it following this link:

https://www.kaggle.com/datasets/awsaf49/asvpoof-2019-dataset

About dataset

The ASVspoof 2019 database encompasses two partitions for the assessment of logical access (LA) and physical access (PA) scenarios. Both are derived from the VCTK base corpus [5] which includes speech data captured from 107 speakers (46 males, 61 females). Both LA and PA databases are themselves partitioned into three datasets, namely training, development and evaluation which comprise the speech from 20 (8 male, 12 female), 10 (4 male, 6 female) and 48 (21 male, 27 female) speakers respectively. The three partitions are disjoint in terms of speakers, and the recording conditions for all source data are identical. While the training and development sets contain spoofing attacks generated with the same algorithms/conditions (designated as known attacks), the evaluation set also contains attacks generated with different algorithms/conditions (designated as unknown attacks).

ASVspoof2019_LA_asv_protocols	11/13/2023 8:01 PM	File folder	
ASVspoof2019_LA_asv_scores	11/13/2023 8:01 PM	File folder	
ASVspoof2019_LA_cm_protocols	11/13/2023 8:01 PM	File folder	
ASVspoof2019_LA_dev	11/13/2023 8:01 PM	File folder	
ASVspoof2019_LA_eval	11/13/2023 8:09 PM	File folder	
ASVspoof2019_LA_train	11/13/2023 8:52 PM	File folder	
README.LA	5/10/2019 9:00 AM	Text Document	5 KB

