UNIVERSITÀ DEGLI STUDI DI PADOVA

DeepFake Audio Detection

GitHub Repository:
https://github.com/LovatoTomas/DF_AudioDeepfakeDetection

Authors : Tomas Lovato, Alisea Bovo

Project Overview

This project focuses on detecting audio DeepFakes using the ASVspoof 2019 dataset, which includes 3 different datasets : Training Set, Development Set, and Evaluation Set with different distributions of spoofing techniques.

The goal is to evaluate preprocessing techniques and machine learning models to distinguish between bonafide (real) speech and spoofed audio.

Dataset Details

Speaker Distribution:

- Training: 20 speakers (8 male, 12 female)
- Development: 20 speakers (8 male, 12 female)
- Evaluation: 48 speakers (21 male, 27 female)

ASVspoof
Automatic Speaker Verification and Spoofing Countermeasures Challenge

Samples:

- Logical Access (LA): Training (25380), Dev (24844), Evaluation (72000)

Spoofing Techniques:

- 6 known attack types (TTS and VC) and 11 unknown attack types.

Audio Preprocessing Strategies #1

The real difference in the solutions lies in how it has been handle the input audio:

Mel Spectrogram (for CNN) :

- Load audio with Librosa
- Fixed sample rate: 16 kHz
- Normalize audio in time
- Pad or truncate to standard duration
- Extract Mel spectrogram

librosa

```python
# Iterate over each file and its label
for file_name, label in tqdm(dataset_labels.items(), desc="Loading and converting files"):
    file_path = os.path.join(dataset_path, file_name + ".flac")

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

        # Compute Mel spectrogram
        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)  # Convert to decibel scale

        # Adjust spectrogram time steps to MAX_TIME_STEPS
        if mel_spectrogram.shape[1] < MAX_TIME_STEPS:
            # Pad if time steps are less than MAX_TIME_STEPS
            mel_spectrogram = np.pad(mel_spectrogram, ((0, 0), (0, MAX_TIME_STEPS - mel_spectrogram.shape[1])), mode='constant')
        else:
            # Trim if time steps exceed MAX_TIME_STEPS
            mel_spectrogram = mel_spectrogram[:, :MAX_TIME_STEPS]

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

    except Exception as e:
        print(f"Error processing file {file_name}: {e}")

# Convert lists to numpy arrays
X = np.array(X)
y = np.array(y)

return X, y
```

Audio Preprocessing Strategies #2

The real difference in the solutions lies in how it has been handle the input audio:

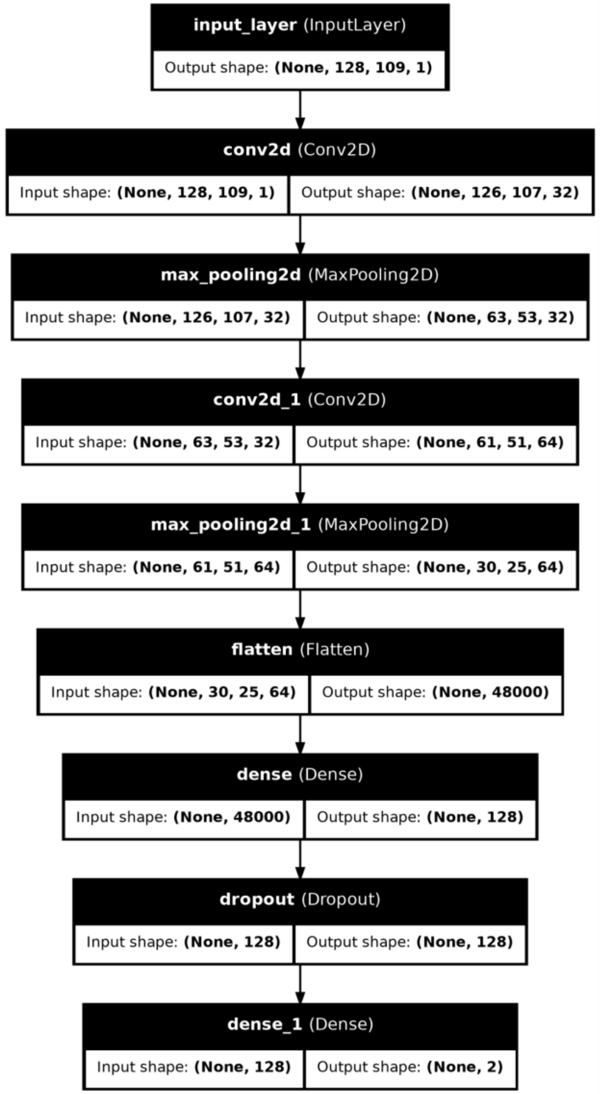
STFT Features (with dimensionality reduction):

- Load audio with Librosa
- Fixed sample rate: 16 kHz
- Normalize audio in time
- Scale values (normalization)
- Extract STFT features
- Reduce features

```python
def compute_stft(audio_files, n_fft=2048, hop_length=512):
    """
    Extract Short-Time Fourier Transform (STFT) features from a list of audio signals.

    Args:
        audio_files (list): List of audio signals.
        n_fft (int): Number of FFT components. Default is 2048.
        hop_length (int): Number of samples between successive frames. Default is 512.

    Returns:
        np.array: Array containing the magnitude spectrograms for each audio signal.
    """
    stft_features = []  # Initialize a list to store STFT features for each audio file.

    for audio in tqdm(audio_files, desc="Extracting STFT features"):
        # Compute the Short-Time Fourier Transform (STFT) of the audio signal.
        stft = librosa.stft(audio, n_fft=n_fft, hop_length=hop_length)

        # Compute the magnitude spectrogram by taking the absolute value of the STFT result.
        spectrogram = np.abs(stft)

        # Append the magnitude spectrogram to the features list.
        stft_features.append(spectrogram)

    # Convert the list of spectrograms to a NumPy array and return it.
    return np.array(stft_features)
```

CNN with Mel Spectrogram

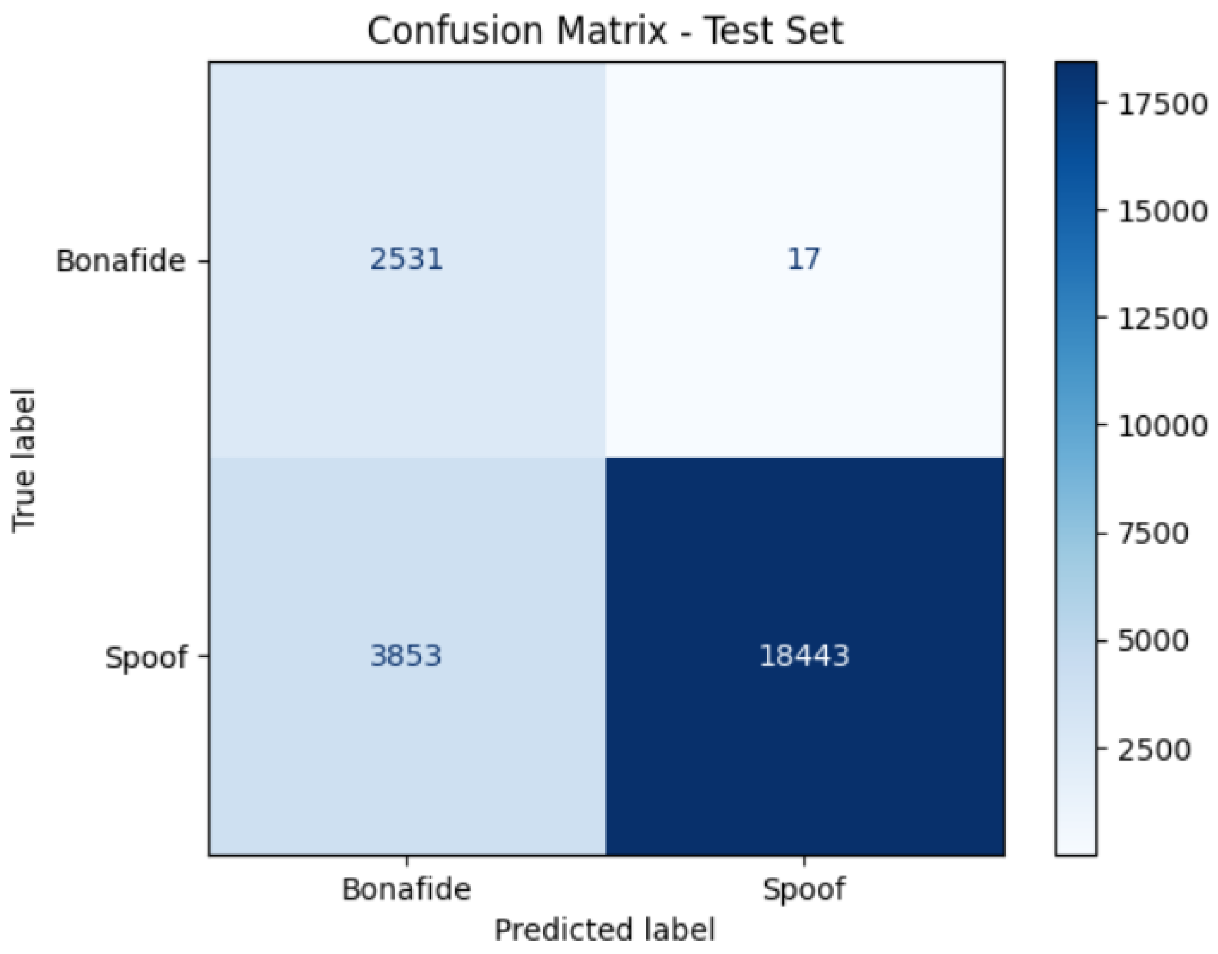
Using a simple CNN model on an unbalanced training set.

