Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Trym Skogseth

Deep learning solution for automatic audio copy move forgery detection

Mel spectrogram-based CNN

Master's thesis in Information Security
Supervisor: Kyle Andrew Porter
June 2024

Trym Skogseth

Deep learning solution for automatic audio copy move forgery detection

Mel spectrogram-based CNN

Master's thesis in Information Security
Supervisor: Kyle Andrew Porter
June 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Abstract

The authenticity of audio recordings is becoming more important in forensic investigations as digital media becomes increasingly prevalent. This masters thesis explores the development of automated solutions for Audio Authenticity Analysis (AAA), focusing on identifying and distinguishing between original audio files and audio copy-move forgeries. The study applies a Convolutional Neural Network (CNN) to the field of audio forensics. By converting audio signals into visual representations, such as Mel spectrograms, and providing them as input to the CNN model, the system can be trained to effectively recognise patterns indicative of manipulations such as copy-move forgeries.

A novel dataset was created for this study, derived from the NB Tale speech database, consisting of both copy-move forgeries and original files. This was used to train and test the performance of the model. The results demonstrate that the CNN model, enhanced with dynamic data augmentation and regularisation techniques, can effectively detect forgeries in short audio clips. However, challenges remain in detecting forgeries in longer audio files where copied and pasted segments are temporally distant.

This research contributes to the evolving field of digital forensics by, developing a system for copy-move forgery detection and providing a Norwegian copy-move forgery dataset. The findings lay the groundwork for future advancements in the automatic analysis of audio authenticity, highlighting the need for further research to improve the model's robustness and applicability in real-world forensic scenarios.

Sammendrag

Autentisiteten til lydopptak er viktig i kriminaltekniske undersøkelser ettersom digitale medier blir stadig mer utbredt. Denne masteroppgaven utforsker utviklingen av automatiserte løsninger for Audio Authenticity Analysis (AAA), med fokus på å identifisere copy-move forfalskninger (CMF) i lydfiler. Oppgaven bruker en maskinlæringsteknikk, spesifikt et Convolutional Neural Network (CNN). Ved å konvertere lydfiler til Mel-spektrogrammer, og bruke de som input til CNN-modellen, kan systemet lære seg å gjenkjenne mønstre som indikerer manipulasjoner som CMF.

Et nytt datasett ble laget for denne oppgaven, basert på lydfiler fra NB-tale datasettet. Datasettet består av forfalskninger og originalfiler. Dette datasettet ble brukt til å trene og teste modellen. Resultatene viser at CNN-modellen effektivt kan oppdage forfalskninger i korte lydklipp. Betydlige utfordringer gjenstår fortsatt knyttet til å oppdage forfalskninger i lengre lydfiler der de kopierte og limte segmentene potensielt ligger langt fra hverandre i lydfilen.

Masteroppgaven bidrar til utviklingen innen digital etterforskning ved å utvikle et system for å oppdage CMF og et Norsk CMF-datasett. Funnene legger grunnlaget for fremtidige fremskritt innen automatisk analyse av lydautentisitet, og fremhever behovet for ytterligere forskning for å forbedre modellens robusthet og anvendelse i kriminalteknisk praksis.

Contents

Figures

Tables

Acronyms

AAA Audio Authenticity Analysis. iv, 1–4, 8, 11, 19, 50

AEM Absolute Error Map. 19

ANN Artificial Neural Network. 1, 11, 22

CMF copy-move forgery. iv, 3, 8, 44, 47, 50, 51

CMFD copy-move forgery detection. 3, 44, 50, 51

CNN Convolutional Neural Network. iv, 1, 4, 5, 11, 15–17, 21, 22, 26, 44, 45, 47, 49–51, 53