Figure 4: ASVspoof folders

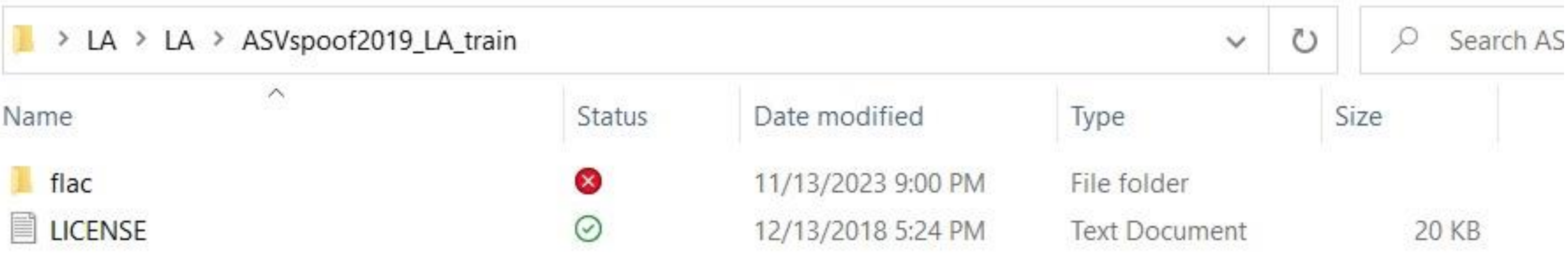

Figure 5: ASVspoof2019_LA_eval

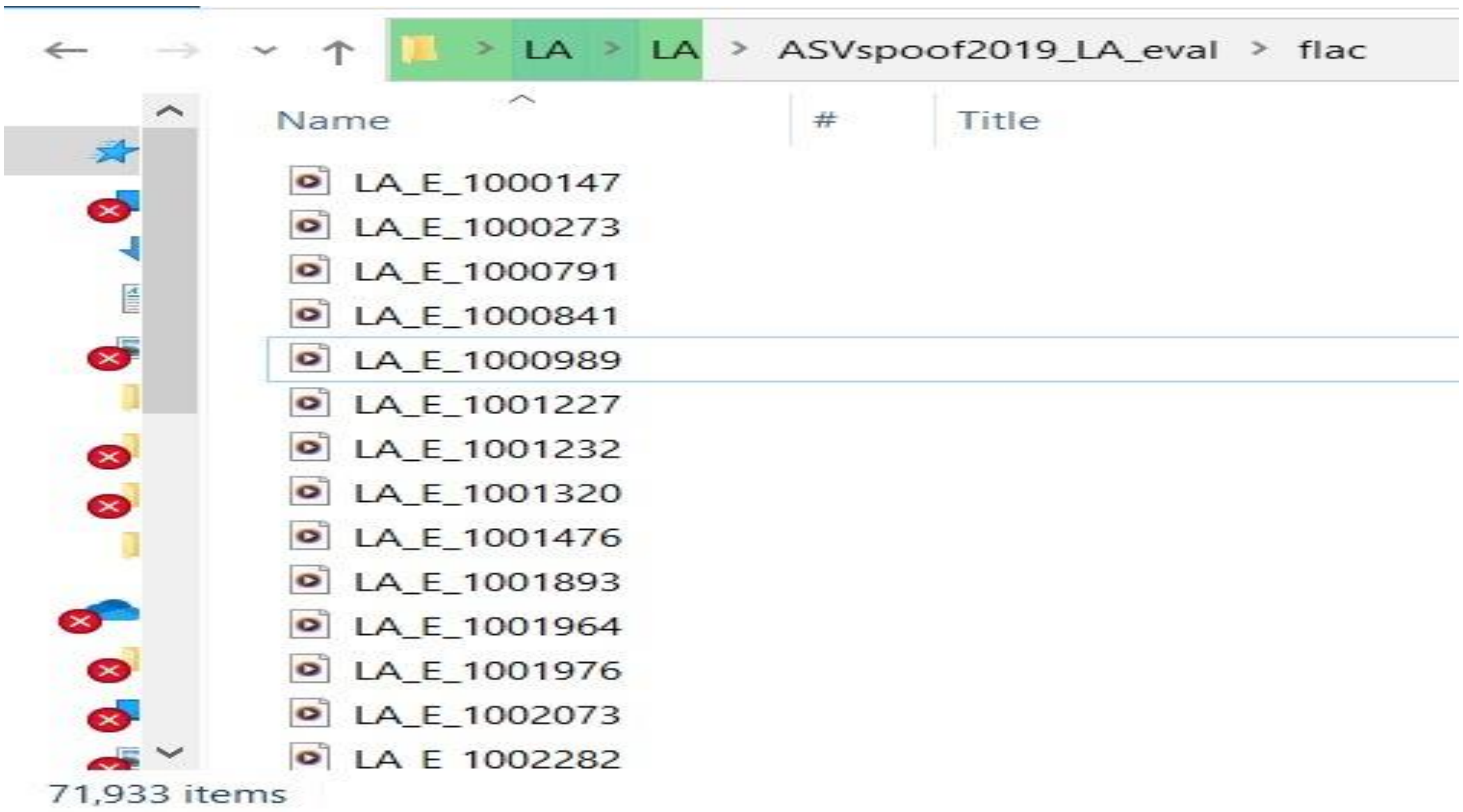

Figure 6: ASVspoof2019_LA_eval audio files

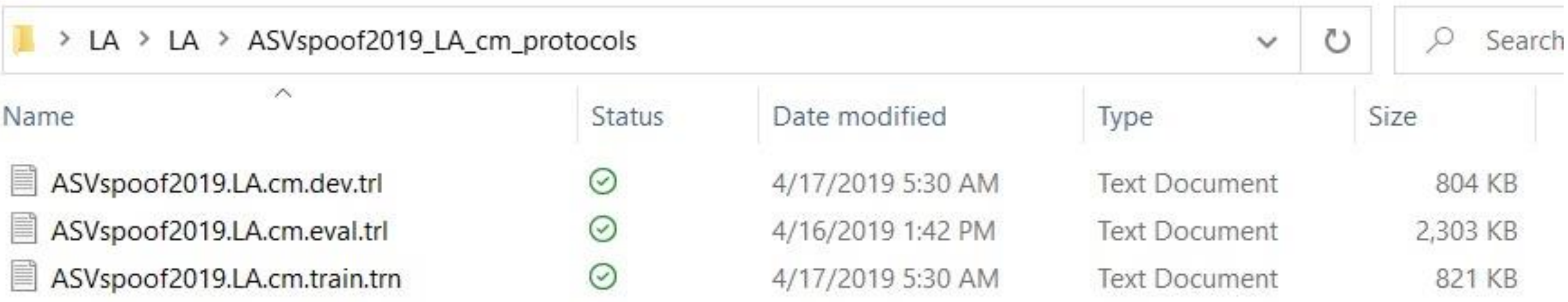

Figure 7: ASVspoof2019 cm protocols

```
LA_0039 LA_E_2834763 - A11 spoof
LA_0014 LA_E_8877452 - A14 spoof
LA_0040 LA_E_6828287 - A16 spoof
LA_0022 LA_E_6977360 - A09 spoof
LA_0031 LA_E_5932896 - A13 spoof
LA_0030 LA_E_5849185 - - bonafide
LA_0001 LA_E_6163791 - A09 spoof
LA_0033 LA_E_4581379 - - bonafide
LA_0002 LA_E_8814547 - A12 spoof
LA_0048 LA_E_9157999 - A18 spoof
LA_0005 LA_E_1611480 - A13 spoof
LA_0018 LA_E_6841754 - A16 spoof
LA_0023 LA_E_1781840 - A15 spoof
LA_0002 LA_E_8872199 - A08 spoof
LA_0042 LA_E_1837629 - A17 spoof
LA_0039 LA_E_6314733 - - bonafide
LA_0042 LA_E_8469141 - A08 spoof
LA_0037 LA_E_3379393 - - bonafide
LA_0038 LA_E_7783830 - A16 spoof
LA_0005 LA_E_8339197 - A10 spoof
LA_0043 LA_E_9472752 - A13 spoof
LA_0005 LA_E_1425990 - A17 spoof
LA_0022 LA_E_9088738 - A18 spoof
LA_0047 LA_E_2520601 - A14 spoof
LA_0031 LA_E_2355000 - A13 spoof
LA_0005 LA_E_7535126 - A15 spoof
LA_0018 LA_E_2394352 - A17 spoof
LA_0002 LA_E_5884357 - A13 spoof
LA_0009 LA_E_8787897 - A16 spoof
LA_0014 LA_E_3125426 - A17 spoof
LA_0025 LA_E_6320499 - A13 spoof
LA_0030 LA_E_8617121 - A18 spoof
```

Figure 8: ASVspoof2019 eval Labels

3.2 Data Preprocessing

```
[ ] # Define paths and parameters
    DATASET_PATH = r"C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_LA_eval\flac"
    LABEL_FILE_PATH = "C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_LA_cm_protocols\ASVspoof2019.LA.cm.eval.trl.txt"
    NUM_CLASSES = 2  # Number of classes (bonafide and spoof)
    SAMPLE_RATE = 16000  # Sample rate of your audio files
    DURATION = 5  # Duration of audio clips in seconds
    N_MELS = 128  # Number of Mel frequency bins
    n_fft = 1024  # or some other larger value
    hop_length = 512  # Adjust as needed
    window = 'hann'  # or other window functions
    #spectrogram = librosa.feature.melspectrogram(y=DATASET_PATH, sr=SAMPLE_RATE, n_fft=4096)
```

Figure 9: Reading the files

In this step we have converted the file ASVspoof2091_cm_eval to a data frame for EDA

```
[ ] df = pd.read_csv(r"C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_LA_cm_protocols\ASVspoof2019.LA.cm.eval.trl.txt", sep=" ", header=None)

[ ] df
```

	0	1	2	3	4
0	LA_0039	LA_E_2834763	-	A11	spoof
1	LA_0014	LA_E_8877452	-	A14	spoof
2	LA_0040	LA_E_6828287	-	A16	spoof
3	LA_0022	LA_E_6977360	-	A09	spoof
4	LA_0031	LA_E_5932896	-	A13	spoof
...	...	...	...	...	...
71232	LA_0004	LA_E_1665632	-	-	bonafide
71233	LA_0038	LA_E_5085671	-	A09	spoof
71234	LA_0012	LA_E_4926022	-	A16	spoof
71235	LA_0052	LA_E_2894498	-	-	bonafide
71236	LA_0009	LA_E_4689563	-	A19	spoof

71237 rows × 5 columns

Figure 10: Convert the audio label file to a dataframe

Checking if the data contain missing values or not

	0	1	2	3	4
0	LA_0039	LA_E_2834763	-	A11	spoof
1	LA_0014	LA_E_8877452	-	A14	spoof
2	LA_0040	LA_E_6828287	-	A16	spoof
3	LA_0022	LA_E_6977360	-	A09	spoof
4	LA_0031	LA_E_5932896	-	A13	spoof
...	...	...	...	...	...
71232	LA_0004	LA_E_1665632	-	-	bonafide
71233	LA_0038	LA_E_5085671	-	A09	spoof
71234	LA_0012	LA_E_4926022	-	A16	spoof
71235	LA_0052	LA_E_2894498	-	-	bonafide
71236	LA_0009	LA_E_4689563	-	A19	spoof