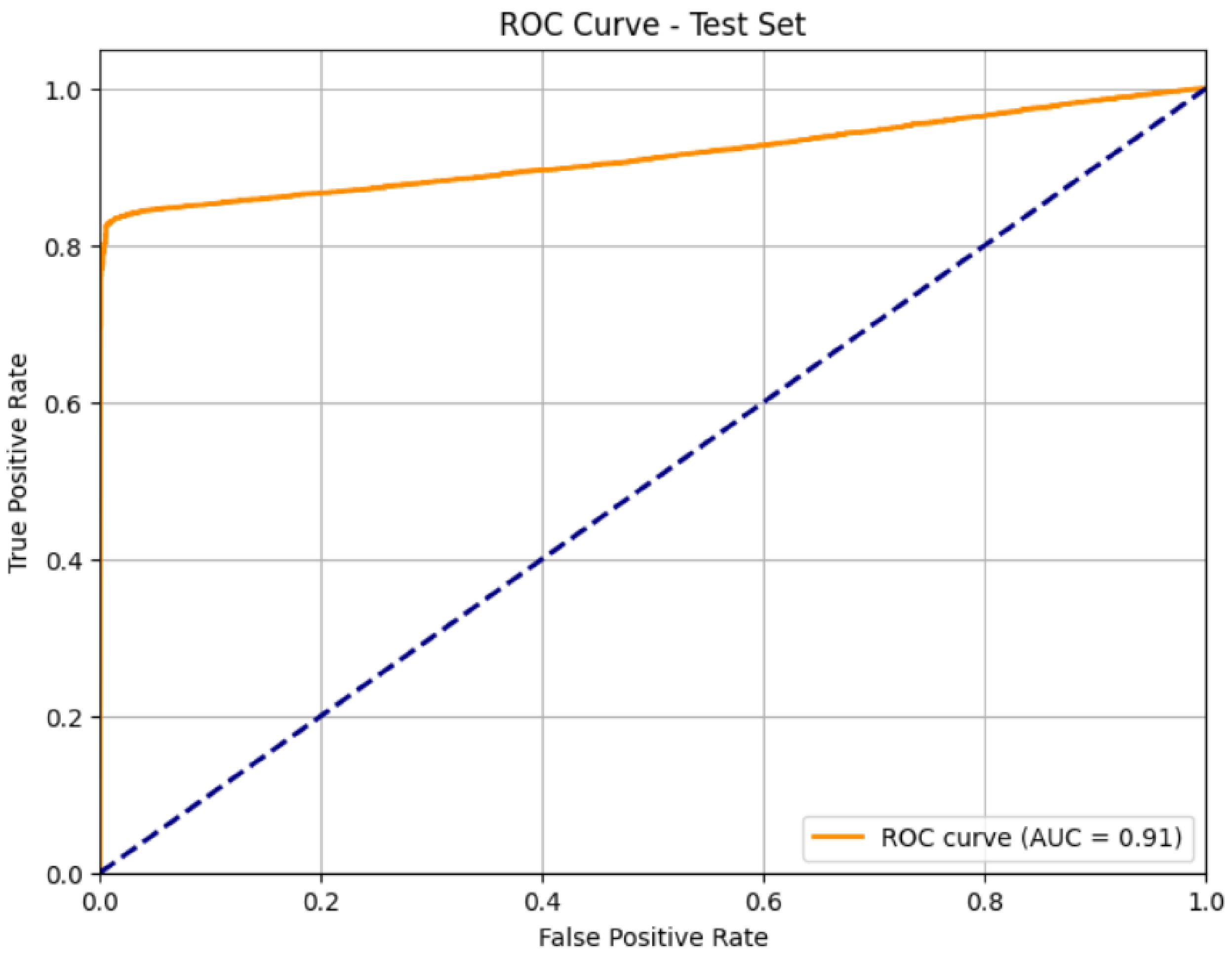


The results obtained are high accuracy, but biased predictions. The model tended to classify most test samples as spoofed.

On the Test Set, the unseen data performed worse.

