UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

# Audio spoofing detection

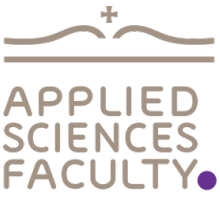
*Author:*
Dmytro IVASHCHENKO

*Supervisor:*
Pablo MALDONADO

*A thesis submitted in fulfillment of the requirements*
*for the degree of Master of Science*

*in the*

Department of Computer Sciences
Faculty of Applied Sciences


Lviv 2023

# Declaration of Authorship

I, Dmytro IVASHCHENKO, declare that this thesis titled, "Audio spoofing detection" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

*"Tools such as artificial intelligence, automated voice systems, machine learning, deepfakes, interactive memes, virtual reality, and augmented reality will make digital disinformation more effective and harder to combat."*

Sam Woolley

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

**Audio spoofing detection**

by Dmytro IVASHCHENKO

# *Abstract*

Efficient and accurate audio spoofing detection is crucial to ensuring audio-based systems' security and integrity. Existing methods often mainly focused on the performance of the detection system. This master thesis focuses on the development of advanced techniques that prioritize efficiency while maintaining high detection performance. We introduced the model, consisting of an encoder and a classifier, which can efficiently learn complex representations with a lack of labeled data. We introduce suitable loss functions to effectively distinguish spoofed and bonafide speech in latent space to keep the performance high. The results demonstrate notable improvements in both encoder performance and classification accuracy, highlighting the potential for enhanced self-supervised audio analysis techniques.

**Keywords:** self-supervised learning, audio spoofing detection, automatic speaker verification, audio processing, speech classification

# *Acknowledgements*

I would like to express my sincere gratitude to my supervisor, Pablo Maldonado, for his invaluable guidance, patience, and constructive feedback throughout the process of writing my diploma. His expertise and support have been instrumental in shaping this research.

I would like to acknowledge and thank the entire UCU collective of lecturers and students for creating an exceptional environment of learning and cooperation. The memories and knowledge I gained from studying and collaborating with such a remarkable community will stay with me.

I would like to extend my heartfelt thanks to Oleksii Molchanovskyi for his exceptional management throughout the whole educational process.

I am deeply grateful to my loved ones for their unwavering support and presence throughout this journey. Their encouragement and belief in me have been a constant source of motivation.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **CNN** | **C**onvolutional **N**eural **N**etwork |
| **LPS** | **L**og-**P**ower **S**pectrum |
| **DFT** | **D**iscrete **F**ourier **T**ransform |
| **FFT** | **F**ast **F**ourier **T**ransform |
| **DCT** | **D**iscrete **C**osine **T**ransform |
| **CQT** | **C**onstant **Q** **T**ransform |
| **LFCC** | **L**inear **F**requency **C**epstral **C**oeficients |
| **MFCC** | **M**el-**F**requency **C**epstral **C**oeficients |
| **IMFCC** | **I**nverted **M**el-**F**requency **C**epstral **C**oeficients |
| **CQCC** | **C**onstant **Q** **C**epstral **C**oeficients |
| **LCNN** | **L**ight **C**onvolutional **N**eural **N**etwork |
| **SSAD** | **S**elf-Supervised **S**poofing **A**udio **D**etection |
| **TCN** | **T**emporal **C**onvolutional **N**etwork |
| **MFM** | **M**ax-**F**eature-**M**ap |
| **MSE** | **M**ean-**S**quared **E**rror |
| **EER** | **E**qual **E**rror **R**ate |

*Dedicated to my beloved parents*

# Chapter 1

# Introduction

## 1.1 Motivation

With the development of information transmission and storage technologies, speech plays an increasingly important role. The speech is highly informative and easy to understand. At the same time, the rapid development of technologies for changing speech characteristics and speech synthesis for fraud creates new challenges in all spheres of life. Moreover, progress in deep fake generation makes spoofed audio more realistic and leads to needing new audio spoofing detection technologies that are more effective and robust.

Audio Spoofing Detection is an important Audio Classification subtask. Its purpose is to build systems that detect whether human speech was spoofed using filters, vocoders, or deep generative models. With the rise of tools for audio spoofing and deep fake generation for phishing individuals, organizations, and governments, the selected topic becomes significantly more relevant. Because of this rapid development, the complexity lies in the relatively small amount of labeled data and the reliability of systems based on classical machine learning approaches.

Self-supervised learning has emerged as a promising approach to leverage the abundance of unlabeled audio data for training neural networks. Self-supervised learning enables models to learn meaningful representations from unannotated data, which can later be transferred to various downstream audio analysis tasks. However, despite the progress made in this area, challenges still need to be addressed further to enhance the efficiency and performance of existing methods. The project aims to explore existing state-of-the-art approaches to audio spoofing detection, especially semi-supervised ones, analyze their advantages and disadvantages, suggest possible modifications, and develop not a computationally expensive effective system with high classification quality.

## 1.2 Thesis structure

In chapter 2, we discuss the most relevant approaches to audio feature extraction, audio classification, and semi-supervised learning related to speech data. After that, in chapter 3, we provide the most popular for-use datasets and data preprocessing explanation. Then, in chapter 4, we propose our solution, including encoder and classifier architectures, loss functions, and evaluation metrics. In chapter 5, we provide an overview of the conducted experiments and analyze the results. Finally, in chapter 6, we list what could be done in future research and conclude the paper.

# Chapter 2

# Related work

## 2.1 Feature extraction techniques

This section briefly discusses the most practical features for speech recognition, verification, and automatic spoofing detection tasks.

### 2.1.1 Spectral features

Spectral transition, which is the movement of frequencies in the part of audio from high to low and vice-versa, plays an important role in human speech perception (Rabiner and Juang, 1993). The part of the utterance where spectral variation was locally maximum contained the essential phonetic information in the syllable. Therefore it is reasonable that using spectral features to distinguish bonafide from spoofed speech should contribute significantly to overall recognition performance.

As a first step of spectral feature extraction, some transformation should be applied to a raw waveform to obtain the signal's spectrum. The Fourier transform plays a central role in spectral analysis. Since speech is a non-stationary time series, the transformation should be applied appropriately, using framing and windowing of the signal, to extract the most information from it (Logan, 2000). Usually, the Discrete Fourier Transform (DFT) is applied to calculate the spectrum.

Along with using DFT in speech processing, there are significant drawbacks. The width of the windowing function relates to how the signal is represented — it determines whether there is good frequency resolution (frequency components close together can be separated) or good time resolution (the time at which frequencies change). A wide window gives better frequency resolution but poor time resolution. A narrower window gives good time resolution but poor frequency resolution (Liu et al., 2019). One of the possible solutions is to use a Constant Q Transform (CQT) (Brown, 1991) with a constant fraction of frequency resolution to time resolution (Q factor).

### 2.1.2 Log-Power Spectrum

To calculate the Log-Power Spectrum (LPS) of the audio waveform, the speech signal is divided into frames by applying windowing functions to remove edge effects. After that, DFT is applied frame-wise, and the square of the output is calculated. The obtained result is called the power spectrum. Finally, the power spectrum's logarithm is taken since a signal's perceived loudness is approximately logarithmic (Logan, 2000). The representation is suitable for a broad range of audio-related tasks. Still, it has high dimensionality, so other techniques were developed to construct features with lower dimensionality based on LPS.

### 2.1.3 Mel-frequency cepstral coefficients

Cepstrum is the result of computing the inverse Fourier transform (IFT) of the LPS. Since the LPS can be treated as a new signal, a cepstrum can be interpreted as the spectrum of the LPS. Cepstral characteristics play a crucial role in speech processing.

According to psychophysical study O'Shaughnessy, 1987, human perception of the frequency content of sounds follows this scale, which is defined in the following way:

$$f_{mel} = 2595 \log_{10} \left(1 + \frac{f}{700}\right) \tag{2.1}$$

where $f_{mel}$ is the subjective pitch in Mels corresponding to $f$, the actual frequency in Hz (Chakroborty, Roy, and Saha, 2008). A subjective pitch is present on Mel Frequency Scale to capture important characteristics of phonetics in speech. Mel-frequency Cepstral Coefficients (MFCC) concept is based on human hearing perceptions which cannot perceive frequencies over 1 kHz (Muda, Begam, and Elamvazuthi, 2010). Thus, the intuition behind the MFCC is the peculiarities of human perception of lower and higher frequencies. MFCCs have two types of filters: spaced linearly at low frequencies below 1000 Hz and logarithmic spacing above 1000 Hz. MFCCs are calculated by applying Mel scaling onto the LPS using the Mel filter bank and summing the energy in each filter. Then the Discrete Cosine Transformation (DCT) is performed to hold as much information encoded on the first coefficients of the decomposition.

### 2.1.4 Linear Frequency Cepstrum Coefficients (LFCC) and Inverted Mel Frequency Cepstral Coefficients (IMFCC)

While MFCC presents a way to transform speech into a perceptually meaningful feature based on the human auditory system, it is not evident that the human ear and, thus, MFCC are suitable for speech recognition tasks. For this reason, modifications were proposed to this approach.

One example is Linear Frequency Cepstrum Coefficients (LFCC). The LFCC is computed similarly to MFCC, but the DCT is performed directly on LPS instead of Mel-scaled spectral representation (Davis and Mermelstein, 1980).

Another example is Inverted Mel Frequency Cepstral Coefficients (IMFCC). In Chakroborty, Roy, and Saha, 2008, the inverted Mel scale is proposed:

$$\widehat{f}_{mel} = 2195.286 - 2595 \log_{10} \left(1 + \frac{4031.25 - f}{700}\right) \tag{2.2}$$

where $\widehat{f}_{mel}$ is the subjective pitch in inverted Mel scale corresponding to $f$, the actual frequency in Hz. IMFCCs are calculated in an MFCC manner, but the inversed Mel scale is applied instead of the Mel scale. While MFCC focuses on lower frequencies, IMFCC targets distinguishing higher frequencies and capturing their characteristics.

### 2.1.5 Constant Q cepstral coefficients

The CQT, initially proposed in music processing (Brown, 1991), employs geometrically spaced frequency bins. This resembles the human perception system, like Mel scale concept. In Todisco, Delgado, and Evans, 2017 introduced Constant Q Cepstral Coefficients (CQCC). CQCCs are calculated similarly to LFCC, but instead of DFT for LPS computing, CQT is used. CQCC could capture a time-frequency spectrum

representation with characteristics missed by more classical approaches to feature extraction.

### 2.1.6 Linear Predictive Coding Coefficients and Linear Prediction Cepstral Coefficients

In many cases, raw waveform analysis shows solid results for the speech recognition and speech verification task. Auto-regression-based features can be one such representation. Linear Predictive Coding (LPC) Coefficients remove the redundancy from a signal and try to predict the following values by linearly combining the previously known coefficients. LPC is the all-pole filter that represents the spectral envelope of a digital speech in compressed form using a linear prediction model (Sharma, Umapathy, and Krishnan, 2020).

First, normalization and preemphasis on the input signal are performed to calculate LPC. Then, the result is split into frames. On each frame, windowing is applied. Then, the auto-correlation analysis should be done, and coefficients obtained from auto-regression on each frame.