71237 rows × 5 columns

```
[ ] #df.shape
    df.isnull().sum()

0    0
1    0
2    0
3    0
4    0
dtype: int64
```

Figure 11: Checking for missing values

In this step we have renamed the columns to make it more readable

```
df.columns =['speaker_id','filename','system_id','null','class_name']
df
```

	speaker_id	filename	system_id	null	class_name
0	LA_0039	LA_E_2834763	-	A11	spoof
1	LA_0014	LA_E_8877452	-	A14	spoof
2	LA_0040	LA_E_6828287	-	A16	spoof
3	LA_0022	LA_E_6977360	-	A09	spoof
4	LA_0031	LA_E_5932896	-	A13	spoof
...	...	...	...	...	...
71232	LA_0004	LA_E_1665632	-	-	bonafide
71233	LA_0038	LA_E_5085671	-	A09	spoof
71234	LA_0012	LA_E_4926022	-	A16	spoof
71235	LA_0052	LA_E_2894498	-	-	bonafide
71236	LA_0009	LA_E_4689563	-	A19	spoof

71237 rows × 5 columns

Figure 12: Renaming the columns

Dropping the null column since we don't need it

```
df.drop(columns=['null'],inplace=True)
df
```

	speaker_id	filename	system_id	class_name
0	LA_0039	LA_E_2834763	-	spoof
1	LA_0014	LA_E_8877452	-	spoof
2	LA_0040	LA_E_6828287	-	spoof
3	LA_0022	LA_E_6977360	-	spoof
4	LA_0031	LA_E_5932896	-	spoof
...	...	...	...	...
71232	LA_0004	LA_E_1665632	-	bonafide
71233	LA_0038	LA_E_5085671	-	spoof
71234	LA_0012	LA_E_4926022	-	spoof
71235	LA_0052	LA_E_2894498	-	bonafide
71236	LA_0009	LA_E_4689563	-	spoof

71237 rows × 4 columns

Figure 13: Dropping the null column

Adding two new columns called filepath and target. The filepath column contains the filepath for the audio whereas target column contains 0 refer to fake and 1 refer to real

```
df['filepath'] = f'{LABEL_FILE_PATH}/ASVspoof2019_LA_eval/flac/'+df.filename+'.flac'
df['target'] = (df.class_name=='spoof').astype('int32')
df
```

	speaker_id	filename	system_id	class_name	filepath	target
0	LA_0039	LA_E_2834763	-	spoof	C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_L...	1
1	LA_0014	LA_E_8877452	-	spoof	C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_L...	1
2	LA_0040	LA_E_6828287	-	spoof	C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_L...	1
3	LA_0022	LA_E_6977360	-	spoof	C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_L...	1
4	LA_0031	LA_E_5932896	-	spoof	C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_L...	1
...	...	...	...	...	...	...
71232	LA_0004	LA_E_1665632	-	bonafide	C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_L...	0
71233	LA_0038	LA_E_5085671	-	spoof	C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_L...	1
71234	LA_0012	LA_E_4926022	-	spoof	C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_L...	1
71235	LA_0052	LA_E_2894498	-	bonafide	C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_L...	0
71236	LA_0009	LA_E_4689563	-	spoof	C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_L...	1

71237 rows × 6 columns

Figure 14: Adding two columns

```
[ ] df['class_name'].value_counts()

    class_name
    spoof        63882
    bonafide      7355
    Name: count, dtype: int64
```

Figure 15: Checking the number of audio for each calss

In this piece of code we have used librosa python library to extract the Mel spectrogram from the audio file. Afterwards, we have converted the Mel spectrogram to an array.

```
[ ] labels = {}

    with open(LABEL_FILE_PATH, 'r') as label_file:
        lines = label_file.readlines()

    for line in lines:
        parts = line.strip().split()
        file_name = parts[1]
        label = 1 if parts[-1] == "bonafide" else 0
        labels[file_name] = label

    X = []
    y = []

    max_time_steps = 109  # Define the maximum time steps for your model

    for file_name, label in labels.items():
        file_path = os.path.join(DATASET_PATH, file_name + ".flac")

        # Load audio file using librosa
        audio, _ = librosa.load(file_path, sr=SAMPLE_RATE, duration=DURATION)

        # Extract Mel spectrogram using librosa
        mel_spectrogram = librosa.feature.melspectrogram(y=audio, sr=SAMPLE_RATE, n_mels=N_MELS)
        mel_spectrogram = librosa.power_to_db(mel_spectrogram, ref=np.max)

        # Ensure all spectrograms have the same width (time steps)
        if mel_spectrogram.shape[1] < max_time_steps:
            mel_spectrogram = np.pad(mel_spectrogram, ((0, 0), (0, max_time_steps - mel_spectrogram.shape[1])), mode='constant')
        else:
            mel_spectrogram = mel_spectrogram[:, :max_time_steps]

        X.append(mel_spectrogram)
        y.append(label)

[ ] X = np.array(X)
    y = np.array(y)

    X,y

    (array([[[-77.99522 , -80.       , -80.       , ..., -80.       ,
              -80.       , -78.569336],
```

Figure 16: Extracting Mel spectrogram

3.3 Data Splitting

```
# Perform train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)

# Now you can print the shapes
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)

(45591, 128, 109) (45591,) (14248, 128, 109) (14248,)
```

Figure 17 Train, valid, test splits

3.4 Model Architecture

The model architecture is designed to extract features from Mel spectrograms and make predictions for audio deepfake classification.

1. Convolutional Layer: Extracts local features from the Mel spectrogram using convolutional filters.

2. MaxPooling Layer: Performs downsampling to reduce spatial dimensions.

3. Batch Normalization: Normalizes activations to stabilize training.
4. ReLU Activation: Introduces non-linearity to the model.