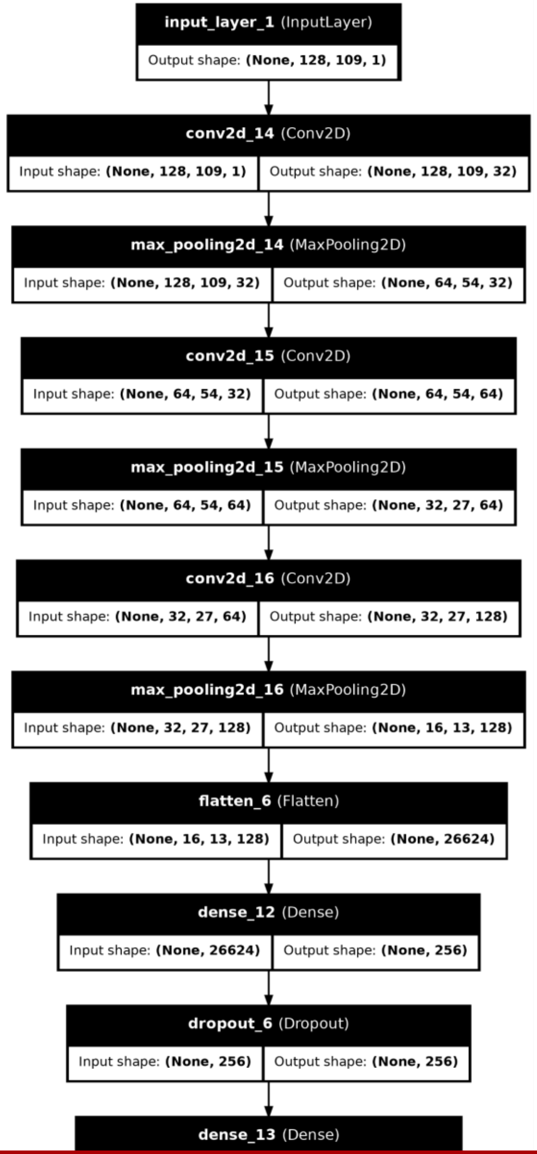
CNN – MEL – Unbalanced Training





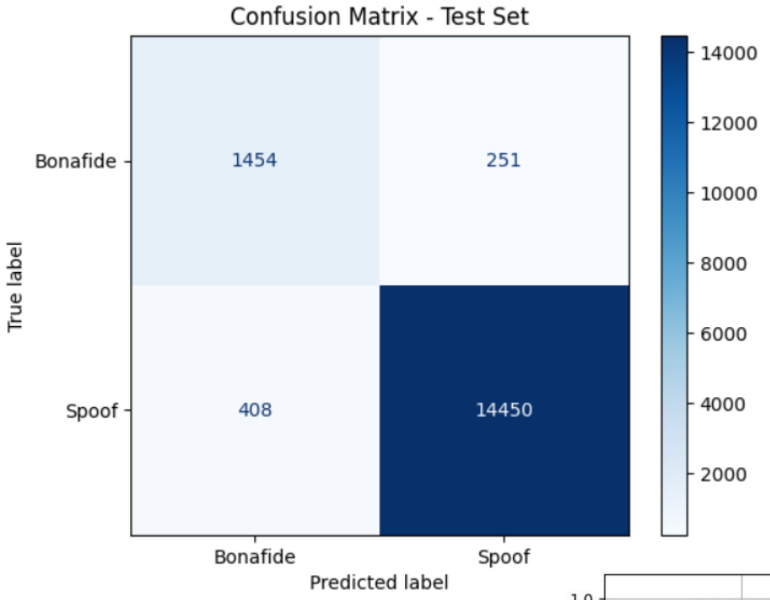
CNN with Mel Spectrogram

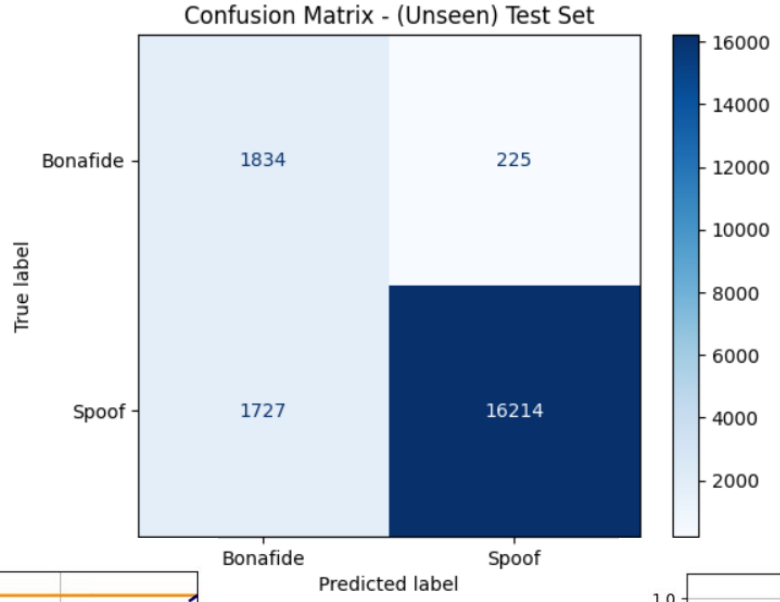
Load the training set, filter the bonafide samples, shuffle the spoof samples, and balance the two classes by selecting an equal number of samples from each.



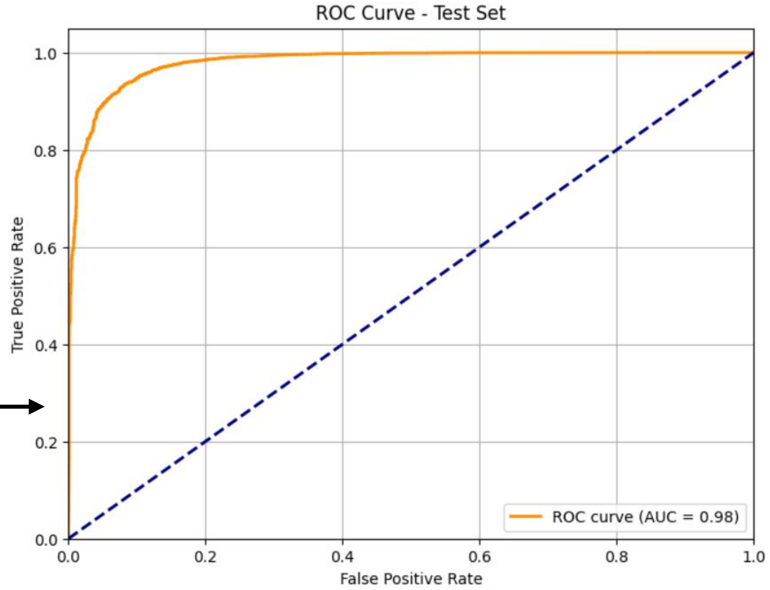
It led to significant improvements in the confusion matrix and ROC scores.
Both for seen and unseen datasets.