Linear Prediction Cepstral Coefficients (LPCC) are computed from LPC in the following way (Davis and Mermelstein, 1980):

$$LPCC_i = LPC_i + \sum_{k=1}^{i-1} \frac{k-i}{i} LPCC_{i-k} LPC_k, \qquad i = 1, 2, \dots \tag{2.3}$$

LPCC can represent the acoustic signal as the frequency for a certain time without signal distortion (Gupta and Gupta, 2016). LPC and LPCC features benefit audio spoofing detection tasks (Sahidullah, Kinnunen, and Hanilçi, 2015).

## 2.2 Classification models

The initial classification models for audio spoofing detection were statistical models based on the abovementioned features. However, with the development of audio synthesis and deep fake technologies, the statistical approach works worse with new data. Thus, more complex models, like deep neural networks, are frequently used to distinguish spoofed audio.

### 2.2.1 Gaussian mixture model

The Gaussian mixture model (GMM) is a widely used generative model in speech processing (Wu et al., 2017). It represents each class as a weighted sum of M multivariate Gaussians, $p(x|\lambda) = \sum_{i=1}^{M} w_i p_i(x)$, where $w_i$ is the i-th mixture weight and $p_i(x)$ is a D-variate Gaussian density function with mean vector $\mu_i$ and covariance matrix $\Sigma_i$. The model parameters are defined by $\lambda = \{w_i, \mu_i, \Sigma_i\}_{i=1}^{M}$. Expectation-maximization (EM) algorithm is used to estimate the parameters of each class independently via the maximum likelihood (ML) criterion. For the test, given the models, $\lambda_b$ and $\lambda_s$, and feature vectors of the test speech, $Y = \{y_1, \dots, y_T\}$, the classification score is computed as,

$$\Lambda(Y) = L(Y|\lambda_b) - L(Y|\lambda_s) \tag{2.4}$$

where $L(Y|\lambda) = \frac{1}{T}\sum_{t=1}^{T} \log p(y_t|\lambda)$ is the average loglikelihood of $Y$ given GMM model $\lambda$. $\lambda_b$ and $\lambda_s$ are the GMMs for bonafide and spoofed classes, respectively Hanilci and Kinnunen, n.d.

### 2.2.2 ResNet-based models

The evolution of neural networks for image classification gave major momentum for developing classification models for audio data. The ResNet-based architecture is proposed in the paper Alzantot, Wang, and Srivastava, 2019. The main advantage of ResNet is that this design solves the gradient vanishing problem utilizing skip connections, which enables the building of deeper models. In the first step, a raw waveform should be transformed by one of the following feature extraction algorithms: MFCC, CQCC, or LPS. This input is treated as a single-channel image and passed through an architecture based on residual blocks, dropout to prevent overfitting, leaky ReLU and sigmoid activation functions to model non-linear dependencies.

Another example of skip connections usage was introduced in Chen et al., 2020. As input, linear filter banks (LFBs) features are similar to LFCC. They provide a lower risk of overfitting and adequate computational costs. The model's architecture was analogous to Alzantot, Wang, and Srivastava, 2019, but four instead of six residual blocks and large margin cosine loss as a loss function were chosen.

The framework proposed in P. et al., 2020 refers to the transfer learning concept. To detect spoofed speech, the authors propose to use a pre-trained ResNet model on Mel-spectrograms as audio features representation. A significant advantage of this approach is that it provides faster training. Still, it is necessary to mention that spectrograms differ significantly from the usual images in their structure, which originates from the nature of sound, so it is not obvious that the pre-trained model for visual object detection will fit well.

In Zhang, Jiang, and Duan, 2021; Ding, Zhang, and Duan, 2022, authors proposed to modify loss functions to classify the spoofed speech effectively. Most spoof detection systems utilized softmax loss for binary classification. AM-Softmax loss function improves the previous function by introducing an angular margin to make the embedding distributions of both classes more compact (Zhang, Jiang, and Duan, 2021). Training a tight embedding space for bonafide speech in voice spoofing detection is reasonable. However, a compact embedding space for the spoofing attacks tends to overfit for known attacks. To address this issue, the authors Ding, Zhang, and Duan, 2022 propose introducing two different margins, using the new loss function called OC-Softmax, to compact the genuine speech better and isolate the spoofing attack, making the classification more adaptive.

### 2.2.3 Light CNN models

A different approach is introduced in Lavrentyeva et al., 2017; Lavrentyeva et al., 2019. The framework, based on CNN, named Light CNN (LCNN), was proposed. The architecture consists of stacked convolutional, pooling, and Max-Feature-Map (MFM) layers. A max-feature-map activation function is used then to calculate the element-wise maximum. The idea behind MFM is that it plays the role of a feature selector. Before the output, there is an FC layer with a softmax activation. As input for the model, LPS representations extracting with CQT, DCT, or FFT, and LFCC were used (Lavrentyeva et al., 2019). As a loss function, AM-Softmax was used. The model from Lavrentyeva et al., 2017 was constructed for the physical access scenario

of the spoofing attack, and it was adopted in Lavrentyeva et al., 2019 for the logical access scenario.

In Lavrentyeva et al., 2017, a bidirectional gated recurrent unit model was proposed, combining CNN and RNN properties, but it did not reach the LCNN performance. Another attempt was made in Gomez-Alanis et al., 2019, using LCNN in GRU to train gates output. One of the advantages was the fact, that the resulting model was comparably small and could effectively model temporal dependencies. However, the resulting model was overfitted, and the performance was moderate.

A notable framework was proposed in Wu et al., 2020. The introduced model takes genuine speech representation in the first stage as the input and generates genuine speech as the output, following the same distribution of genuine speech. The idea is that the output for the spoofed speech differs significantly from the bonafide speech on which the transformer was trained. Then the output is classified using the LCNN model. We can see that this model is more complex than the previous one, but the quality change was small, so it seems that more data should be used for training the transformer.

## 2.3 Self-supervised approach models

Labeled data is hard and expensive to obtain in real-world problems, so unsupervised learning is preferable. Instead of modeling complex dependencies with labeled data usage, we can find a suitable non-linear feature representation using the unlabeled data and then train the classifier on the resulting latent space using the labeled data. The advantage of the approach is that we can model complex non-linear dependencies without a lot of labeled data.

In the Chorowski et al., 2019, the WaveNet encoder-decoder model was proposed. This model aims to extract features using ResNet-like architecture for the encoder. As input for the encoder, MFCCs are passed. As the output, the best decoder model is trained to reconstruct the original raw waveform of the speech, using two additional feature sets besides encoder features: autoregressive data to predict future samples based on the previous and the identity of the input speaker. The model was primarily used for speech processing, but the concept of ResNet-based architecture is viable for latent space representations.

Another example of deep learning features representation extraction from the raw waveform proposed in Jung et al., 2021. The framework is called AA-SIST. Firstly, the raw waveform is passed to a RawNet2 encoder, which extracts features using predefined filters at the start, and the result is passed to the residual CNN blocks. Then, the spectral and temporal graphs are constructed using graph attention and pooling. Thereafter, they are combined by adding them. Later, heterogeneous stacking graph attention and max graph operation are applied. The result is concatenated and used for further classification. The model's main advantage is its relatively small number of trainable parameters and high performance. But despite the good classification results with the proposed latent space representation, it is hard to train the model effectively in a self-supervised manner.

One of the possible solutions to get more informative latent space representations is solving different self-supervised tasks jointly. In the paper Pascual et al., 2019, a problem-agnostic speech encoder (PASE) was proposed, and in Ravanelli et al., 2020, a refined version, PASE+. The encoder consists of the convolution of the raw input waveform with a set of parameterized $sinc(x) = \frac{sin(x)}{x}$ functions that implement rectangular band-pass filters and 7 ConvNet blocks. In PASE+, a quasi-recurrent neural

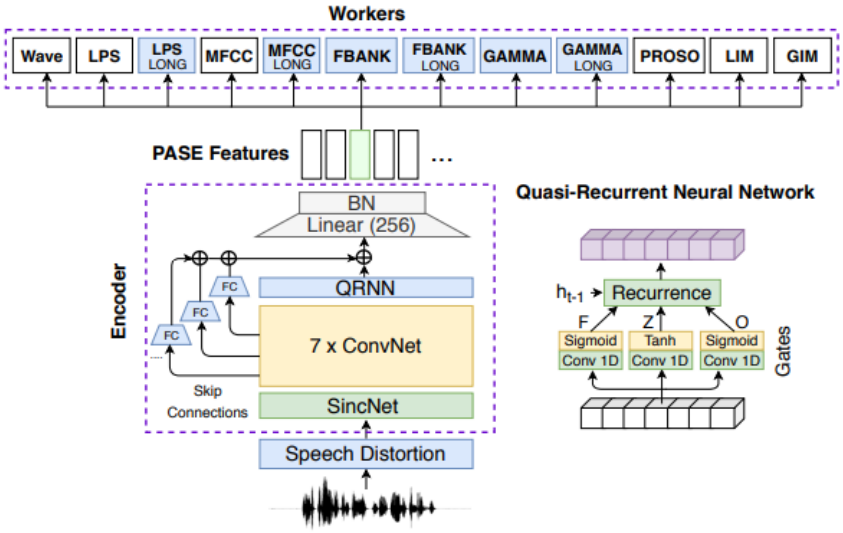



FIGURE 2.1: PASE+ architecture scheme (Source: Ravanelli et al., 2020)

network is also performed to learn long-term dependencies efficiently, and skip connections are introduced to improve gradient flows (see Fig. 2.1). Workers, which are small feed-forward NNs, are fed by the encoded representation and solve multiple self-supervised tasks, defined as regression or binary discrimination tasks. They include original waveform, LPS, MFCC representations reconstruction as regression tasks, local info max, global info max, and sequence predicting coding prediction as binary discrimination tasks (Pascual et al., 2019). Representations obtained with the framework were used to achieve good results in speaker identification, speech emotion classification, and automatic speech recognition tasks.