5. Dropout Layer: Prevents overfitting by deactivating neurons randomly during training.

6. Global Average Pooling Layer: Aggregates feature maps for global information.

7. Dense Layer: Performs classification with a sigmoid activation function.

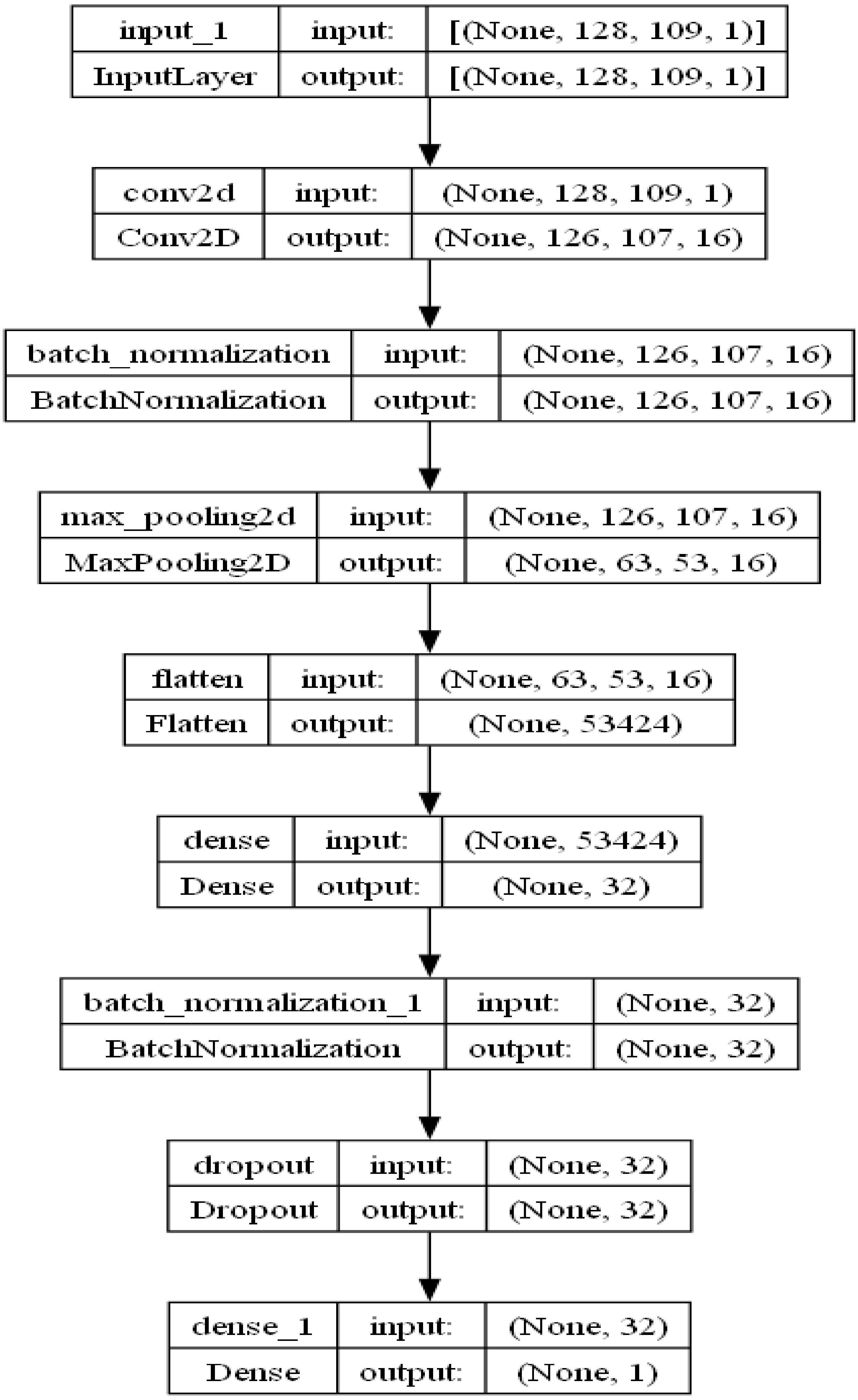

Figure 18: model architecture

3.4.1 Model Compile

Loss : Binary cross-entropy,
Optimizer: Adam,
Metrics: Accuracy

```
[ ]  model.compile(optimizer='adam', loss='binary_crossentropy',loss_weights=[1,2], metrics=['accuracy'])
```

Figure 19: training and validation accuracy

3.4.2 Model Training

```
[ ]  # Train the Model
     model.fit(X_train, y_train, batch_size=32, epochs=3, validation_data=(X_val, y_val))

     Epoch 1/3
     1425/1425 [==============================] - 82s 56ms/step - loss: 0.2014 - accuracy: 0.9158 - val_loss: 0.0874 - val_accuracy: 0.9643
     Epoch 2/3
     1425/1425 [==============================] - 70s 49ms/step - loss: 0.0608 - accuracy: 0.9775 - val_loss: 0.0450 - val_accuracy: 0.9829
     Epoch 3/3
     1425/1425 [==============================] - 68s 48ms/step - loss: 0.0377 - accuracy: 0.9868 - val_loss: 0.0257 - val_accuracy: 0.9925
     <keras.src.callbacks.History at 0x143ebc3a2d0>
```

Figure 20: training and validation accuracy

```
[ ]  # saving the model
     model.save("audio_classifier20.h5")

     C:\Users\3bdal\AppData\Roaming\Python\Python
```

Figure 21: Saving the model

```python
# Define paths and parameters
#r"C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_LA_train\flac"
TEST_DATASET_PATH = r"C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_LA_train\flac"
MODEL_PATH = "audio_classifier20.h5"  # Replace with the actual path to your saved model
SAMPLE_RATE = 16000
DURATION = 5
N_MELS = 128
MAX_TIME_STEPS = 109
```

```python
# Load the saved model
model = load_model(MODEL_PATH)
```

Figure 22: Reading the files and load the model

```python
# Set the desired sample size
#sample_size = 16369  # Adjust this to your desired size
sample_size = len(df_eval)

# Extract a random sample of file names from the test data
sample_test_files = random.sample(os.listdir(TEST_DATASET_PATH), sample_size)

X_teste = []

for file_name in sample_test_files:
    file_path = os.path.join(TEST_DATASET_PATH, file_name)

    # Load audio file using librosa
    audio, _ = librosa.load(file_path, sr=SAMPLE_RATE, duration=DURATION)

    # Extract Mel spectrogram using librosa
    mel_spectrogram = librosa.feature.melspectrogram(y=audio, sr=SAMPLE_RATE, n_mels=N_MELS)
    mel_spectrogram = librosa.power_to_db(mel_spectrogram, ref=np.max)

    # Ensure all spectrograms have the same width (time steps)
    if mel_spectrogram.shape[1] < MAX_TIME_STEPS:
        mel_spectrogram = np.pad(mel_spectrogram, ((0, 0), (0, MAX_TIME_STEPS - mel_spectrogram.shape[1])), mode='constant')
    else:
        mel_spectrogram = mel_spectrogram[:, :MAX_TIME_STEPS]

    X_teste.append(mel_spectrogram)

# Convert list to numpy array
X_teste = np.array(X_teste)

# Predict using the loaded model
y_pred = model.predict(X_teste>0.5)

# Convert probabilities to predicted classes
y_pred_classes = np.argmax(y_pred, axis=1)

y_pred
```

```
800/800 [==============================] - 10s 12ms/step
array([[0.],
       [0.],
       [0.],
       ...,
       [0.],
       [0.],
       [0.]], dtype=float32)
```