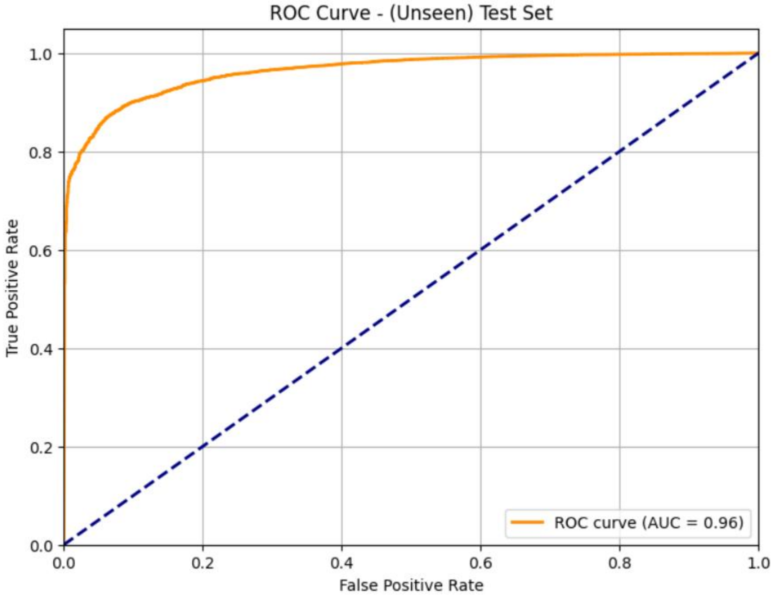
CNN – MEL – Balanced Training





Test set with a different distribution

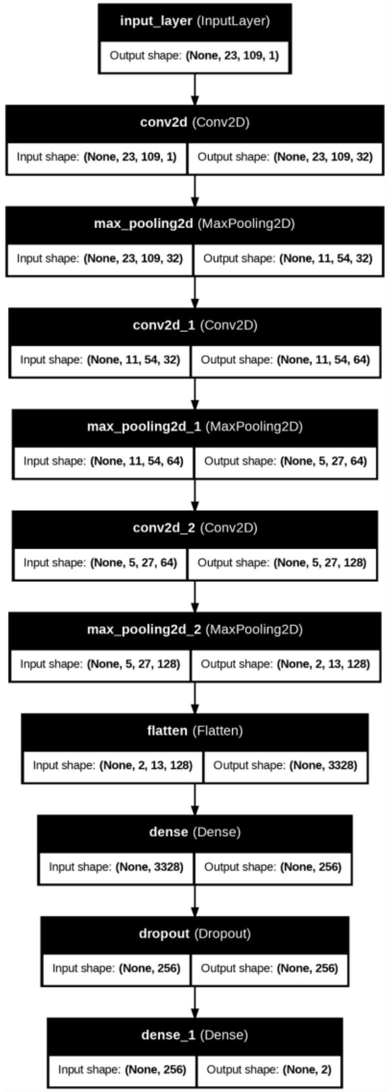




Test set with same distribution of the training set

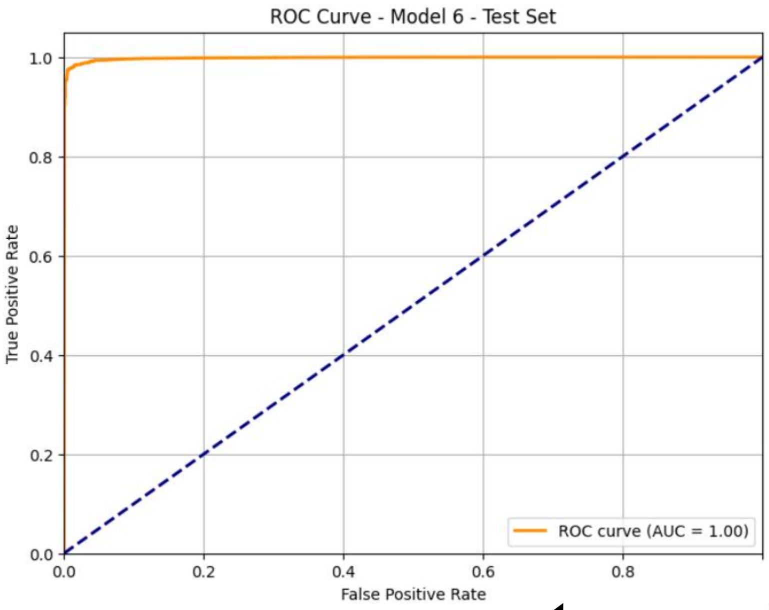
Final CNN Architecture

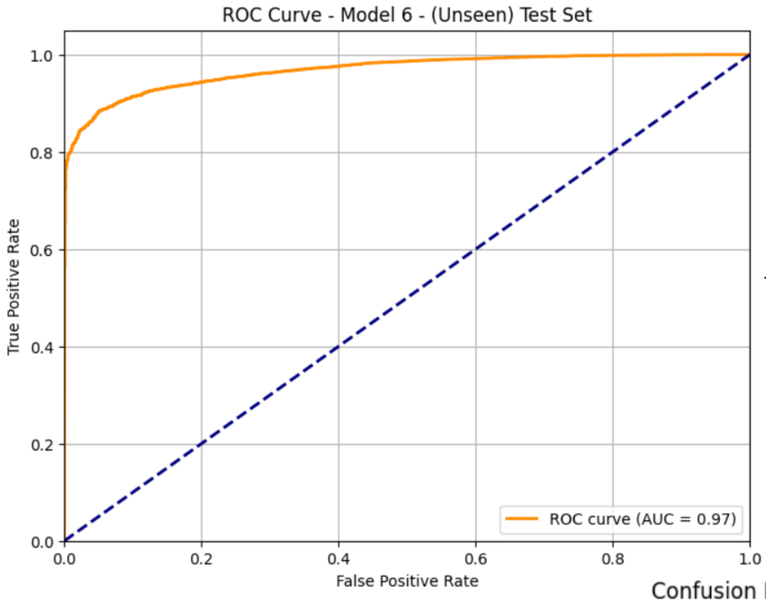
Larger kernels were used in the convolution process to better capture temporal dependencies.



As a result, the model achieved near-perfect metrics on both seen and unseen datasets.