DFT Discrete Fourier Transform. 6

ENF Electric Network Frequency. 8, 10, 19

LBP Local Binary Point. 20

LPC Linear Prediction Coefficient. 22

MFCC Mel Frequency Cepstral Coefficient. 22

ML Machine Learning. 1, 11, 17, 26, 50

SNR Signal-to-Noise Ratio. 10, 33, 34, 40

STFT Short Time Fourier Transform. 6, 7

VAD Voice-Activity Detection. 20, 22, 27, 36, 43–47

Chapter 1

Introduction

1.1 Topic

Smartphones and social media are providing people both the ability and incentive to record and share multimedia with one another. The barrier to entry for media editing has also been lowered, allowing anyone with a smartphone to easily alter media files without professional training. Any edits made to a file compromises its authenticity and integrity. Editing media files can serve multiple purposes, such as editing an audio file of a conversation to change the conveyed message. Edited audio files can also threaten biometric data like voices, which are used to access numerous digital applications [1]. Edited audio files with intent to obfuscate or alter the data within an audio file are referred to as audio forgeries.

As media files are a big part of our daily lives, many criminal investigations have audio or video files associated with them. If the media files should be used as evidence in court, there can be no question to their authenticity and integrity [2], i.e. forensic analysts need methods to distinguish between original files and forgeries. This is the basis of the field of Audio Authenticity Analysis (AAA). Traditionally, audio analysis relies heavily on manual inspection and signal processing techniques [2]. The growth in volume of media files potentially relevant to investigations creates a need for tools to streamline this process and make it more effective and efficient. The significance of automatic AAA is further reinforced by the recent rise in prevalence of deepfake audio. However, this project primarily targets detecting more basic audio manipulations, including edits like cuts, splices, and copy-move forgeries. Despite being less sophisticated, these techniques pose significant challenges in digital forensics due to their widespread use and effectivity.

The evolution of Machine Learning (ML) and emergence of advanced Artificial Neural Networks (ANNs) such as Convolutional Neural Networks (CNNs) are revolutionising this field. By transforming audio files into a visual domain via spectrograms, we can leverage the pattern recognition abilities of CNNs. Automatic AAA is improving to the point where it might be able to reduce the time required for analysis, while still achieving a high enough accuracy to be used in real-life cases. This could greatly benefit forensic analysts looking to save time on the AAA process.

1.2 Keywords

CCS concepts:

- Security and privacy
- Applied computing →Computer forensics →Investigation techniques
- Applied computing →Computer forensics →Evidence collection, storage and analysis
- Computing methodologies →Machine learning →Machine learning approaches →Neural networks

Other keywords:

- Audio authenticity analysis
- Audio forensics
- Audio forgery
- Copy-move forgery
- Deep learning
- Digital forensics

1.3 Problem description

Audio forensics is divided into active and passive forensics, with active forensics using techniques like watermarks and signatures to ensure authenticity, while passive forensics relies on the properties of the audio itself without additional information [3]. For example, in active forensics, a audio recording can be made with a recorder that generates an unique hash as the recording stops. Passive forensics, lacking embedded information, presents greater challenges for detection. Unless a file is created with specialised equipment, it is currently impossible to prove its originality. In such cases, attempts to find indications of forgery are done manually by forensic audio analysts, searching the files for various artefacts. This is both time-consuming and tedious labour; a five-minute audio file might take up to six hours to thoroughly examine [1]. These indications might be audio artefacts such as: absence of the sound of a button press or screen touch to start or stop the recording; sudden changes or stops in background noise; contextual clues such as illogical or unfinished sentences; similar or identical patterns in the spectrogram, which will never occur naturally [4].

Past research on deep learning methods for detecting copy-move forgeries is based on training on short audio files where one segment has been copied and pasted at a different location in the audio file. A significant gap in the literature is the exclusion of testing the deep learning methods on held out validation/test

[1]This is an experience given anecdotally from a representative of Oslo Police District.

data; only one study from this year addresses the validation accuracy of their system [1]. There are also no mentions of how to apply the methods to longer audio files in the related work. In a realistic forensic scenario, recordings being analysed can span multiple minutes.