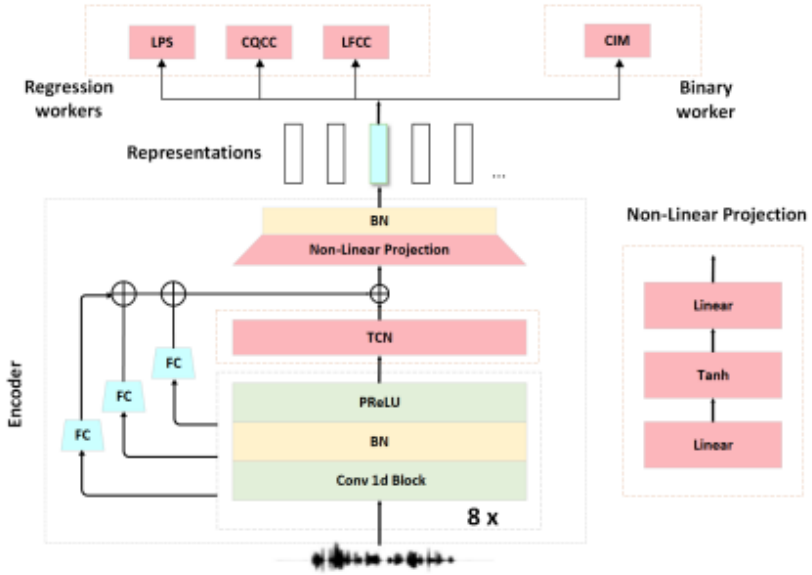



FIGURE 2.2: SSAD architecture scheme (Source: Jiang et al., 2020)

One example of adapting the previous approach was introduced in Jiang et al., 2020. The framework is called a self-supervised spoofing audio detection scheme

(SSAD). The architecture is shown in Fig. 2.2. A temporal convolutional network (TCN) is used to learn long-term dependencies more efficiently. The model has only four workers: LPS, CQCC, and LFCC as regression tasks and congener info max (CIM) binary feature as a binary discriminative task. The trained encoder was used for the spoofing detection task with SENet12, LCNN-big, and LCNN-small classifiers. The SENet integrates the ResNet with the squeeze-and-excision (SE) block. The SE block can adoptively acquire the importance of each feature channel and explicitly model the interdependencies between them by assigning weights to them (Zhang, Wang, and Zhang, 2021).

# Chapter 3

# Data

This chapter describes the main data sources in spoofed audio detection tasks with the selected dataset, data preprocessing steps, and key metrics on which the models are evaluated.

## 3.1 Spoofed audio datasets

Today exists a wide variety of datasets and competitions on audio classification topics. Nevertheless, few consider fake audio detection as the main task. The central challenge in this field remains the Automatic Speaker Verification Spoofing and Countermeasures Challenge (ASVspoof) (Wu et al., 2017).

The database for the first challenge, the ASVspoof 2015 (Wu et al., 2015), contains both bonafide and spoofed speech data collected from 106 speakers (45 male and 61 female). All bonafide speech recordings were gathered in the same conditions. Spoofed speech samples of each speaker were generated artificially using one of ten different, well-known speech synthesis or voice-conversion spoofing-attack algorithms. The dataset was refined for ASVspoof 2019 (Todisco et al., 2019). It consists of two use-case scenarios: logical access (LA) and physical access (PA). The LA scenario involves spoofing attacks injected directly into the ASV system. Attacks in the LA scenario are generated using text-to-speech synthesis (TTS) and voice conversion (VC) technologies. For the PA scenario, speech data is assumed to be captured by a microphone in a physical (Todisco et al., 2019), reverberant space. In ASVspoof 2021 dataset (Delgado et al., 2021), the speech deep fake (DF) scenario was separated from the LA scenario, but the data primarily originates from previous challenges.

Challenges data is partitioned into three datasets: training, development, and evaluation. The three partitions are disjoint in terms of speakers, and the recording prerequisites for all source data are identical. The training and development sets contain spoofing attacks generated with the same algorithms and conditions (Todisco et al., 2019). The evaluation set contains attacks generated with different algorithms or conditions (Todisco et al., 2019).

Our work uses the LA partition of ASVspoof 2019 dataset since it is widely used in the spoofing detection task, which is crucial to compare our results to the existing models. The LA database includes genuine and fake speech data created by 17 different TTS and VC systems. The training data for these systems is obtained from the VCTK database but does not overlap with the data in the 2019 database. Of these 17 systems, 6 are known attacks, while the other 11 are unknown. The training and development sets only include known attacks, but the evaluation set has 2 known and 11 unknown spoofing attacks (Todisco et al., 2019).

Among the 6 known attacks, 2 are VC systems, and 4 are TTS systems. The VC systems use neural-network-based and spectral-filtering-based approaches. The

TTS systems use waveform concatenation or neural-network-based speech synthesis, which employs either a conventional source-filter vocoder or a WaveNet-based vocoder. The 11 unknown systems comprise 2 VC, 6 TTS, and 3 hybrid TTS-VC (Todisco et al., 2019) systems with various waveform generation methods. These methods include classical vocoding, GriffinLim, generative adversarial networks, neural waveform models, waveform concatenation, waveform filtering, spectral filtering, and combinations (Todisco et al., 2019).

There are 2580 bonafide and 22800 spoofed utterances for the training subset, resulting in 25380 samples. Similarly, there are 2548 bonafide and 22296 spoofed utterances for the development subset, resulting in 24844 samples. Each audio file is stored in FLAC format with the same sample rate of 22500 Hz and different signal lengths (see Figure 3.1).

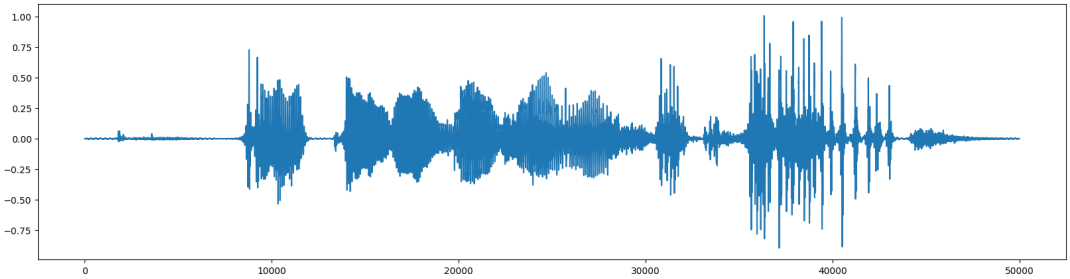



FIGURE 3.1: Audio, which is represented as a normalized signal

## 3.2 Data preprocessing

### 3.2.1 Samples splitting

As the first step, we split all the samples into chunks of 5 seconds in length. If the chunk is smaller, then it is padded with zeros. The main reason for this operation is to feed multiple waveforms simultaneously and, thus, make the processing, primarily during the training, more effective, saturating GPU memory. These chunks are used as the input data, and all the feature representations are calculated from this.

### 3.2.2 Framing and windowing

Before transferring the audio data to 2-dimensional representations, we perform the framing. Framing is a common technique used in speech processing to analyze the speech signal over short, overlapping time intervals. We use the frame length of 20 ms and the hop length of 10 ms.

Since the frames are half-overlapped, we should prevent spectral leakage. Windowing is a technique that helps to reduce this effect by multiplying the signal with a window function before taking the Fourier Transform, removing the edge effects. The Hamming window is a popular window function used for this purpose because it provides a good compromise between main lobe width and side lobe attenuation. The Hamming window has a bell-shaped curve that smoothly tapers the edges of the signal to zero, reducing spectral leakage. We use it to process sample chunks before feature extraction.

### 3.2.3 Features extraction

As features representations, we use different cepstral coefficients. Cepstral features are commonly used in audio spoofing detection because they are effective in capturing the unique properties of the vocal tract and can be used to distinguish between genuine and spoofed speech signals.

For each audio chunk, we extract 20 first coefficients of CQCC (see Figure 3.2), MFCC (see Figure 3.3), IMFCC (see Figure 3.4), and LFCC (see Figure 3.5).

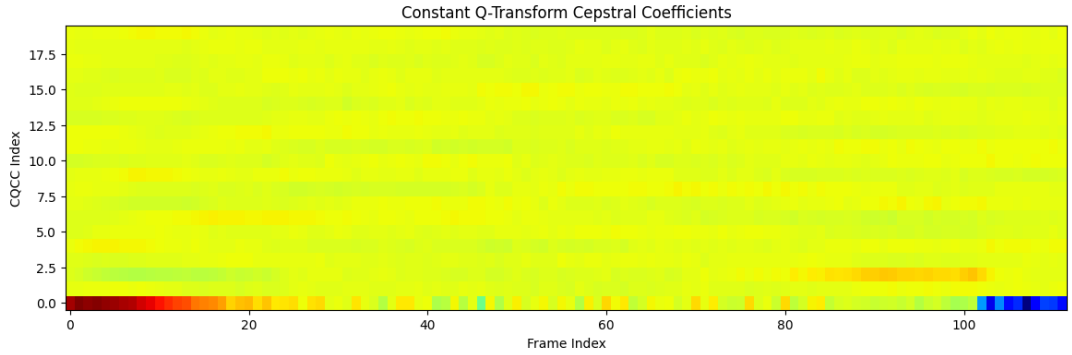



FIGURE 3.2: Example of CQCCs of audio chunk

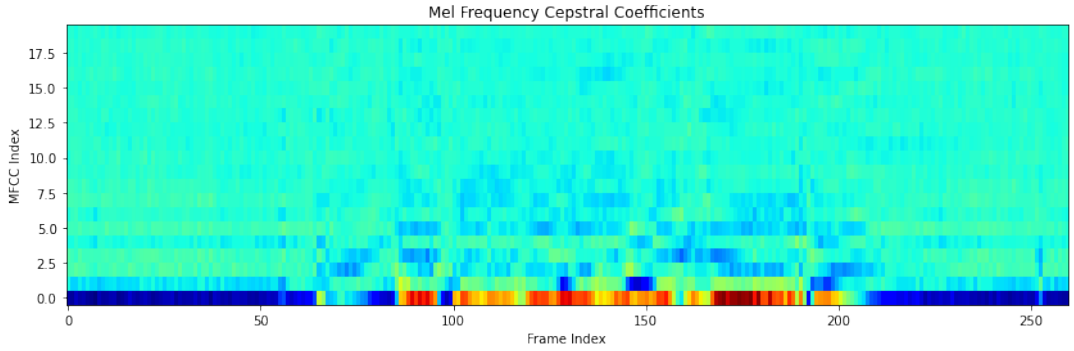



FIGURE 3.3: Example of MFCCs of audio chunk

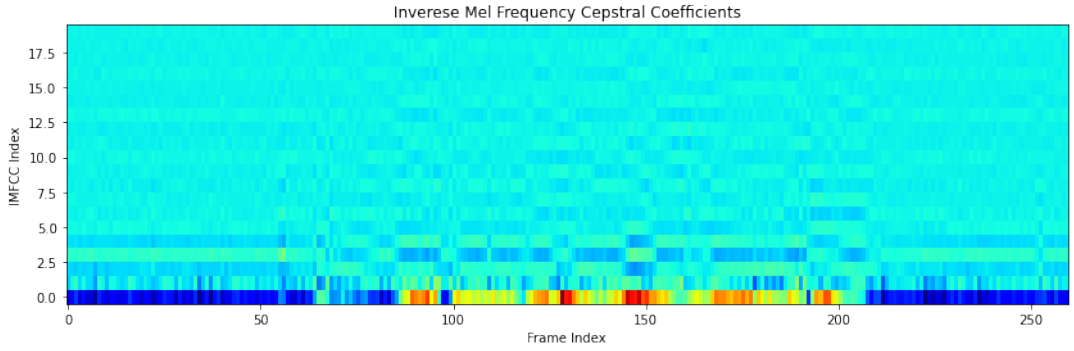



FIGURE 3.4: Example of IMFCCs of audio chunkl

The reason for using the first 20 cepstral coefficients is that they correspond to the lower frequency components of the speech signal, which are more important for capturing the characteristics of the vocal tract. The first few cepstral coefficients typically correspond to the overall amplitude of the speech signal, while higher-order

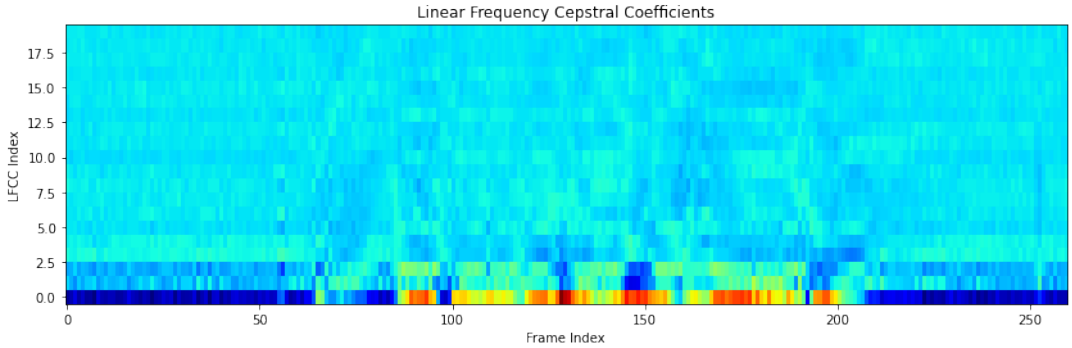



FIGURE 3.5: Example of LFCCs of audio chunk

coefficients capture finer details of the spectral envelope. The higher-order cepstral coefficients generally capture less important information about the spectral envelope and can be more sensitive to noise and other distortions in the speech signal.