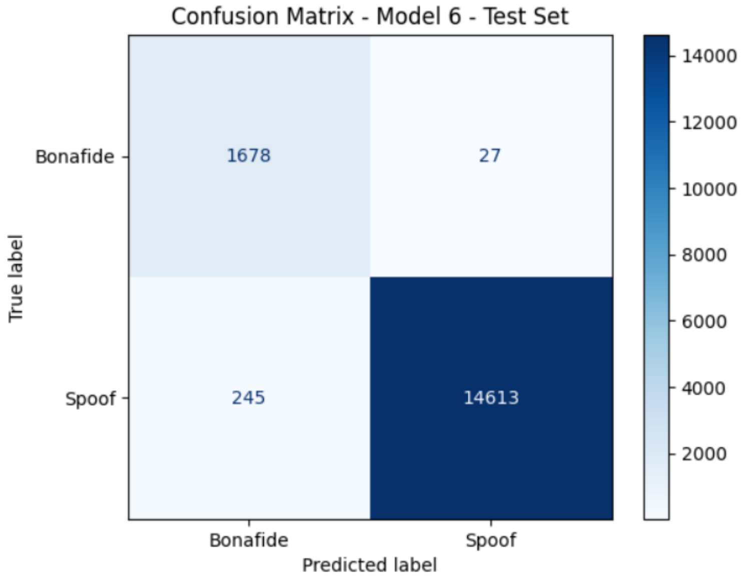
Final CNN Architecture

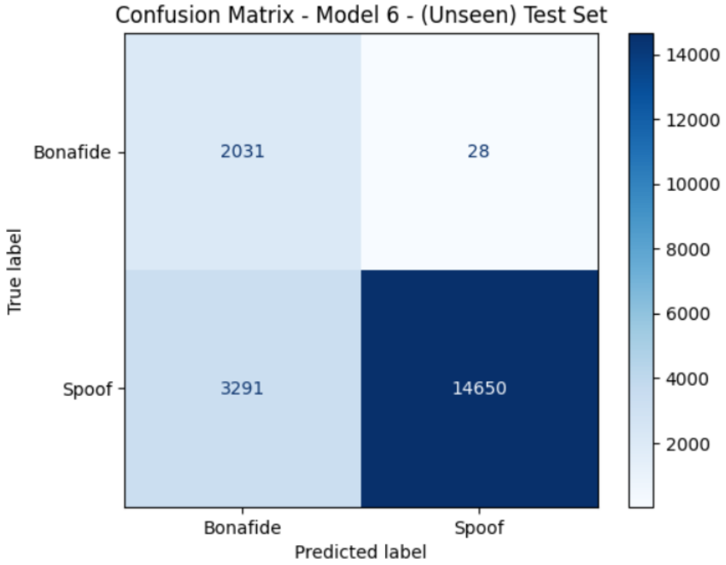




Test set with a different distribution

Test set with same distribution of the training set

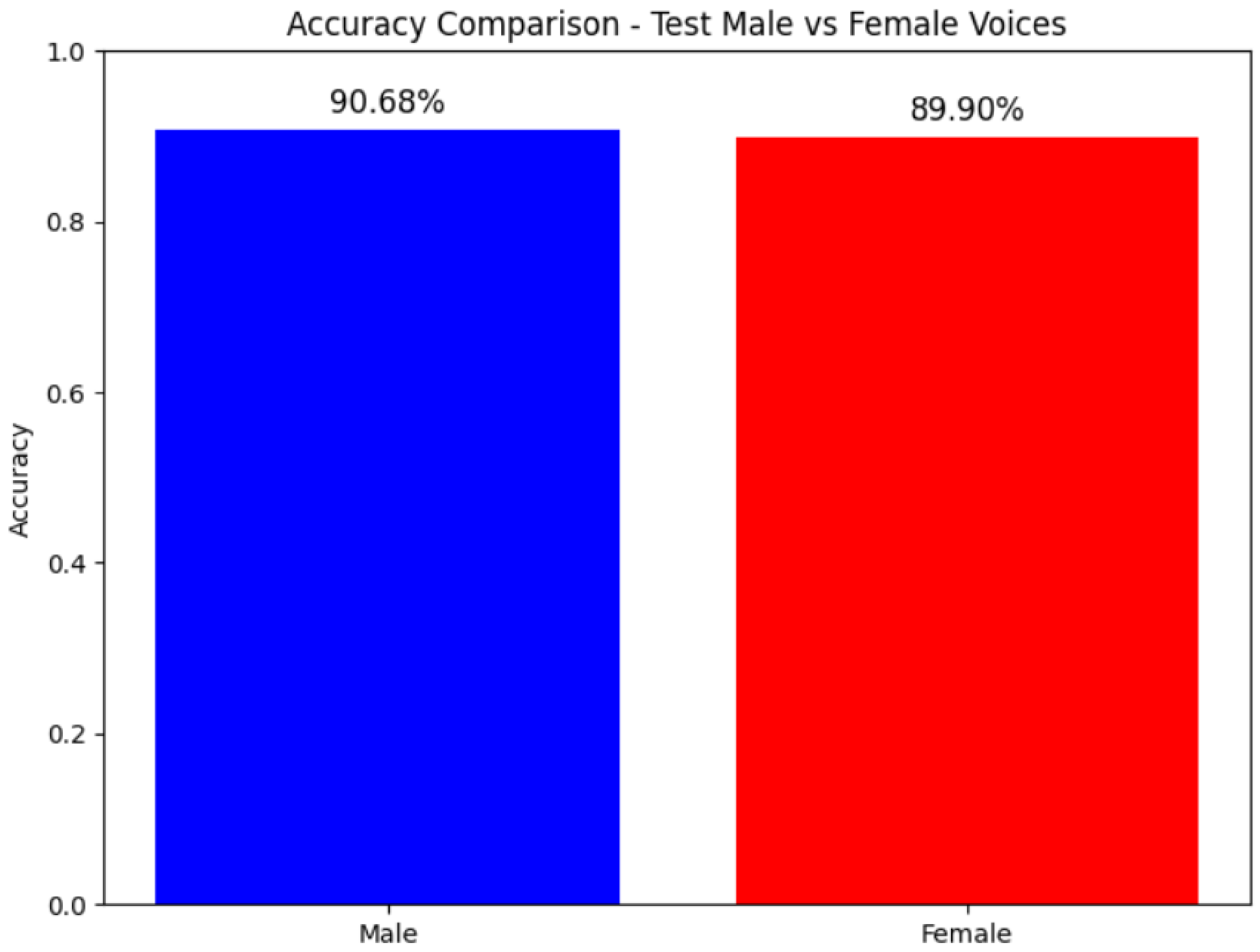




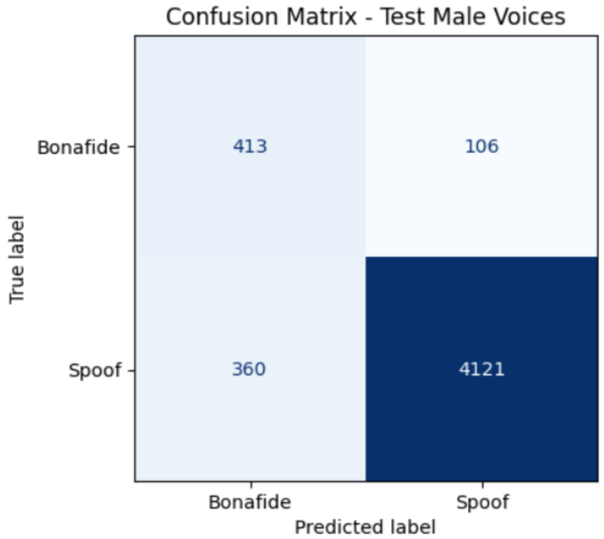
Fairness Analysis

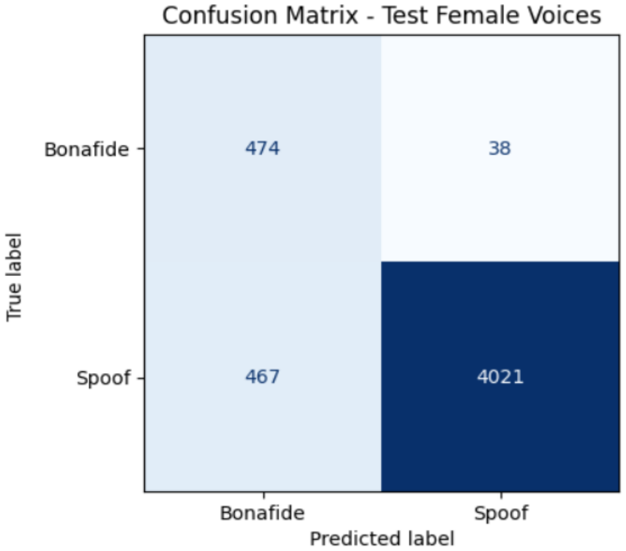
Despite a higher number of female samples, the model did not exhibit gender bias.

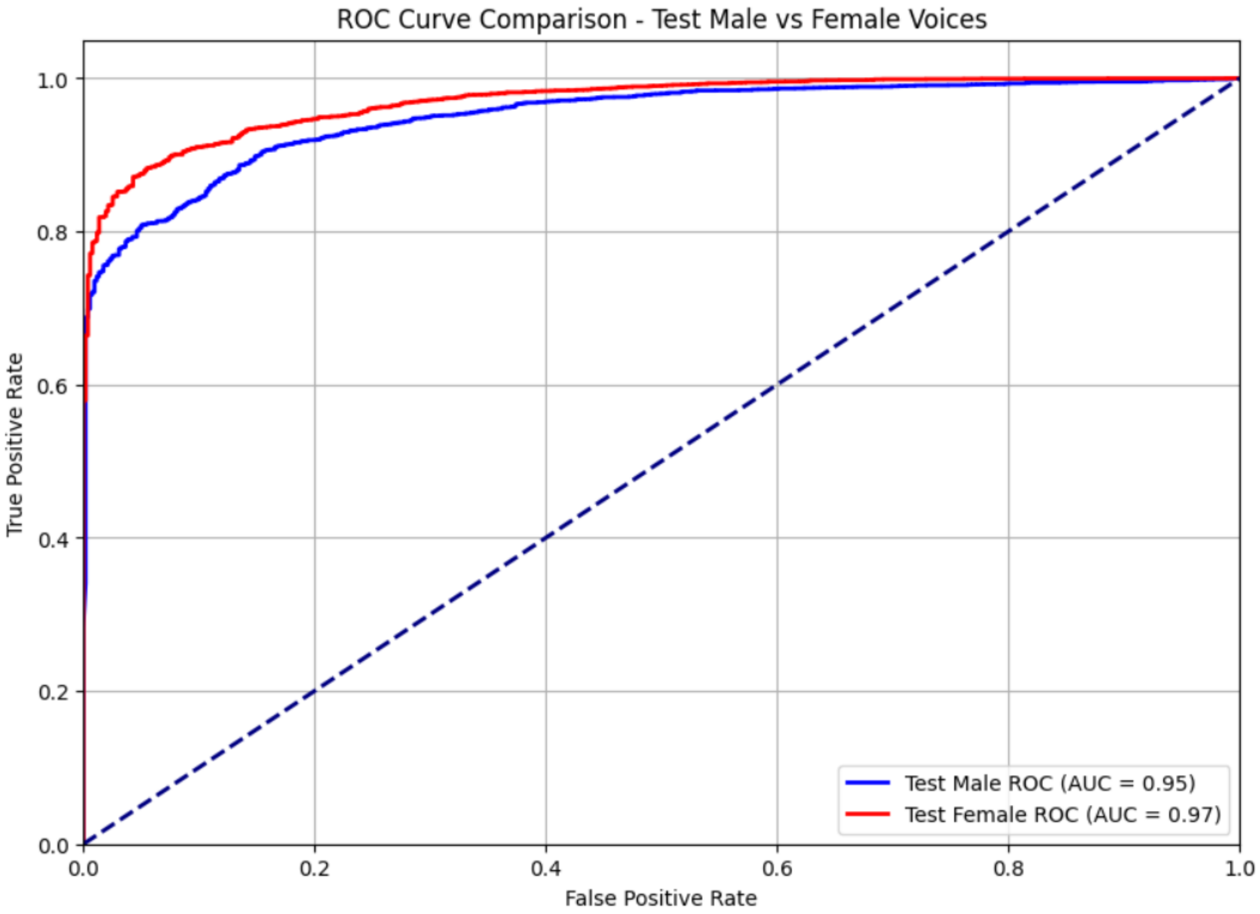
Slightly better performance for female voices in some cases.