Furthermore, there is a lack of available datasets to compare different methods for detecting copy-move forgeries. One dataset is available in Arabic [5], but no others have been found. In this project, a novel dataset containing original and copy-move forgeries of Norwegian speech is generated to address this issue.

1.4 Justification, motivation and benefits

Voice data is increasingly used in various industries, including health, but modifications to audio in contexts such as terrorism and criminal cases necessitate robust authentication processes [1]. Audio files can be of importance to an investigation, but without a reliable method to verify the integrity of these files automatically, forensic audio analysts spend significant time assessing the authenticity of audio files. Audio editing software and techniques are widely available and accessible, enabling anyone to make simple edits to audio, such as copying and moving a segment of the audio, or splicing in a segment from another audio file, to alter the meaning of the recording [5]. This creates a need for a system to detect audio copy-move forgery (CMF), a type of forgery where segments of audio are moved within the same file, compromising the recording's integrity and authenticity.

Adversaries who create audio CMFs may use various post-processing operations to hide the forgery and make it harder to detect by, e.g., comparing segments of the waveform of the audio to find similar patterns [3]. These post-processing techniques include noise addition, filtering, compression, frequency scaling, pitch shifting, and re-sampling. Operations such as these can alter the audio signal in ways that obscure the forgery artefacts, posing additional challenges for forensic analysts. The adversaries could also apply various effects only to the copied segment, altering it in some way. Another potential benefit of applying deep-learning methods to AAA is that a trained model may be able to recognise similar patterns even in cases where either the entire file or just the pasted segment has been altered.

The primary stakeholder for this thesis is OPD. Their current methods for AAA are reportedly slow and tedious, and automating parts of the process would increase efficiency of the daily operations of the police. This could allow them to shift their resources to other tasks, which potentially would reduce the time they use on investigating a case. The National Criminal Investigation Service (KRIPOS) and other departments/districts of the Norwegian police are also stakeholders for the project, by extension. The broader field of research on copy-move forgery detection (CMFD) may also benefit from the findings of the thesis.

1.5 Research questions

(RQ1) To what extent can Mel spectrogram-based CNN models be applied to copy-move forgery detection in realistic forensic scenarios?

(RQ2) What are the limitations of Mel spectrogram-based CNN models for copy-move forgery detection in forensic scenarios?"

1.6 Contributions

This thesis aims to contribute to the fields of digital forensics and AAA by addressing the gaps identified in the previous sections. It presents the creation and release of a labelled dataset of original and forged Norwegian speech files, which is used for training and testing deep-learning methods for copy-move forgery detection. Additionally, it details the methodology for utilising a CNN model trained on Mel spectrograms and explores its potential application in realistic forensic scenarios. The results and subsequent analysis from training and extensive testing are presented to assess the model's performance in controlled testing environments as a baseline, as well as in more realistic scenarios. Furthermore, the thesis includes a thorough literature review of the current state of automatic AAA and the manual methods currently used by forensic analysts.

Chapter 2

Background

In this chapter, the concepts and technologies of relevance to the research on audio forgery detection are presented and explained. This includes a comprehensive overview of audio signal processing techniques, the nature and detection of audio forgeries, and the application of deep learning methods in audio forensics. The aim is to provide a thorough understanding of the technical and theoretical background necessary for the development and evaluation of the Convolutional Neural Network (CNN) model applied in the thesis.

As part of the project planning report preceeding the master's thesis work [4], the state of the art and related work has been identified and studied. Some of the material from section 2 in the project planning report is therefore similar to some of the content in chapters 2 and 3.

2.1 Audio signal processing

In audio signal processing, it is common practice to transform audio data into a more interpretable form. Audio is inherently a time-based one-dimensional vector, but it can be visualised as an image that captures nuances of sound similar to how humans experience it. Visual representations like spectrograms allow machine learning models to leverage image classification techniques to analyse audio data. In this section, this process is described, and visualisations of the transformations are provided using the file `p1_g01_f1_4_x-a0001.wav` from NB Tale Part 1 [6].