# Chapter 4

# Proposed approach

In this chapter, firstly, we define the problem statement. Secondly, we describe the model architecture, key components, and objective functions. Finally, we define the metrics we use to evaluate the results.

## 4.1 Problem statement

The problem is developing an effective and robust audio spoofing detection system that can accurately differentiate between genuine human speech and spoofed waveforms. The goal is to improve the system's efficiency, keeping the system robust and reliable. The audio spoofing detection task involves training a machine learning model that accurately classifies audio samples as genuine or spoofed based on their acoustic characteristics and temporal patterns.

Audio spoofing detection has several challenges that should be handled to build an efficient and reliable system. First, attackers use diverse strategies to create audio spoofs, such as voice conversion, speech synthesis, and impersonation, with physical devices or deepfakes. Each spoofing technique introduces different acoustic artifacts, making capturing and analyzing various spoofing patterns necessary.

The second thing is that collecting a diverse and representative dataset of genuine and spoofed audio samples is challenging in real-world scenarios. The system should be capable of identifying known spoofing methods while being resilient to unknown or novel attacks and simultaneously distinguishing bonafide speech.

Last but not least, audio spoofing detection needs to be performed in real-time. The system should process waveforms efficiently, providing prompt and accurate decisions without introducing excessive latency.

## 4.2 Model architecture

The proposed model combines the encoder, based on the Self-Supervised Audio Detection Scheme (Jiang et al., 2020) and LCNN (Light CNN) based classifier (Lavrentyeva et al., 2017) to create a comprehensive and powerful system for audio spoofing detection. This model aims to effectively differentiate between genuine and spoofed waveforms, ensuring the security and reliability of speaker verification systems.

In the first step of the model, the encoder is used. It is designed to capture waveforms' acoustic characteristics and temporal patterns. It consists of several components that work together to extract encoded compact representations from the input audio based on cepstral representations of the input signal.

Following the encoder, the model incorporates the LCNN-based classifier. LCNN, or Light CNN, is a lightweight convolutional neural network architecture specifically designed for face recognition tasks (Wu et al., 2018). In the proposed


FIGURE 4.1: General pipeline of model for spoofed audio classification

model, LCNN is adapted for the audio spoofing detection task, leveraging its ability to extract high-level discriminative features.

The LCNN-based classifier consists of multiple convolutional layers followed by fully connected layers. These layers learn hierarchical representations of the input audio features, gradually capturing more abstract and discriminative information. The model benefits from the design principles of LCNN, such as lightweight convolutional filters and efficient parameter utilization, which enable it to achieve accurate and efficient classification results. The output of the LCNN-based classifier provides the final prediction of the audio sample, indicating whether it is genuine or spoofed.

By combining the encoder and LCNN-based classifier, the model leverages the strengths of both components. The encoder captures fine-grained acoustic characteristics and temporal patterns, while the LCNN classifier focuses on extracting high-level discriminative features. This multi-stage approach allows the model to learn and differentiate between genuine and spoofed waveforms effectively.

The code of the proposed solution is available on GitHub. [1]

### 4.2.1 Feature extractor

The encoder is an important part of the proposed audio spoofing detection model. It is used to extract the feature representation using neural networks and is trained on cepstral audio representations. It consists of convolutional blocks, residual blocks, a temporal convolutional network (TCN), a non-linear projector, and batch normalization layers that are applied sequentially to obtain the most useful speech characteristics into a more compact representation called embedding. The models were trained on CQCC, LFCC, MFCC, and IMFCC representations of the input speech signal. These features are calculated as described in sections 3.2.2 and 3.2.3. To transform the embeddings into cepstral features, workers are used. The chosen loss function is the average Mean Squared Error (MSE) loss between the workers' output and corresponding cepstral features.

Each input to the neural network is a 3-dimensional tensor with the shape of $B \times 1 \times T$, where $B$ - batch size, $T$ defines the length of the input signal, which is, in our case, 110250 (corresponding to signal with the sampling rate of 22050 Hz and 5 seconds length); input sequence is normalized into the range from -1 to 1. The output tensor has a size of $B \times H \times W$, where $B$ - batch size, $H$ has a size of 128 in our case, and $W$ can vary depending on input waveform length $I$, kernel sizes, strides, padding parameters of convolutions in convolutional and residual blocks, and also, from the number of residual blocks. In our case, $W$ depends on the kernel size of convolutions in residual blocks and the number of residual blocks.

[1] https://github.com/d-ivashchenko/AudioSpoofingDetection

### Convolutional Blocks

The encoder begins with two Convolutional blocks. Each Convolutional block applies a 1-dimensional convolution operation, followed by batch normalization and a parametric rectified linear unit (PReLU) activation:

$$PReLU(x) = \max(0, x) + a \min(0, x) \tag{4.1}$$

where $a$ is a trainable parameter.

These convolutional layers aim to simulate the windowing of the input signal. The first layer has a kernel size of 220 and a stride of 1, which allows applying a trainable smoothing on the window of 1 ms long. Also, it has a changeable parameter of the number of output channels, which varies in our case from 8 to 32 channels. The parameter defines how he second layer has a kernel size of 20 and a stride of 10. Batch normalization layers normalize the activations within each mini-batch, helping to stabilize and accelerate the training process, improving the model's ability to generalize, and reducing the impact of covariate shifts.

### Residual Blocks

Following the Convolutional blocks, the encoder incorporates Residual blocks (ResBlock). They consist of two Convolutional blocks (ConvBlock) connected with skip connections. The skip connections allow the model to capture both shallow and deep contextual information, improving the ability of the model to train.

ConvBlock 1 takes the input data with $I$ channels and applies a 1-dimensional convolution operation. The convolutional layer convolves the input with a kernel of size $K$ and a stride of size $S$. The output channels are increased in $F$ times, so the number of output channels is $I \cdot F$.

ConvBlock 2 operates on the output of ConvBlock 1. It applies a 1-dimensional convolution with the same number of input and output channels of size $I \cdot F$, using a kernel of size $K$ and a stride of 1.

The skip connection preserves the input data by applying a 1-dimensional convolution with a kernel size calculated based on the stride and padding of ConvBlock 1 and stride of size $S$. This ensures that the skip connection has the same number of output channels as ConvBlock 2.

The output of ConvBlock 2 is combined with the output of the skip connection using element-wise addition. This fusion of information allows for the preservation of important features while enabling the flow of gradients during backpropagation. The resulting output is the final output of the ResBlock, containing enriched representations.

We use kernel sizes $K$ from 7 to 11, stride $S = 2$, multiplication factor $F = 2$, and a number of input channels for the first ResBlock $I$ from 8 to 32. For different models, we use 3 or 4 ResBlocks connected sequentially.

### Temporal Convolutional Network (TCN)

After ResBlocks, a Temporal Convolutional Network (TCN) is applied to model the temporal dependencies within the audio. The model (proposed in Bai, Kolter, and Koltun, 2018) consists of multiple TemporalBlocks, where each block applies dilated convolutions to capture long-range temporal patterns. The dilation rate increases exponentially with each block (multiplied by 2), allowing the model to effectively

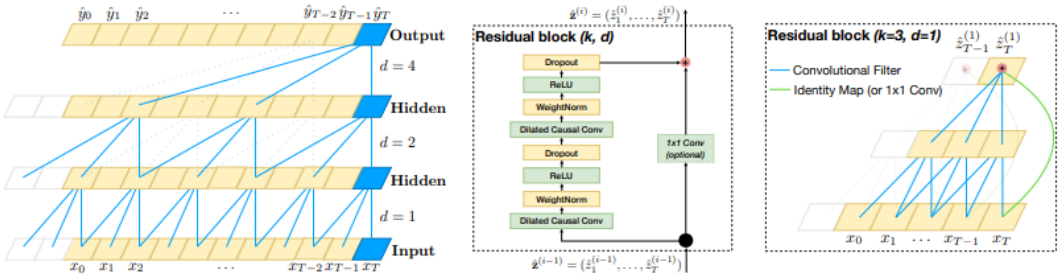



FIGURE 4.2: Architectural elements in a TCN
(Source: Bai, Kolter, and Koltun, 2018, Fig. 1)

capture temporal dependencies at various scales. The TCN aids in learning temporal patterns that are crucial for distinguishing between genuine and spoofed waveforms.

The Temporal Block (4.2) takes a 1-dimensional sequence of data as input and applies a series of operations. The block starts with two 1-dimensional convolutional layers. The first convolutional layer applies a dilated convolution operation to the input sequence. The dilation parameter determines the spacing between the values in the convolution kernel. By increasing the dilation parameter in each subsequent block, the network can capture information over larger receptive fields, allowing it to model long-term dependencies in the data. Weight normalization is applied to these convolutional layers, which helps with stabilizing the learning process.

After each convolutional layer, a Chomp1d operation is performed. Chomp1d trims the rightmost elements from the output of the convolutional layer to ensure that the output sequence has the same length as the input sequence. This operation helps maintain the causality of the network. The ReLU activation function is applied after each Chomp1d operation. ReLU introduces non-linearity and helps the network capture complex patterns in the data. After each ReLU activation, dropout is applied.

Also, a residual connection is established between the output of the second convolutional layer and the block's input. If the number of input channels is different from the number of output channels, a $1 \times 1$ convolution is applied to match the dimensions.

We use convolution kernel size $K = 2$, the number of output channels of 128 or 256, 1 or 2 temporal blocks. Also, we use the dropout rate of 0.2, which means that during training, approximately 20% of the output values of the network's neurons in the specified layer or block will be randomly set to zero.