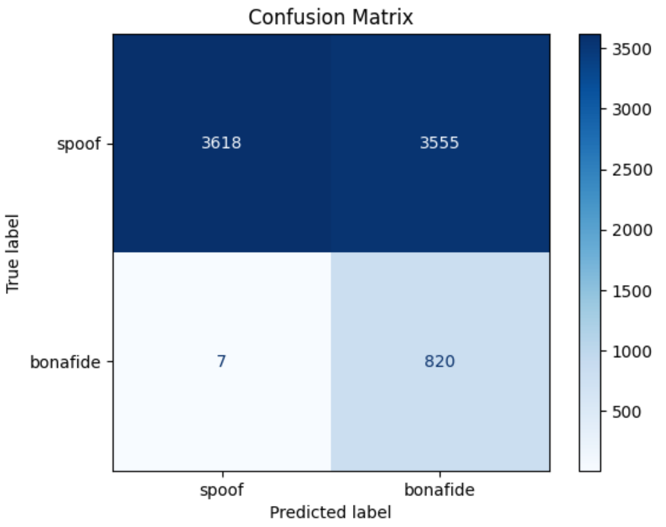
Fairness Analysis

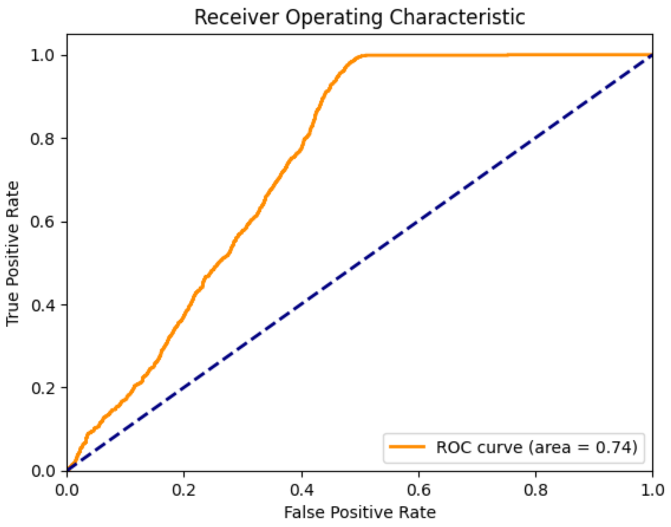


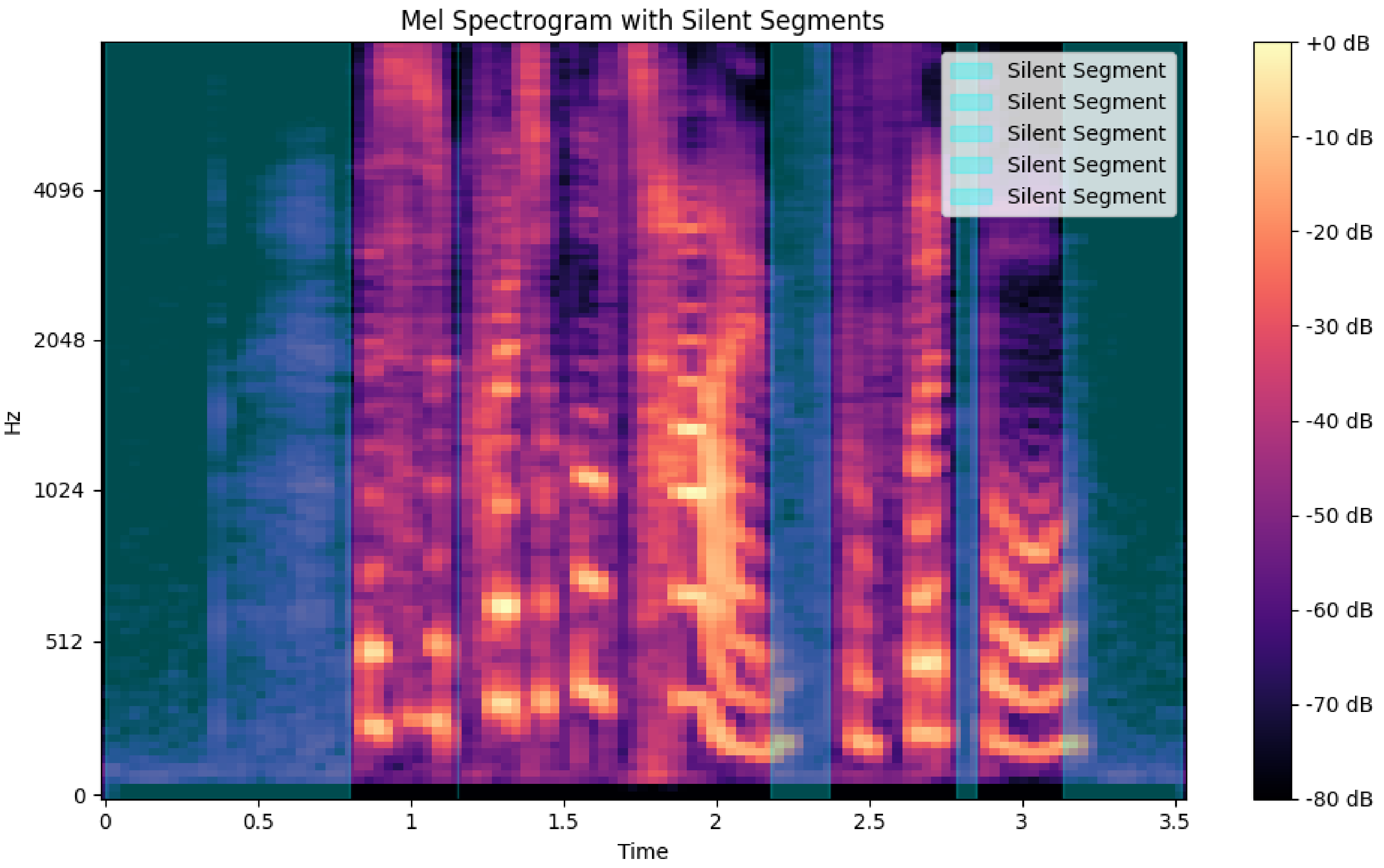




Silence Filtering

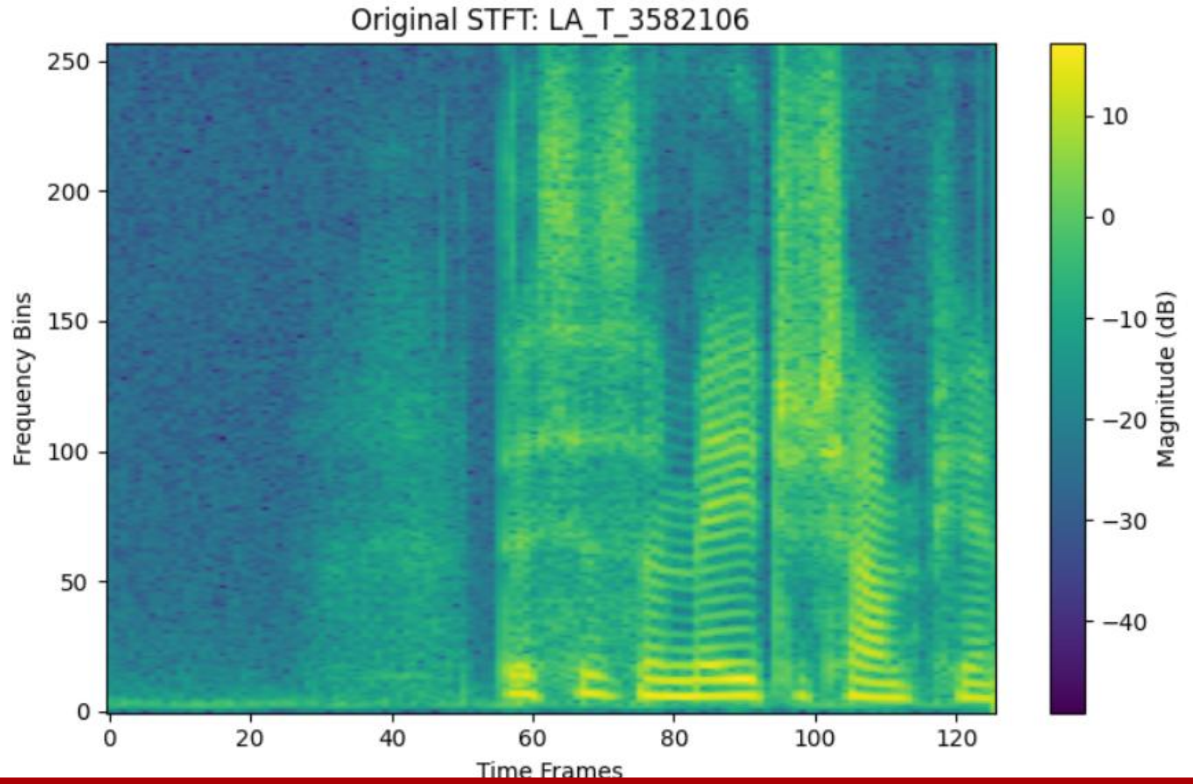


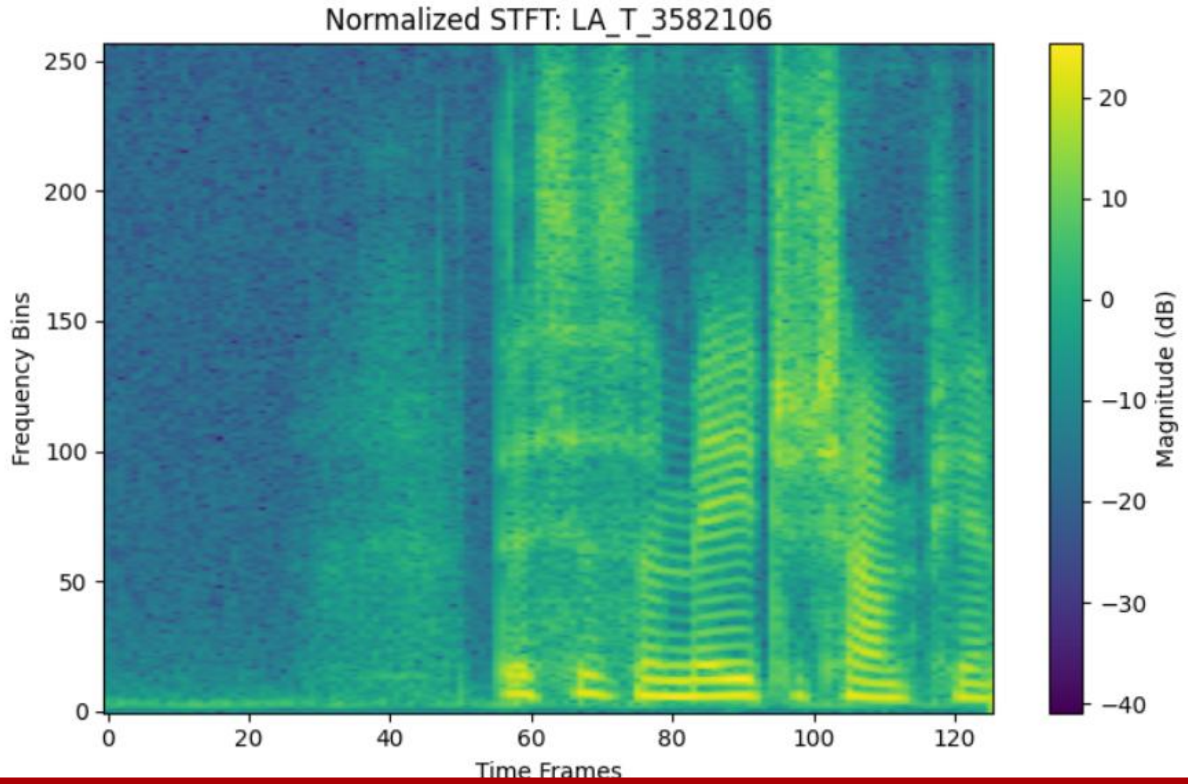




STFT Feature Extraction

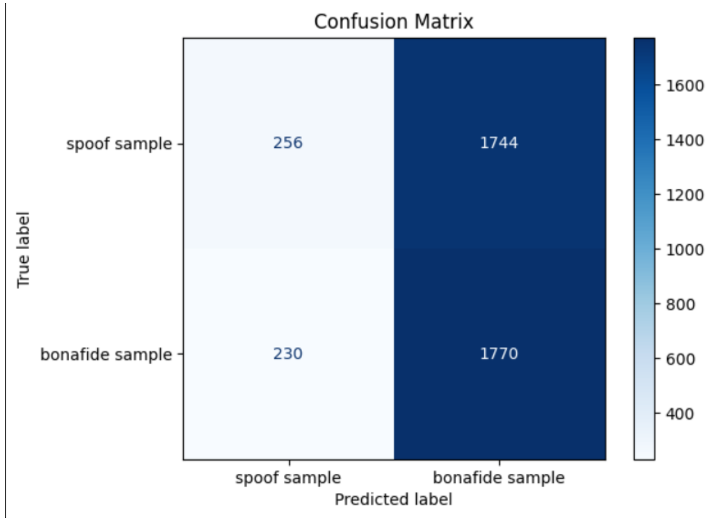
STFT features required dimensionality reduction using Autoencoders (AE). It did not perform well due to the similarity between bona fide and spoofed data and it is limited in its ability to manage inputs with more than one dimension.
Prior to each operation, the dataset undergoes initial normalization.