Figure 23: Convert the audio file to Mel spectrogram and then convert it into array and make prediction

```python
# Get True Labels

# Path to the ASVspoof 2019 protocol file
PROTOCOL_FILE_PATH = "C:\mine\SDAIA T5\CAPSTONE\LA\LA\ASVspoof2019_LA_cm_protocols\ASVspoof2019.LA.cm.train1.trn.txt"


# Dictionary to store true labels for each file
true_labels = {}

# Read the protocol file
with open(PROTOCOL_FILE_PATH, 'rb') as protocol_file:
    lines = protocol_file.read().decode('utf-8').splitlines()
    print(lines)

for line in lines:
    line = line.strip()  # Strip leading/trailing whitespace
    parts = line.split()
    if len(parts) > 1:  # Check if line has enough parts to extract label
        file_name = parts[1]
        label = parts[-1]  # Last part contains the label
        true_labels[file_name] = label

# Now 'true_labels' contains the true labels for each file
true_labels
```

```
['LA_0079 LA_T_1138215 - - bonafide', 'LA_0079 LA_T_1271820 - - bonafide', 'LA_0079 LA_T_1272637 - - bonafide', 'LA_0079 LA_T_1276960 - -
{'LA_T_1138215': 'bonafide',
 'LA_T_1271820': 'bonafide',
 'LA_T_1272637': 'bonafide',
 'LA_T_1276960': 'bonafide',
 'LA_T_1341447': 'bonafide',
 'LA_T_1363611': 'bonafide',
 'LA_T_1596451': 'bonafide',
 'LA_T_1608170': 'bonafide',
 'LA_T_1684951': 'bonafide',
```

Figure 24: Reading the file label for train files

```python
y_true = np.array([1 if label == "bonafide" else 0 for label in true_labels.values()]) # y_true are the true labels for each file
y_true
```

```
array([1, 1, 1, ..., 0, 0, 0])
```

Figure 25: Convert labels into an array of 0 and 1

4. Discussion and Results

4.1 Evaluation Metrics

4.1.1 Train result

```
# CONFUSION MATRIX

#y_true = pd.DataFrame(y_train).apply(np.argmax, axis=1)
#y_pred = np.argmax(model.predict(X_train), axis=1)

cm = confusion_matrix(y_train, model.predict(X_train)>0.5)
#cm = confusion_matrix(pd.DataFrame(y_train).apply(np.argmax, axis = 1), model.predict(X_train))

# Display the confusion matrix
classes = ["FAKE", "REAL"]
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classes)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()
```

```
1425/1425 [==============================] - 18s 13ms/step
```

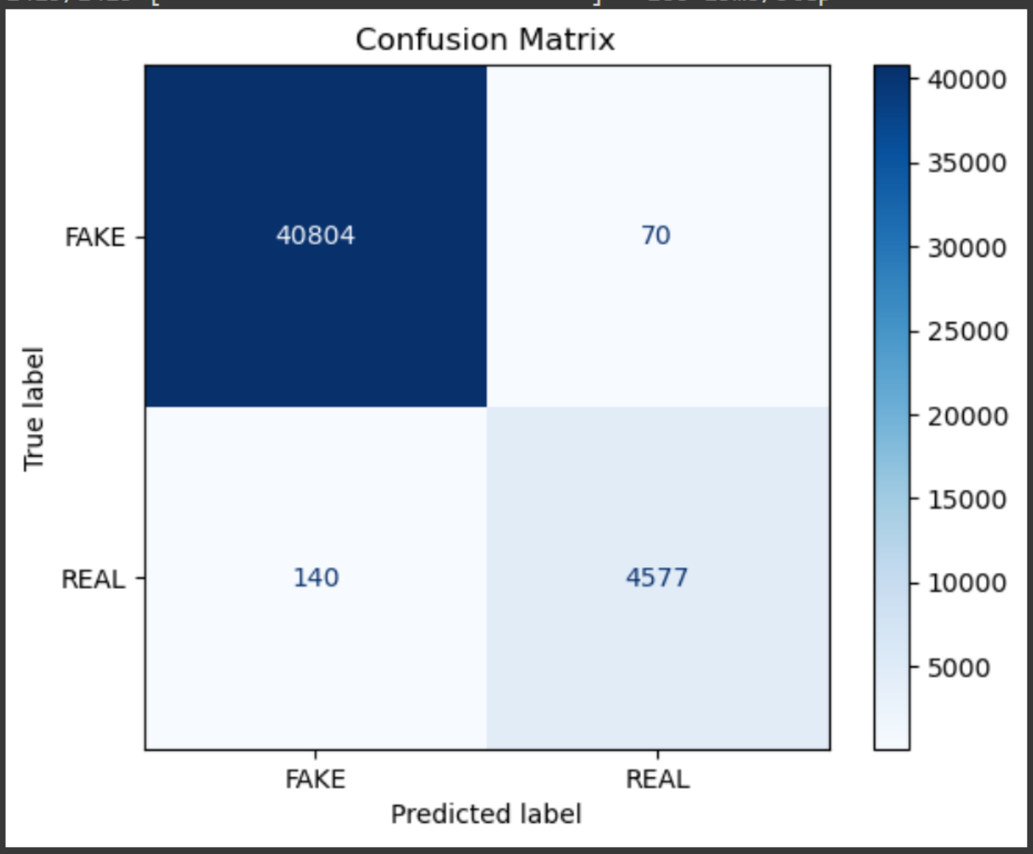

Figure 26: Train Confusion Matrix

4.1.2 Valid result

```
[ ] #Display the confusion matrix
    cm = confusion_matrix(y_val, model.predict(X_val)>0.5)
    classes = ["spoof", "bonafide"]
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classes)
    disp.plot(cmap=plt.cm.Blues)
    plt.title("Confusion Matrix")
    plt.show()

    357/357 [==============================] - 5s 13ms/step
```