### Non-Linear Projector

After the TCN, the encoder incorporates a Non-Linear Projector. This module performs non-linear transformations on the output of the TCN. The Non-Linear projection consists of 3 components: 2 linear fully-connected layers with nonlinear activation function - hyperbolic tangent (tanh). The Non-Linear Projector serves multiple purposes, including introducing non-linear mappings and reducing dimensionality. In our case, the output of this layer is the output of the encoder and has the size of $B \times H \times W$, where $B$ is the batch size, $H$ and $W$ are the embedding sizes, $H = 128$, $W$ depends on other parameters of the encoder.

#### Workers

The worker is a small neural network comprising two components: a fully-connected layer and a convolutional layer. It is designed for transforming input data, which is embedding, calculated with an encoder model, with one size into output data with a different size. The fully-connected layer takes the input data and performs a linear transformation to map it to a lower-dimensional space. The convolutional layer then operates on the transformed data, applying a 1-dimensional convolution with a specific kernel size, stride, and padding to fit the needed output size.

Workers are not used for the classification task; their only purpose is to transform 2-dimensional embedding from the encoder model to the corresponding 2-dimensional audio representations, which are cepstral feature representations in our case, during the self-supervised training step. Workers are fed by the encoded representation calculated by the encoder model. Moreover, the encoder is trained with regression tasks via workers: the average mean squared error (MSE) between these small neural networks and corresponding feature representations propagates back to the encoder to generate suitable audio embeddings.

### 4.2.2 Classifier

Our implementation is based on Light CNN (LCNN) model (Lavrentyeva et al., 2017), which is a convolutional neural network architecture designed for efficient and accurate image classification. It consists of multiple convolutional layers, max-feature-map, and max-pooling operations. The model takes input images and applies a series of convolutional and MFM operations to extract hierarchical features. Max pooling is performed to downsample the feature maps. The extracted features are then passed through fully connected layers for classification. The LCNN model is known for its compact structure, computational efficiency, and strong performance in various image classification tasks.

#### Max-Feature-Map operation

Max-Feature-Map (MFM) serves as an activation function, and it is defined as:

$$y_{ij} = \max(x_{ij}^{k}, x_{ij}^{k+\frac{N}{2}}), \forall i = \overline{1,H}, j = \overline{1,W}, k = \overline{1,N/2} \tag{4.2}$$

where $x$ is the input tensor of size $H \times W \times N$, $y$ is the output tensor of size $H \times W \times \frac{N}{2}$, $H$ defines the number of features, $W$ defines the size in temporal domain, $N$ defines the number of channels (see Fig. 4.3). MFM plays the role of feature selector (Lavrentyeva et al., 2017).

#### LCNN architecture

The model takes as input an embedding with size $B \times H \times W$, where $B$ is the batch size, $H$ and $W$ are the embedding sizes, $H = 128$, $W$ depends on other parameters of the encoder. The LCNN-based classifier has the following structure: Conv1 $\rightarrow$ MaxPool $\rightarrow$ Conv2a $\rightarrow$ Conv2b $\rightarrow$ MaxPool $\rightarrow$ Conv3a $\rightarrow$ Conv3b $\rightarrow$ MaxPool $\rightarrow$ Conv4a $\rightarrow$ Conv4b $\rightarrow$ MaxPool $\rightarrow$ FC1 $\rightarrow$ FC2.

The main structural block is an MFM layer consisting of a 2-dimensional convolution with MFM activation. The first MFM layer, Conv1, has one input channel, 16 output channels, a kernel size of 5, stride 1, and padding 2. MFM layers Conv2a,

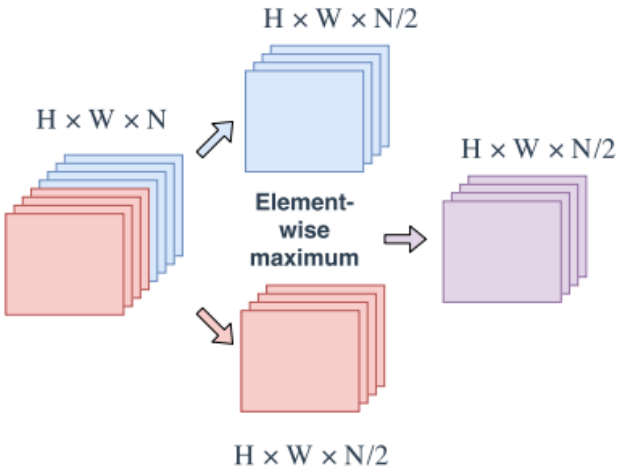



FIGURE 4.3: Max-feature-map activation
(Source: Lavrentyeva et al., 2017, Fig. 3)

Conv3a, and Conv4a take the output feature maps from the previous layer as input with input channels $I$ of 16, 24, and 32, respectively, and perform a $1 \times 1$ convolution operation with output channels $I$, stride $S = 1$ and padding $P = 0$. MFM layers Conv2b, Conv3b, and Conv4b take the output feature maps from the corresponding layers and perform a $3 \times 3$ convolution operation with stride $S = 1$ and padding $P = 1$. The output from these layers has 24, 32, and 16 channels, respectively. After layers Conv1, Conv2b, Conv3b, and Conv4b, max pooling operations are applied with a kernel size of $2 \times 2$ and no padding.

The output feature maps are then flattened and passed through a fully connected MFM layer, FC1, which has an input size determined by the spatial dimensions of the feature maps and outputs an embedding of size $E$, which is, in our case, 64 or 128. Finally, the embedding is fed into a linear layer, FC2, which produces the final classification output of size, corresponding to the number of classes, which is 2 in our case. The weights of FC2 are initialized with a normal distribution with a standard deviation of 0.001.

## 4.3 Loss functions

Along with standard cross-entropy loss with softmax activation, custom loss functions for classifiers are often used in spoofed audio detection tasks to address specific challenges and characteristics of the problem. Spoofed audio detection involves distinguishing between genuine and manipulated/fake audio recordings. This task may have imbalanced class distributions between bonafide and spoofed samples. In such cases, using a softmax can lead to a bias towards the majority class. This bias can result in poor performance in detecting spoofed audio. In our work, we used three different loss functions: softmax, AM-Softmax, and OC-Softmax.

### 4.3.1 Softmax

A Softmax loss for a binary classification task can be defined in the following way:

$$\mathcal{L}_S = \frac{1}{N}\sum_{i=1}^{N}\log(1+e^{(w_{y_i}-w_{1-y_i})^T x_i}) \tag{4.3}$$

where $x_i$ is the embedding vector of the $i$-th sample, $y_i \in \{0,1\}$ is the label of the $i$-th sample, $w_0, w_1$ are weight vectors for bonafide and spoofed speech respectively, $N$ is the number of samples.

### 4.3.2 AM-Softmax

AM-Softmax (Additive Margin Softmax) (Wang et al., 2018) introduces an additive margin term to the standard softmax loss, aiming to increase the angular margin between different classes' feature representations. By incorporating this margin, AM-Softmax encourages more discriminative feature learning, making the decision boundaries between classes more distinct. The AM-Softmax can be defined in the next way:

$$\mathcal{L}_{AMS} = \frac{1}{N}\sum_{i=1}^{N}\log(1+e^{\alpha(m-(\hat{w}_{y_i}-\hat{w}_{1-y_i})^T\hat{x}_i)}) \tag{4.4}$$

where $\hat{x}_i$ is the normalized embedding vector of the $i$-th sample, $y_i \in \{0,1\}$ is the label of the $i$-th sample, $\hat{w}_0, \hat{w}_1$ are normalized weight vectors for bonafide and spoofed speech respectively, $\alpha$ is the scaling factor ($\alpha = 20$ in our case), $m$ is the angular margin ($m \in \{0.8, 0.3, -0.3\}$ in our case), $N$ is the number of samples.

### 4.3.3 OC-Softmax

Training a compact embedding space for bonafide speech in voice spoofing detection is reasonable. However, if we also train a compact embedding space for the spoofing attacks, it may overfit (Zhang, Jiang, and Duan, 2021) to known attacks. The proposed loss function One-class Softmax (OC-Softmax) (Zhang, Jiang, and Duan, 2021) solves this issue by introducing two different margins to obtain more compact both bonafide and spoofed speech representations in latent space simultaneously. The loss function can be formulated as follows:

$$\mathcal{L}_{OCS} = \frac{1}{N}\sum_{i=1}^{N}\log(1+e^{\alpha(m_{y_i}-\hat{w}_0^T\hat{x}_i)(-1)^{y_i}}) \tag{4.5}$$

where $\hat{x}_i$ is the normalized embedding vector of the $i$-th sample, $y_i \in \{0,1\}$ is the label of the $i$-th sample, $\hat{w}_0$ is the normalized weight vector for bonafide speech, $m_0, m_1$ are angular margins for bonafide and spoofed speech respectively ($m_0, m_1 \in \{0.8, 0.3, 0.2, -0.3\}$ in our case), $\alpha$ is the scaling factor ($\alpha = 20$ in our case), $N$ is the number of samples.

## 4.4 Metrics

The equal error rate (EER) is the main statistic we will use to evaluate the results. The EER is the location on a ROC or DET curve where the false acceptance and rejection rates are equal.

Another widely used evaluation metric is tandem detection cost function (t-DCF), proposed in Kinnunen et al., 2019. It includes the costs for missed attacks, false alarms, and prior probabilities for these events. A lower result means better

performance. The evaluation results of the discussed systems are presented in Table 4.1.

TABLE 4.1: Metrics achieved on the evaluation part of the ASVspoof 2019 dataset (LA) with different Spoofing Audio Detection (SAD) systems.

| **SAD system** | **EER (%)** | **t-DCF** |
|---|---|---|
| CQCC + GMM (Alzantot, Wang, and Srivastava, 2019) | 9.57 | 0.2366 |
| LFCC + GMM (Alzantot, Wang, and Srivastava, 2019) | 8.09 | 0.2116 |
| CQCC + ResNet (Alzantot, Wang, and Srivastava, 2019) | 7.69 | 0.2166 |
| MFCC + ResNet (Alzantot, Wang, and Srivastava, 2019) | 9.33 | 0.2042 |
| LFB + ResNet (Chen et al., 2020) | 1.26 | - |
| MFCC + ResNet (pre-trained) (P. et al., 2020) | 5.32 | 0.1514 |
| LFCC + ResNet + OC-Softmax (Zhang, Jiang, and Duan, 2021) | 2.19 | 0.059 |
| LPS + LCNN (Lavrentyeva et al., 2019) | 4.53 | 0.1028 |
| LCNN + transformer (Wu et al., 2020) | 4.07 | 0.102 |
| LC-GRNN (Gomez-Alanis et al., 2019) | 6.28 | 0.1523 |
| AA-SIST (Jung et al., 2021) | 0.83 | 0.0275 |
| SAMO (Ding, Zhang, and Duan, 2022) | 1.08 | 0.0356 |
| SSAD (LCNN-big) (Jiang et al., 2020) | 5.31 | - |

# Chapter 5

# Experiments and Results

This chapter provides a detailed description of the conducted experiments and their results. All the experiments can be naturally divided into two parts: experiments with the encoder and experiments with the classifier. Our goal was to improve the inference of the model, keeping the accuracy as high as possible.