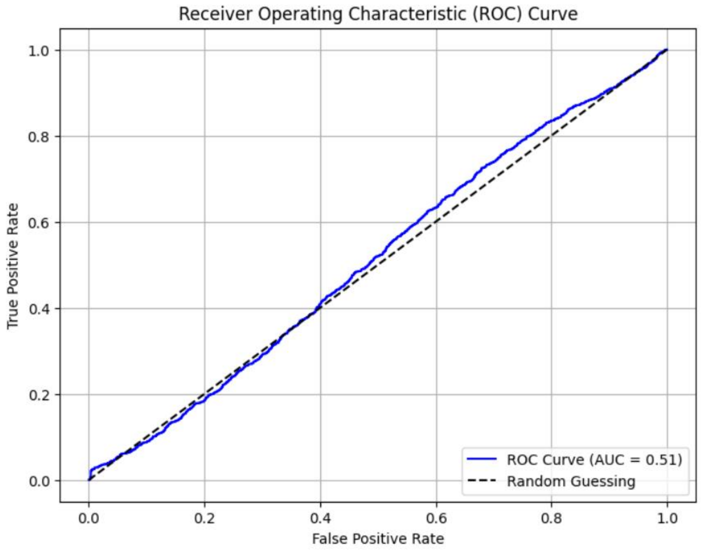


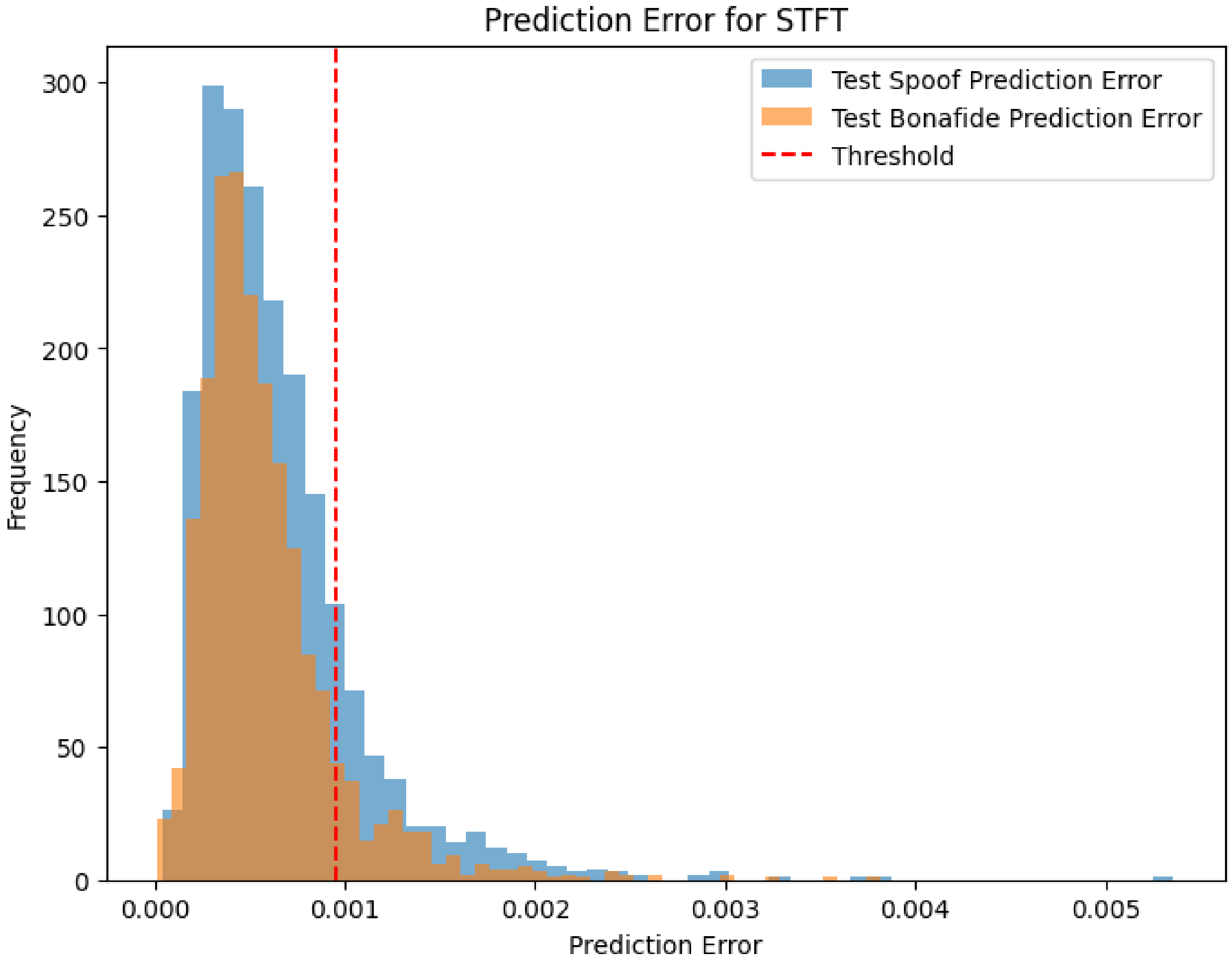


Università degli Studi di Padova

STFT Feature Reduction AE

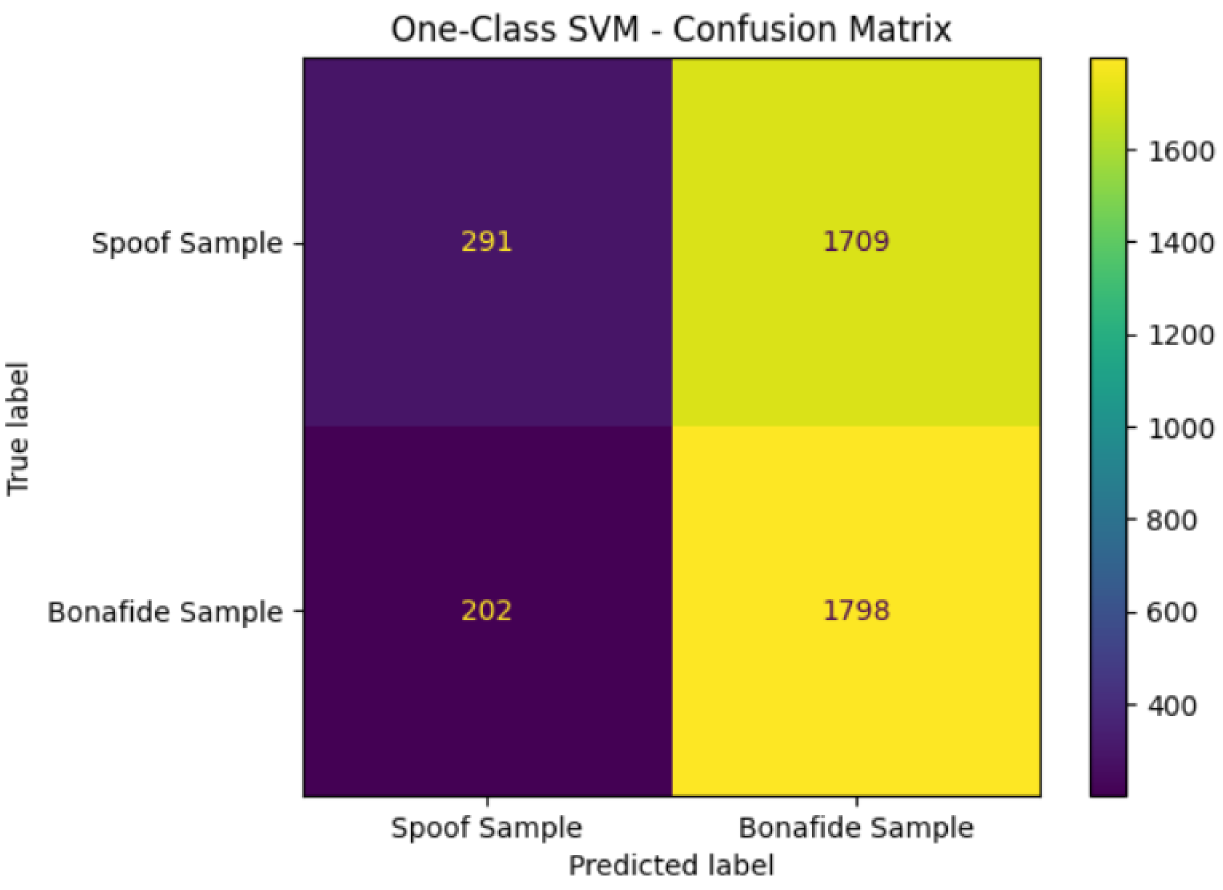


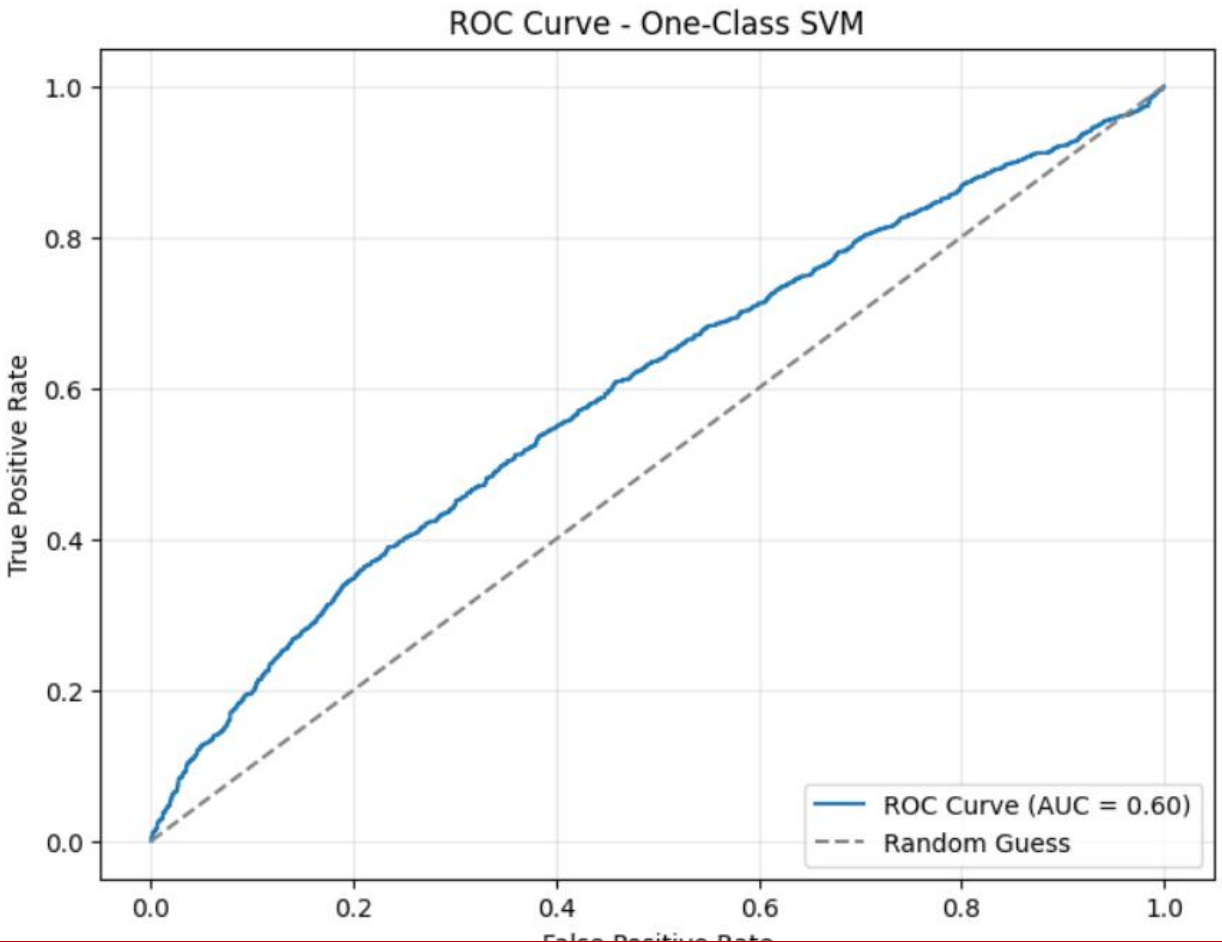




STFT – OCSVM Prediction

This One-class-SVM receives as input the reconstructed samples from the latent space of the autoencoder.





Alternative work

- MFCC Features and SVM:

The model achieved an ROC score of around 0.7, with moderate false positives and negatives, and future work will focus on parameter tuning through grid search to enhance performance.

- Mel spectrogram and decision tree:

The model didn’t perform well, infact it was obtained a negative R^2. This indicates that the model performs worse than a baseline model that predicts the most frequent class for all instances.

Conclusions

Achievements:
We successfully applied effective preprocessing of Mel Spectrograms with CNNs, leading to improved DeepFake detection performance. Additionally, using MFCC and STFT features with SVM provided valuable insights for better model accuracy.

Challenges:
The model faces challenges in generalizing to unseen spoofing attacks, and the computational complexity of feature extraction remains a hurdle.

To improve:
Future efforts will focus on improving model generalization, optimizing feature extraction, and enhancing computational efficiency for practical deployment.