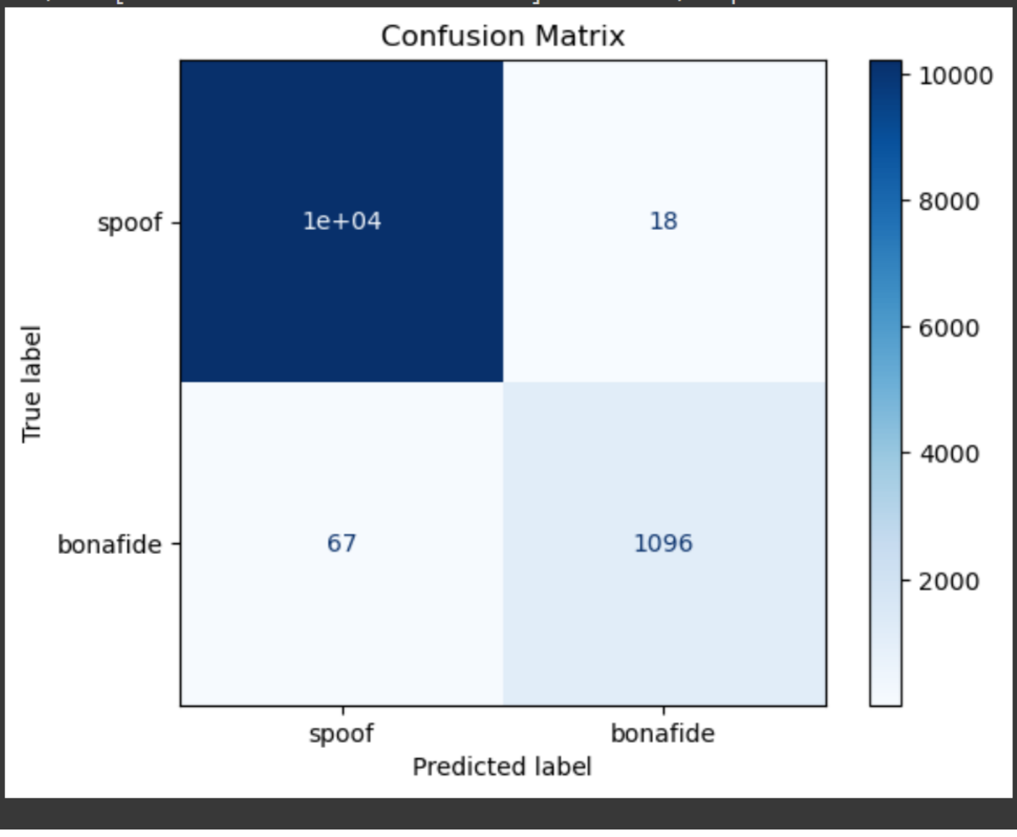

Figure 27: Valid Confusion Matrix

4.1.3 Test result

```
#Display the confusion matrix
cm = confusion_matrix(y_test, model.predict(X_test)>0.5)
classes = ["spoof", "bonafide"]
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classes)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()

446/446 [==============================] - 6s 12ms/step
```