## 5.1 Experiments with encoder

The encoder used in these experiments is based on the SSAD (Jiang et al., 2020) encoder, a specialized architecture designed for detecting speech falsification. Our model also incorporates unsupervised training techniques on various cepstral coefficients, like LFCC, MFCC, IMFCC, and CQCC. The encoder can learn meaningful representations from audio data without relying on labeled annotations by leveraging unsupervised training on these features. This self-supervised approach helps to enhance the encoder's ability to capture relevant features and patterns that distinguish between genuine and spoofed audio.

One of the significant drawbacks is that the mentioned architecture is a complex model with multiple layers and a significant number of parameters. Due to this, training and inference of the SSAD encoder can be computationally intensive and challenging. Thus, the decision was to modify this architecture and reduce the number of layers and parameters. Since the exact parameters of the SSAD model are unknown, the results cannot be reproduced with high accuracy. Nevertheless, we took the starting parameters from the similar PASE encoder (Pascual et al., 2019).

The optimizer used for training the model is Stochastic Gradient Descent (SGD), initialized with a learning rate of 0.001 and a momentum value of 0.9. The training process involves running the model for a total of 10 epochs. The batch size was set to 50. The experiments are performed on Google Colab. The Tesla T4 GPU was utilized. TensorBoard was used for monitoring and visualizing the training progress and performance of the SSAD encoder.

### 5.1.1 Number of channels in ResBlocks

The first parameter we chose to experiment with was the number of channels of 1-dimensional convolutions. They are determined by the number of input channels $I$ for the first ResBlock (see 4.2.1). We proposed an encoder with three residual blocks with a 1-dimensional convolution kernel size of 11 and one temporal block in TCN with 256 output channels for the experiment. Table 5.1 shows the results. The "Number of input channels $I$" column indicates the parameter being evaluated for each model. The "Train average MSE" and "Dev average MSE" columns represent the average mean squared error (MSE) obtained during training and evaluation on the development set on the last epoch, respectively. The "Number of parameters"

column indicates the total number of parameters in each model that are learnable during training.

TABLE 5.1: Comparison of encoders with different numbers of input channels in ResBlocks

| Num. of input channels $I$ | Train average MSE | Dev average MSE | Num. of parameters |
|---|---|---|---|
| 32 | **0.0516** | **0.0555** | 2925064 |
| 16 | 0.0525 | 0.0565 | 936200 |
| 8 | 0.0529 | 0,0557 | 396552 |

Analyzing the results, we can see that the model with 32 input channels reached the lowest MSE values both on training and validation sets. This suggests that a higher number of input channels in the ResBlocks leads to better performance in terms of MSE. However, it is worth noting that the model with 32 input channels also has the highest number of trainable parameters, indicating a higher complexity. On the other hand, reducing the number of input channels to 16 or 8 slightly increases the MSE values but significantly reduces the number of trainable parameters.

The results suggest a trade-off between model complexity (number of parameters) and performance (MSE). From that point of view, it is reasonable to use 16 or 8 channels, keeping in mind the purposes of the research.

### 5.1.2 Number of temporal blocks and output channels in TCN

Another component that can influence the performance and inference speed of the encoder is TCN. We conducted the experiments with two base models. The first model has 3 residual blocks with 8 input channels for the first ResBlock and convolutions with a kernel size of 11. The numerical results of these models are shown in Table 5.2.

TABLE 5.2: Comparison of encoders with different architectures of TCN for the first proposed model

| TCN architecture | Train avg. MSE | Dev avg. MSE | Num. of parameters |
|---|---|---|---|
| 1 temporal block with 256 output channels | 0.0529 | 0.0557 | 396552 |
| 1 temporal block with 128 output channels | 0.0524 | 0.0558 | 256648 |
| 2 temporal blocks with 128 and 128 output channels | **0.0522** | **0.0556** | 322696 |
| 2 temporal blocks with 128 and 256 output channels | 0.0524 | 0.0558 | 503688 |

From the results, we observe that the architecture with two temporal blocks with 128 and 128 output channels achieves the lowest average MSE on both the training and development sets. The model stands out as a favorable choice, providing a good trade-off between accuracy and model complexity.

The second model has 3 residual blocks with 16 input channels for the first ResBlock and convolutions with a kernel size of 11. The metrics for trained models are shown in Table 5.3.

TABLE 5.3: Comparison of encoders with different architectures of TCN for the second proposed model

| TCN architecture | Train avg. MSE | Dev avg. MSE | Num. of parameters |
|---|---|---|---|
| 1 temporal block with 128 output channels | **0.0519** | 0.0562 | 755208 |
| 1 temporal block with 256 output channels | 0.0525 | 0.0565 | 936200 |
| 2 temporal blocks with 256 and 256 output channels | 0.052 | **0.056** | 1199368 |

The results show that the architecture with 1 temporal block and 128 output channels achieves the lowest average MSE on the training set, and the architecture with 2 temporal blocks with 256 and 256 output channels shows the best performance on the development set. However, it is a lot more complex, so it offers a worse balance between accuracy and model complexity. Thus, the first model is preferable for our goals.

### 5.1.3 Number of residual blocks

In this experiment, we tried to explore how the number of residual blocks impacts the encoder model. For this purpose, we took two different base architectures. They both have 8 input channels for the first ResBlock and convolutions with a kernel size of 11, but different TCN layers structures: the first one has 1 temporal block with 128 output channels, and the second one has 2 temporal blocks with 128 and 256 output channels. The results of the experiment are described in Table 5.4.

TABLE 5.4: Comparison of encoders with different numbers of residual blocks

| TCN architecture | Num. of ResBlocks | Train avg. MSE | Dev avg. MSE | Num. of parameters |
|---|---|---|---|---|
| 1 temporal block with 128 output channels | 3 | 0.0524 | 0.0558 | 256648 |
| | 4 | **0.0517** | **0.0557** | 757386 |
| 2 temporal blocks with 128 and 256 output channels | 3 | 0.0524 | 0.0558 | 503688 |
| | 4 | 0.0529 | 0.0562 | 1004426 |

In terms of training average MSE and development average MSE, the results show that increasing the number of residual blocks can lead to better performance. Especially for the first TCN architecture, the model with 4 residual blocks achieved the lowest training and development MSE. On the contrary, for the second TCN architecture, the model with 4 residual blocks did not outperform the model with 3 blocks in terms of MSE. At the same time, as the number of residual blocks increases, the number of trainable parameters also increases drastically, which can impact the inference speed of the model. For these reasons, using 4 residual blocks instead of 3 is not worthwhile.

### 5.1.4 Convolution kernel size

In this experiment, we compare encoders with different kernel sizes of convolutions in the residual blocks to evaluate their impact on the performance of spoofing audio detection. As the base model, we trained an encoder with 3 residual blocks and 16 input channels for the first ResBlock, one temporal convolution layer with a number of output channels 128. Three kernel sizes, 7, 9, and 11, were examined. Table 5.5 shows the results.

TABLE 5.5: Comparison of encoders with different kernel sizes of convolutions in the residual blocks

| Kernel size | Train avg. MSE | Dev avg. MSE | Num. of parameters |
|---|---|---|---|
| 7 | 0.053 | 0.0566 | 497160 |
| 9 | 0.0522 | **0.0555** | 626184 |
| 11 | **0.0519** | 0.0562 | 755208 |

Overall, the experiment demonstrates that the choice of kernel size in the convolutional layers of the residual blocks significantly affects the encoder's performance.

The results show that the encoder with a kernel size of 11 achieves the lowest average MSE on the training set. However, the encoder with a kernel size of 9 shows promising results with a lower average MSE on the development set. Also, we conducted the experiment with the changed number of output channels from the temporal block to 256 and compared it with the previously trained model (Table 5.6), and it has shown the same pattern between the models with a kernel size of 9 and 11.

TABLE 5.6: Comparison of encoders with different kernel sizes of convolutions in the residual blocks with another TCN architecture

| Kernel size | Train avg. MSE | Dev avg. MSE | Num. of parameters |
|---|---|---|---|
| 9 | 0.0527 | **0.056** | 807176 |
| 11 | **0.0525** | 0.0565 | 936200 |

## 5.2 Experiments with classifier

As was mentioned earlier, the classifier model is based on the LCNN model (Lavrentyeva et al., 2017). It was designed for audio or speech-related classification tasks. As input, it takes the embeddings generated from the encoder and learns to identify discriminative features from it to differentiate between genuine and spoofed speech.

Unlike Jiang et al., 2020, in our work, the encoder aims to learn more general audio embeddings. In order to obtain a more discriminative latent space representation for our specific task, we incorporated loss functions such as AM-Softmax (Wang et al., 2018) and OC-Softmax (Zhang, Jiang, and Duan, 2021) in our training process. We encourage the encoder to learn more distinct and separable embeddings in the latent space by introducing additional angular margin constraints. This is particularly beneficial for audio classification tasks.

For classifiers, we used three encoder models, which were trained for 10 epochs in the same setup from the previous section:

- with 3 residual blocks with 32 input channels for the first ResBlock and a 1-dimensional convolution kernel size of 11 and 1 temporal block in TCN with 256 output channels (Encoder 1)
- with 3 residual blocks with 16 input channels for the first ResBlock and a 1-dimensional convolution kernel size of 9 and 1 temporal block in TCN with 128 output channels (Encoder 2)
- with 3 residual blocks with 8 input channels for the first ResBlock and a 1-dimensional convolution kernel size of 11 and 2 temporal blocks in TCN with 128 and 128 output channels (Encoder 3)

The experiment setup for the classifier involved the utilization of different optimizers and settings. Specifically, the LCNN optimizer optimized the Adam model with a learning rate of $10^{-5}, \beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The OC-Softmax and AM-Softmax loss functions employed the SGD optimizer with a learning rate of 0.01. The training process was conducted for 10 epochs. The batch size was set to 25. The experiments are performed on Google Colab. The Tesla T4 GPU was utilized. TensorBoard was used for monitoring and visualizing the training progress and performance of the model.

### 5.2.1 Softmax loss

Our first experiment was training the classifier with the encoder fine-tuning using the softmax loss function. Also, we tried the LCNN embedding size of 128 and 64, which is the size of the output of the FC1 layer of the classifier (see 4.2.2). The results are shown in Table 5.7.

TABLE 5.7: Classifier with Softmax loss performance on validation data

| Encoder | LCNN embedding size | Min Dev EER, % | Num. of parameters |
|---|---|---|---|
| Encoder 1 | 128 | **7.454** | 5717882 |
| Encoder 1 | 64 | 8.684 | 4341626 |
| Encoder 2 | 64 | 8.57 | 2059130 |
| Encoder 3 | 64 | 8.583 | 1739258 |

Based on the performance metrics, Encoder 1, with an LCNN embedding size of 128, achieves the lowest EER. At the same time, that model has the highest number of parameters, which means it is a lot more complex. For that reason, it is better to choose models LCNN embedding size of 64.

### 5.2.2 AM-Softmax loss

In this experiment, we measured the impact of changing the margin between the bonafide and spoofed speech in latent space using the AM-Softmax loss function. The margin parameter determines the separation between different classes in the learned embedding space (Wang et al., 2018). It controls the degree of intra-class compactness and inter-class separability. Choosing an appropriate margin value is crucial to ensure a good balance between class discrimination and model stability. The LCNN embedding size was set to 64. The results are described in Table 5.8.

TABLE 5.8: Classifier with AM-Softmax loss performance on validation data

| Encoder | Margin $m$ | Min Dev EER, % | Num. of parameters |
|---|---|---|---|
| Encoder 1 | 0.8 | 6.88 | 4341754 |
| Encoder 1 | 0.3 | 5.651 | 4341754 |
| Encoder 1 | -0.3 | 7.928 | 4341754 |
| Encoder 2 | 0.8 | 5.516 | 2059258 |
| Encoder 2 | 0.3 | **4.938** | 2059258 |
| Encoder 2 | -0.3 | 8.648 | 2059258 |
| Encoder 3 | 0.8 | 8.436 | 1739386 |
| Encoder 3 | 0.3 | 8.404 | 1739386 |
| Encoder 3 | -0.3 | 7.458 | 1739386 |

For Encoder 2, a margin value of 0.3 achieves the lowest minimum development EER of 4.938%. Similarly, for Encoder 1, a margin value of 0.3 reaches the minimum EER of 5.651%. This indicates that the model can better separate the classes in the learned embedding space with a smaller margin, leading to improved discrimination between spoofed and genuine audio samples. However, as the margin value deviates further from this optimal value, the performance worsens with higher EER values. This could indicate that the latent space should have more dimensions to have the ability to separate the classes with a desired margin.

### 5.2.3 OC-Softmax loss

Another loss function we considered was the OC-Softmax. This function was developed for this task, and the main ideas were to introduce two different margins for both classes and classify whether the sample belongs to bonafide speech or not Zhang, Jiang, and Duan, 2021. We assumed that with this idea, the classifier would be more flexible and less likely to overfit on known spoofing algorithms. The results are shown in Table 5.9. $m_0$ corresponds to genuine speech margin and $m_1$ to spoofed speech margin.

TABLE 5.9: Classifier with OC-Softmax loss performance on validation data

| Encoder | $m_0$ | $m_1$ | Min Dev EER, % | Num. of parameters |
|---|---|---|---|---|
| Encoder 1 | 0.8 | 0.2 | 7.565 | 4341690 |
| Encoder 1 | 0.3 | -0.3 | 5.118 | 4341690 |
| Encoder 2 | 0.8 | 0.2 | 4.689 | 2059194 |
| Encoder 2 | 0.3 | -0.3 | **4.652** | 2059194 |
| Encoder 3 | 0.8 | 0.2 | 7.537 | 1739324 |
| Encoder 3 | 0.3 | -0.3 | 7.277 | 1739324 |

The analysis suggests that OC-Softmax loss, particularly with margin values of $m_0 = 0.3$ and $m_1 = -0.3$, enhances the classifier's ability to discriminate between genuine and spoofed audio samples. Encoder 2 with these margin values appears to be the most effective configuration, delivering the lowest minimum development EER.

## 5.3 Model evaluation

We found the parameters and models that are less complex and perform well from the preceding experiments. We took Encoder 1 and Encoder 2, margin $m = 0.3$ for AM-Softmax, and margins $m_0 = 0.3, m_1 = -0.3$ for OC-Softmax. The classifiers were trained for 40 epochs instead of 10, with the same optimization parameters but with 50% learning rate decay every 10 epochs. For evaluation, the models with the lowest development EER were selected. In Table 5.10, the results of the evaluation are given. The last column denotes the average classification time on the one chunk of the length of 5 seconds. The measurement was performed with a batch size of 1 and averaged all samples from the evaluation set.

TABLE 5.10: Performance of the models on development and evaluation sets

| Encoder | Loss | Epoch | Dev EER, % | Eval EER, % | Avg. classification time, ms |
|---|---|---|---|---|---|
| Encoder 1 | Softmax | 33 | 6.27 | 12.8 | 3.89 |
| Encoder 1 | AM-Softmax | 12 | 5.73 | 13.71 | 6.26 |
| Encoder 1 | OC-Softmax | 34 | 4.1 | 12.12 | 6.75 |
| Encoder 2 | Softmax | 11 | 7.53 | 14.55 | 3.87 |
| Encoder 2 | AM-Softmax | 26 | **4.0** | **12.06** | 6.25 |
| Encoder 2 | OC-Softmax | 40 | 4.72 | 12.55 | 6.54 |

In the case of Encoder 2, the AM-Softmax loss demonstrates the best performance on both the development and evaluation sets with EERs of 4.0% and 12.06%, respectively. This indicates that the AM-Softmax loss not only improves the performance

on the development set but also generalizes well to unseen data. It achieves the lowest EER among all the models, suggesting that the AM-Softmax loss helps in learning more discriminative and robust representations for spoofed audio detection. However, further analysis is necessary to understand the generalization capabilities of the models and investigate overfitting issues.

It is worth noting that the number of epochs required for convergence varies across the models and loss functions. Models with OC-Softmax loss took the highest number of epochs to converge: 34 for Encoder 1 and 40 for Encoder 2.

Another thing is that the introduced loss functions increase the classification quality relative to Softmax loss. At the same time, these functions do not affect the inference speed of the model significantly, they all show near real-time performance.

An interesting point is how the encoder learns to extract discriminative features from the input signal via the classifier training. In Fig. 5.1 we see the representation from the Encoder 2 after unsupervised training during the 10 epochs. In Fig. 5.2 we can see the same representation from Encoder 2 after supervised training with the LCNN classifier and AM-Softmax loss function during the 40 epochs, and the starting initialization of the model was with weights, trained via unsupervised learning for 10 epochs. It is worth mentioning that there are common patterns, but the representation changed significantly during the training.

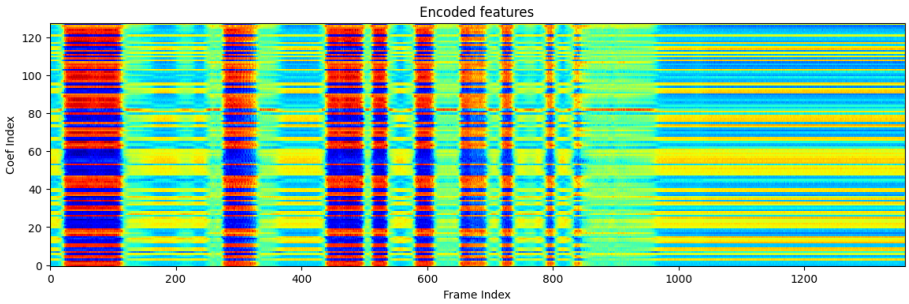



FIGURE 5.1: Features representation of Encoder 2 after unsupervised training

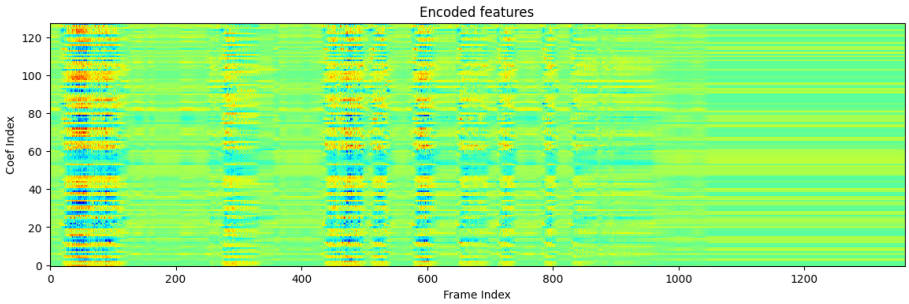



FIGURE 5.2: Features representation of Encoder 2 after fine-tuning with the classifier

# Chapter 6

# Conclusion

## 6.1 Discussion and future work

The choice of encoder architecture plays a crucial role in the performance of the models. Encoder 2 demonstrates better results than Encoder 1 and Encoder 3 across all the loss functions. Nevertheless, it is less complex than Encoder 1. This highlights the importance of designing encoder architectures to capture and preserve the relevant audio features necessary for spoofed audio detection.

Further investigation and experimentation with different encoder architectures can improve performance. Different audio representations, such as LPS or Gamma-Tone Cepstral Coefficients (GTCC), could be used for training the encoder. There can be a lot of improvements with TCN usage for better capturing the temporal dependencies in the audio data. The hyperparameters, such as the size of the kernel of the temporal block, dropout rate, number of temporal blocks, and output channels, can be optimized to outperform the proposed models. Additionally, the classifier can be improved by introducing additional MFM and pooling layers to reduce the dimensionality of the learned representations.

The results of the experiments indicate the effectiveness of different loss functions and encoders in spoofed audio detection. The analysis shows that the choice of the loss function can significantly impact the performance of the models. The AM-Softmax and OC-Softmax losses consistently outperform the Softmax loss, achieving the lowest EER on both the development and evaluation sets. This suggests that the AM-Softmax and OC-Softmax losses help learn more discriminative and robust representations for distinguishing between genuine and spoofed audio. A better hyperparameter optimization can lead to potential quality improvements for these loss functions.

It is worth mentioning that while the models perform well on the development set, there is a slight drop in performance on the evaluation set. This suggests some overfitting to the development set or lacking generalization to unseen data. Future work could address these challenges by employing data augmentation or using other datasets of human speech to enrich the sample.

## 6.2 Conclusion

This paper focused on audio spoofing detection's important and challenging task. We have gained valuable insights into the state-of-the-art methods employed in this field through an analysis of various techniques and approaches.

We started by investigating the characteristics of audio spoofing attacks and the potential vulnerabilities of automatic speaker verification systems. This foundational knowledge provided the necessary context for developing effective countermeasures against spoofing attempts.

After that, we explored the use of various features, such as CQCC, LFCC, MFCC, and IMFCC, and assessed their effectiveness in capturing relevant information for spoofing detection. Based on the previous study, we developed encoder and classifier architectures for effective spoofing detection. Additionally, we analyzed the impact of different loss functions, such as Softmax, AM-Softmax, and OC-Softmax, on the performance of the spoofing detection classifiers. These findings highlighted the importance of carefully selecting and fine-tuning the loss function to enhance the discriminative power of the models.

Throughout this thesis, we also considered the computational efficiency of the proposed systems, recognizing the need for real-time or near-real-time deployment in practical scenarios. Therefore, we optimized our models and algorithms to achieve a balance between detection accuracy and computational speed, making them suitable for real-world applications. Considerable attention was also paid to the problem of the lack of labeled data and semi-supervised approaches to address this issue. Overall, the results obtained from our experiments and evaluations demonstrated the efficacy of the developed audio spoofing detection systems.