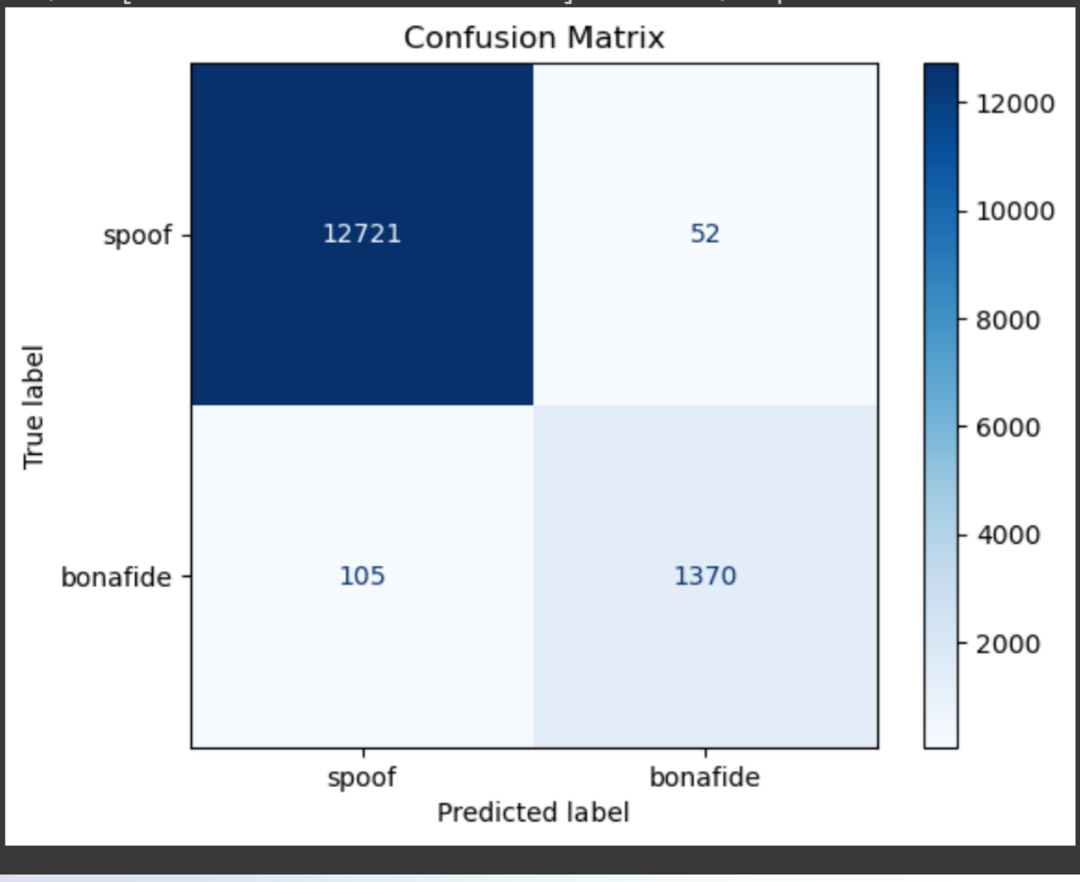

Figure 28 Test Confusion Matrix

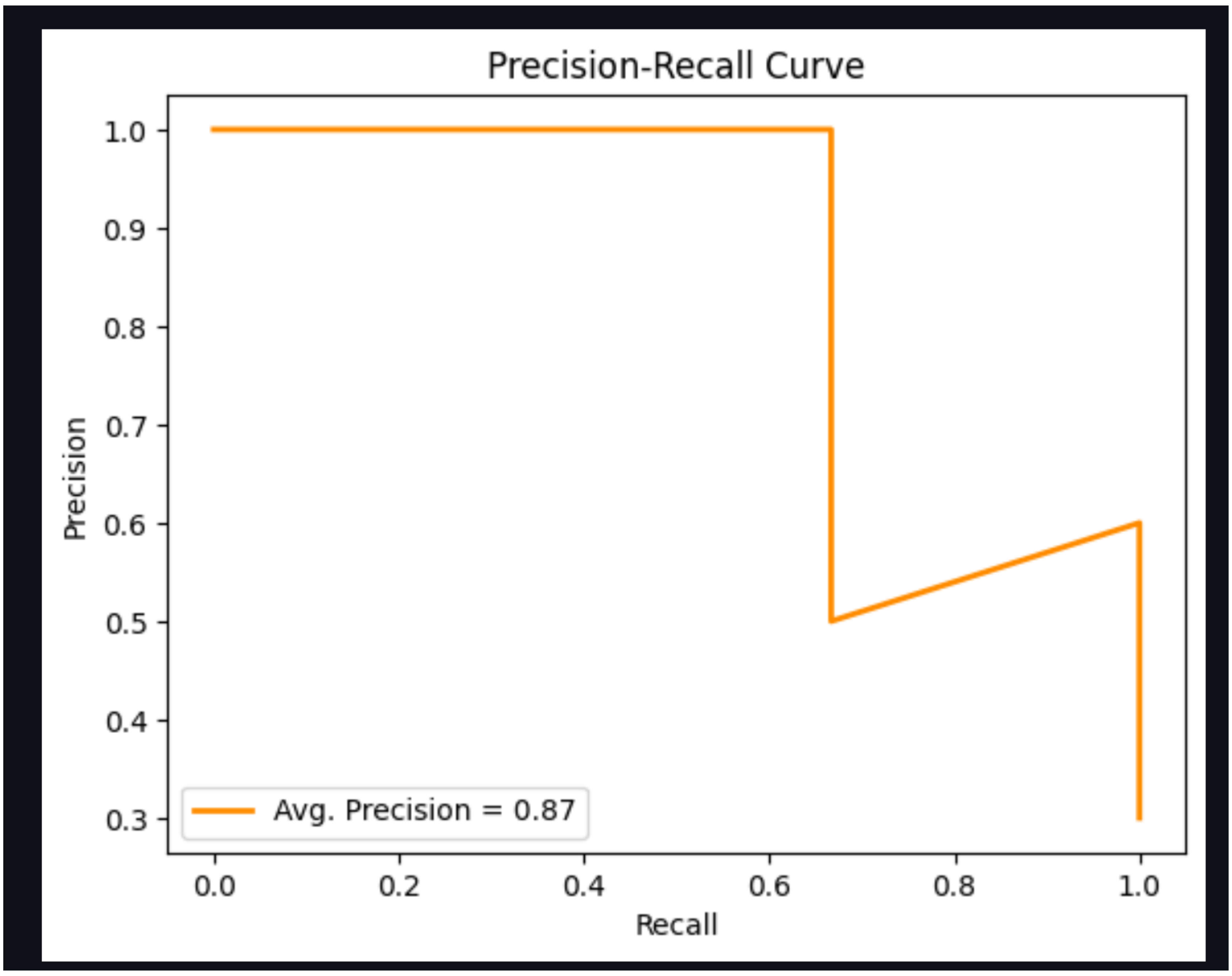

Figure 29: Precision-Recall Curve

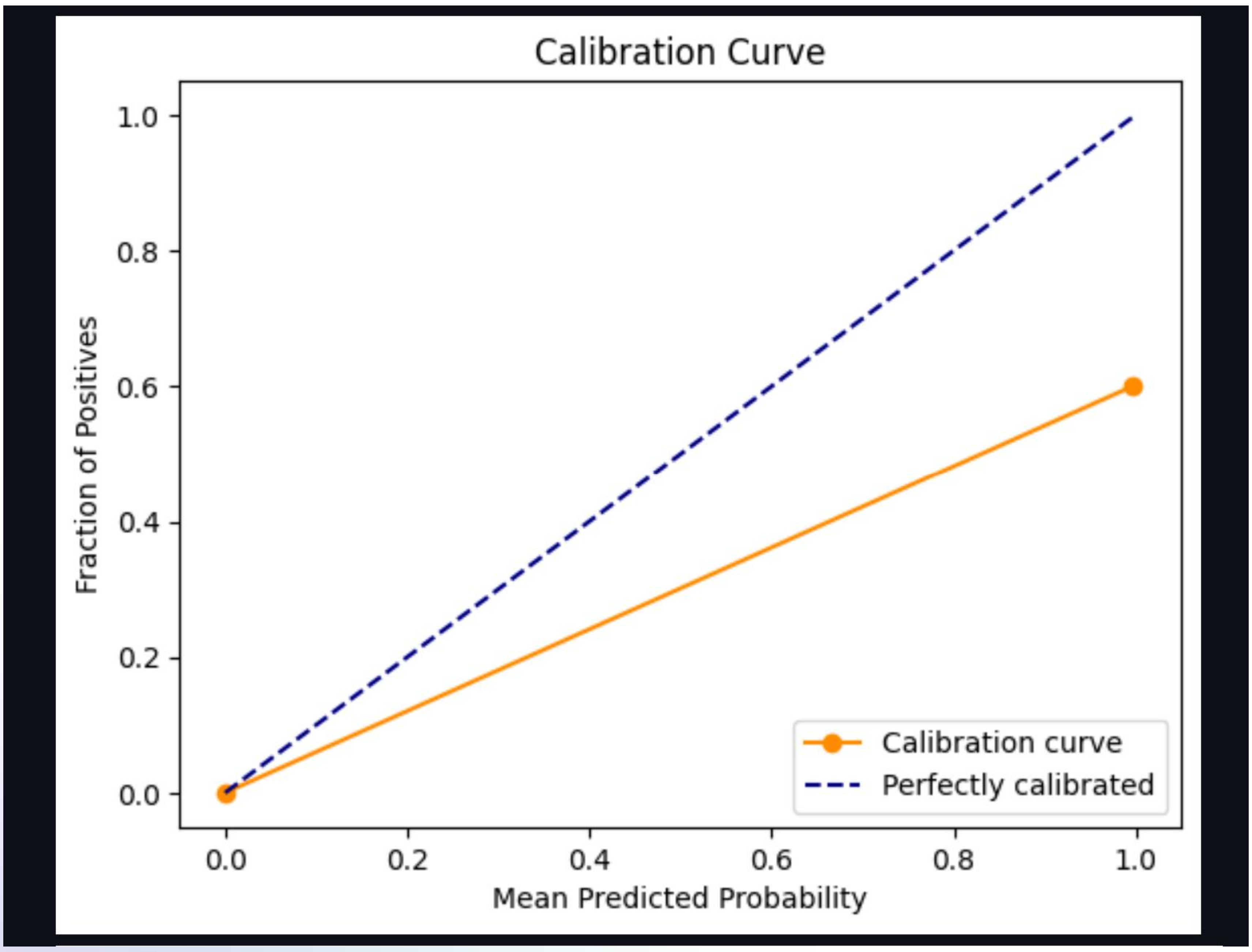

Figure 30: Calibration Curve

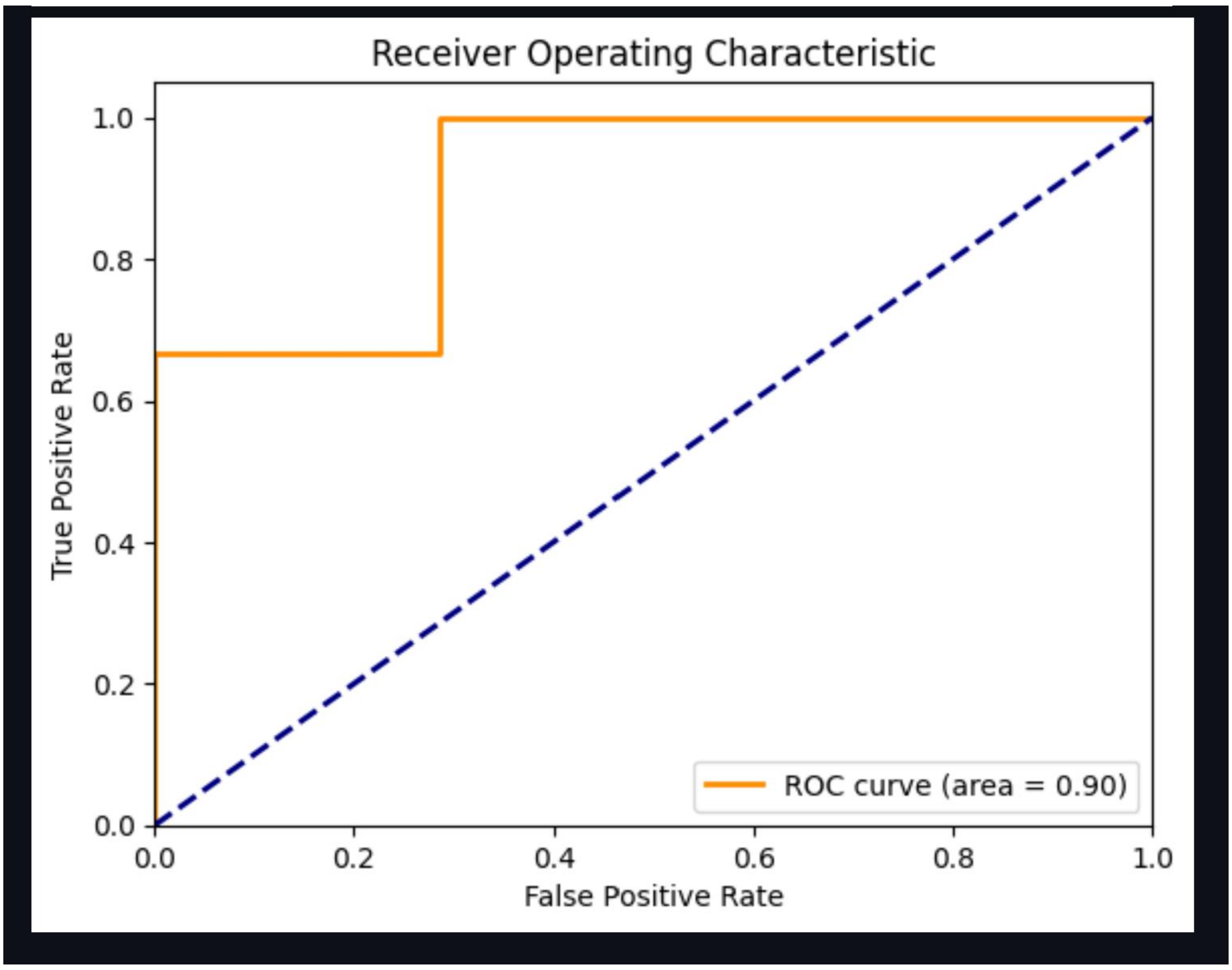

Figure 31: Receiver Operating Characteristics

Conclusion

In conclusion, the integration of deepfake detection technology aligns seamlessly with Saudi Arabia's Vision 2030, reflecting the nation's commitment to innovation, technological advancement, and safeguarding the integrity of digital content. As the Kingdom pursues its ambitious goals of diversifying the economy, fostering a knowledge-based society, and promoting a vibrant and dynamic digital landscape, addressing the challenges posed by deepfake technology becomes paramount.

By investing in deepfake detection mechanisms, Saudi Arabia not only protects its citizens and institutions from the potential misuse of manipulated content but also ensures the trustworthiness of information in an increasingly digital world. The incorporation of technologies for deepfake detection not only enhances the nation's cybersecurity infrastructure but also promotes a secure and reliable digital environment conducive to the growth of various industries

Future Work

For our future work will focus on refining the model to detect more sophisticated softwares and integration of the model with real-time hardwares. Also, support multiple languages as well as the following:

1 - Deepfake image detection

Next big milestone is to develop a model that can detect deepfake images.

2 - Deepfake video detection

After completing deepfake image detection, the next milestone is to develop deepfake detection for videos.

3-interpretability

Finally, after completing these milestones, next up is to interpret based on what does our model predict for these three projects.

Team

Abdulaziz Alnajjar | 3bdal3zez20@gmail.com

Saad Almoghirh | saadalmoghirh@gmail.com

Salem Bawazir | sobawazir10@gmail.com