In conclusion, this thesis has advanced our understanding of audio spoofing attacks and their detection and laid the groundwork for future research and development in this area. We hope that the insights and methodologies presented here will lead to the development of the research field, giving the opportunity to improve the existing techniques.

# Bibliography


Alzantot, Moustafa Farid, Ziqi Wang, and Mani B. Srivastava (2019). "Deep Residual Neural Networks for Audio Spoofing Detection". In: *Interspeech*.

Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun (2018). *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. arXiv: 1803.01271 [cs.LG].

Brown, Judith (Jan. 1991). "Calculation of a Constant Q Spectral Transform". In: *Journal of the Acoustical Society of America* 89, pp. 425–. DOI: 10.1121/1.400476.

Chakroborty, Sandipan, Anindya Roy, and Goutam Saha (2008). "Improved Closed Set Text-Independent Speaker Identification by combining MFCC with Evidence from Flipped Filter Banks". en. In: 4.2.

Chen, Tianxiang et al. (Nov. 2020). "Generalization of Audio Deepfake Detection". en. In: *The Speaker and Language Recognition Workshop (Odyssey 2020)*. ISCA, pp. 132–137. DOI: 10.21437/Odyssey.2020-19. URL: https://www.isca-speech.org/archive/odyssey_2020/chen20_odyssey.html (visited on 01/18/2023).

Chorowski, Jan et al. (Dec. 2019). "Unsupervised speech representation learning using WaveNet autoencoders". en. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.12. arXiv:1901.08810 [cs, eess, stat], pp. 2041–2053. ISSN: 2329-9290, 2329-9304. DOI: 10.1109/TASLP.2019.2938863. URL: http://arxiv.org/abs/1901.08810 (visited on 01/19/2023).

Davis, S. and P. Mermelstein (1980). "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.4, pp. 357–366. DOI: 10.1109/TASSP.1980.1163420.

Delgado, Héctor et al. (Sept. 2021). *ASVspoof 2021: Automatic Speaker Verification Spoofing and Countermeasures Challenge Evaluation Plan*. en. Tech. rep. arXiv:2109.00535. arXiv:2109.00535 [cs, eess] type: article. arXiv. URL: http://arxiv.org/abs/2109.00535 (visited on 01/15/2023).

Ding, Siwen, You Zhang, and Zhiyao Duan (Nov. 2022). *SAMO: Speaker Attractor Multi-Center One-Class Learning for Voice Anti-Spoofing*. en. Tech. rep. arXiv:2211.02718. arXiv:2211.02718 [cs, eess] type: article. arXiv. URL: http://arxiv.org/abs/2211.02718 (visited on 01/18/2023).

Gomez-Alanis, Alejandro et al. (Sept. 2019). "A Light Convolutional GRU-RNN Deep Feature Extractor for ASV Spoofing Detection". en. In: *Interspeech 2019*. ISCA, pp. 1068–1072. DOI: 10.21437/Interspeech.2019-2212. URL: https://www.isca-speech.org/archive/interspeech_2019/gomezalanis19_interspeech.html (visited on 01/19/2023).

Gupta, Harshita and Divya Gupta (Jan. 2016). "LPC and LPCC method of feature extraction in Speech Recognition System". In: pp. 498–502. DOI: 10.1109/CONFLUENCE.2016.7508171.

Hanilci, Cemal and Tomi Kinnunen (n.d.). "Classifiers for Synthetic Speech Detection: A Comparison". en. In: ().

Jiang, Ziyue et al. (Oct. 2020). "Self-Supervised Spoofing Audio Detection Scheme". en. In: *Interspeech 2020*. ISCA, pp. 4223–4227. DOI: 10.21437/Interspeech.2020-

1760. URL: https://www.isca-speech.org/archive/interspeech_2020/jiang20b_interspeech.html (visited on 01/19/2023).

Jung, Jee-weon et al. (Oct. 2021). *AASIST: Audio Anti-Spoofing using Integrated Spectro-Temporal Graph Attention Networks*. en. Tech. rep. arXiv:2110.01200. arXiv:2110.01200 [cs, eess] type: article. arXiv. URL: http://arxiv.org/abs/2110.01200 (visited on 01/19/2023).

Kinnunen, Tomi et al. (Apr. 2019). *t-DCF: a Detection Cost Function for the Tandem Assessment of Spoofing Countermeasures and Automatic Speaker Verification*. en. Tech. rep. arXiv:1804.09618. arXiv:1804.09618 [cs, eess, stat] type: article. arXiv. URL: http://arxiv.org/abs/1804.09618 (visited on 01/15/2023).

Lavrentyeva, Galina et al. (Aug. 2017). "Audio Replay Attack Detection with Deep Learning Frameworks". en. In: *Interspeech 2017*. ISCA, pp. 82–86. DOI: 10.21437/Interspeech.2017-360. URL: https://www.isca-speech.org/archive/interspeech_2017/lavrentyeva17_interspeech.html (visited on 01/19/2023).

Lavrentyeva, Galina et al. (Sept. 2019). "STC Antispoofing Systems for the ASVspoof2019 Challenge". en. In: *Interspeech 2019*. ISCA, pp. 1033–1037. DOI: 10.21437/Interspeech.2019-1768. URL: https://www.isca-speech.org/archive/interspeech_2019/lavrentyeva19_interspeech.html (visited on 01/19/2023).

Liu, Songxiang et al. (2019). *Adversarial Attacks on Spoofing Countermeasures of automatic speaker verification*. arXiv: 1910.08716 [eess.AS].

Logan, Beth (Nov. 2000). "Mel Frequency Cepstral Coefficients for Music Modeling". In: *Proc. 1st Int. Symposium Music Information Retrieval*.

Muda, Lindasalwa, Mumtaj Begam, and I Elamvazuthi (2010). "Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques". en. In: 2.3.

O'Shaughnessy, Douglas (1987). *Speech communication: human and machine*. Addison-Wesley.

P., Rahul T. et al. (Aug. 2020). *Audio Spoofing Verification using Deep Convolutional Neural Networks by Transfer Learning*. en. Tech. rep. arXiv:2008.03464. arXiv:2008.03464 [cs, eess] type: article. arXiv. URL: http://arxiv.org/abs/2008.03464 (visited on 01/18/2023).

Pascual, Santiago et al. (Apr. 2019). *Learning Problem-agnostic Speech Representations from Multiple Self-supervised Tasks*. en. Tech. rep. arXiv:1904.03416. arXiv:1904.03416 [cs, eess, stat] type: article. arXiv. URL: http://arxiv.org/abs/1904.03416 (visited on 01/19/2023).

Rabiner, L.R. and B.H. Juang (1993). *Fundamentals of Speech Recognition*. Prentice-Hall Signal Processing Series: Advanced monographs. PTR Prentice Hall. ISBN: 9780130151575. URL: https://books.google.com.ua/books?id=XEVqQgAACAAJ.

Ravanelli, Mirco et al. (Apr. 2020). *Multi-task self-supervised learning for Robust Speech Recognition*. en. Tech. rep. arXiv:2001.09239. arXiv:2001.09239 [cs, eess] type: article. arXiv. URL: http://arxiv.org/abs/2001.09239 (visited on 01/19/2023).

Sahidullah, Md., Tomi Kinnunen, and Cemal Hanilçi (Sept. 2015). "A comparison of features for synthetic speech detection". en. In: *Interspeech 2015*. ISCA, pp. 2087–2091. DOI: 10.21437/Interspeech.2015-472. URL: https://www.isca-speech.org/archive/interspeech_2015/sahidullah15_interspeech.html (visited on 01/17/2023).

Sharma, Garima, Kartikeyan Umapathy, and Sridhar Krishnan (Jan. 2020). "Trends in audio signal feature extraction methods". en. In: *Applied Acoustics* 158, p. 107020. ISSN: 0003682X. DOI: 10.1016/j.apacoust.2019.107020. URL: https:

//linkinghub.elsevier.com/retrieve/pii/S0003682X19308795 (visited on 01/17/2023).

Todisco, Massimiliano, Héctor Delgado, and Nicholas Evans (Sept. 2017). "Constant Q cepstral coefficients: A spoofing countermeasure for automatic speaker verification". en. In: *Computer Speech & Language* 45, pp. 516–535. ISSN: 08852308. DOI: 10.1016/j.csl.2017.01.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S0885230816303114 (visited on 01/16/2023).

Todisco, Massimiliano et al. (Sept. 2019). "ASVspoof 2019: Future Horizons in Spoofed and Fake Audio Detection". en. In: *Interspeech 2019*. ISCA, pp. 1008–1012. DOI: 10.21437/Interspeech.2019-2249. URL: https://www.isca-speech.org/archive/interspeech_2019/todisco19_interspeech.html (visited on 01/15/2023).

Wang, Feng et al. (2018). "Additive Margin Softmax for Face Verification". In: *IEEE Signal Processing Letters* 25.7, pp. 926–930. DOI: 10.1109/lsp.2018.2822810. URL: https://doi.org/10.1109%2Flsp.2018.2822810.

Wu, Xiang et al. (2018). *A Light CNN for Deep Face Representation with Noisy Labels*. arXiv: 1511.02683 [cs.CV].

Wu, Zhenzong et al. (Sept. 2020). *Light Convolutional Neural Network with Feature Genuinization for Detection of Synthetic Speech Attacks*. en. Tech. rep. arXiv:2009.09637. arXiv:2009.09637 [eess] type: article. arXiv. URL: http://arxiv.org/abs/2009.09637 (visited on 01/19/2023).

Wu, Zhizheng et al. (Sept. 2015). "ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge". en. In: *Interspeech 2015*. ISCA, pp. 2037–2041. DOI: 10.21437/Interspeech.2015-462. URL: https://www.isca-speech.org/archive/interspeech_2015/wu15e_interspeech.html (visited on 01/15/2023).

Wu, Zhizheng et al. (June 2017). "ASVspoof: The Automatic Speaker Verification Spoofing and Countermeasures Challenge". en. In: *IEEE Journal of Selected Topics in Signal Processing* 11.4, pp. 588–604. ISSN: 1932-4553, 1941-0484. DOI: 10.1109/JSTSP.2017.2671435. URL: http://ieeexplore.ieee.org/document/7858696/ (visited on 01/15/2023).

Zhang, You, Fei Jiang, and Zhiyao Duan (2021). "One-class Learning Towards Synthetic Voice Spoofing Detection". en. In: *IEEE Signal Processing Letters* 28. arXiv:2010.13995 [cs, eess], pp. 937–941. ISSN: 1070-9908, 1558-2361. DOI: 10.1109/LSP.2021.3076358. URL: http://arxiv.org/abs/2010.13995 (visited on 01/18/2023).

Zhang, Yuxiang, Wenchao Wang, and Pengyuan Zhang (Aug. 2021). "The Effect of Silence and Dual-Band Fusion in Anti-Spoofing System". en. In: *Interspeech 2021*. ISCA, pp. 4279–4283. DOI: 10.21437/Interspeech.2021-1281. URL: https://www.isca-speech.org/archive/interspeech_2021/zhang21da_interspeech.html (visited on 01/19/2023).