Sound is digitised by sampling sound waves at fixed intervals; the sampling rate defines the number of samples captured per second [7]. In the physical world, sound is created by variations in air pressure within human-audible frequencies. Loudness, which we perceive, corresponds to the sound's amplitude, measured in decibels (dB). In digital audio, each sample captures a moment's amplitude, with bit depth defining its precision. Higher bit depth ensures a closer match to the original sound, enhancing the fidelity of digital audio representation. Digital audio amplitude is also measured in decibels (dB), mirroring the logarithmic nature of human hearing. This accounts for human's increased sensitivity to changes in quiet sounds over loud ones. For real-world sounds, 0 dB is the threshold of human hearing, with louder sounds registering higher dB values. Conversely, in digital audio, 0 dB signifies the peak loudness, with lower amplitudes expressed as negative values. A decrease of -6 dB approximately halves the amplitude, and sounds below -60 dB are usually imperceptible without significant volume increase [7].

Digital audio can be visualised as a waveform. The waveform visualisation represents sound by plotting amplitude changes over time, providing insight into the audio features such as event timing, signal loudness, and any audio irregularities or noise [7].

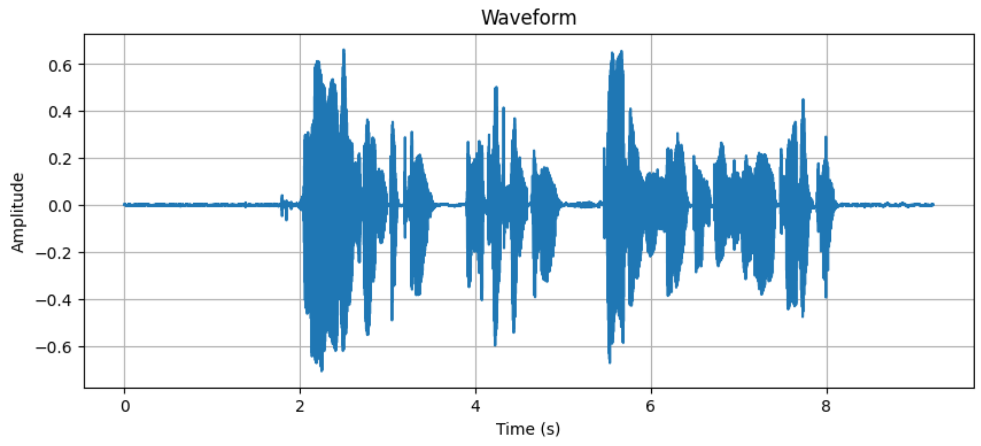

Figure 2.1: Time domain representation (waveform) of the example file

Plotting the frequency spectrum, which is another method to visualise audio data, involves using the Discrete Fourier Transform (DFT) to visualise the signal's frequencies and its intensities. This frequency domain representation provides the same information as the waveform (time domain representation), but focuses on the strength of various frequencies at a specific moment, contrasting the waveform's emphasis on amplitude changes over time [7].

To track changing frequencies in audio one can use the Short Time Fourier Transform (STFT) to create a spectrogram. This technique involves taking multiple DFTs over consecutive, short time frames and layering these spectra. A spectrogram visualises the frequency, time, and amplitude of an audio signal on a single graph, useful for, e.g., distinguishing vowel sounds in speech by their characteristic frequencies [7].

Mel spectrograms are graphical representations of sound that mimic how the human ear perceives different frequencies, using the Mel scale for this purpose [7]. The Mel scale, introduced by Stevens, Volkmann and Newman [8], is a perceptual scale of pitches judged by listeners to be equal in distance from one another. It mimics the human ear's response to different frequencies, which is more sensitive to variations in lower frequencies than to those in higher frequencies. This scale is logarithmic in nature, especially beyond 500 Hz, closely resembling the human ear's perception. It is widely used in audio processing, to ensure that processed

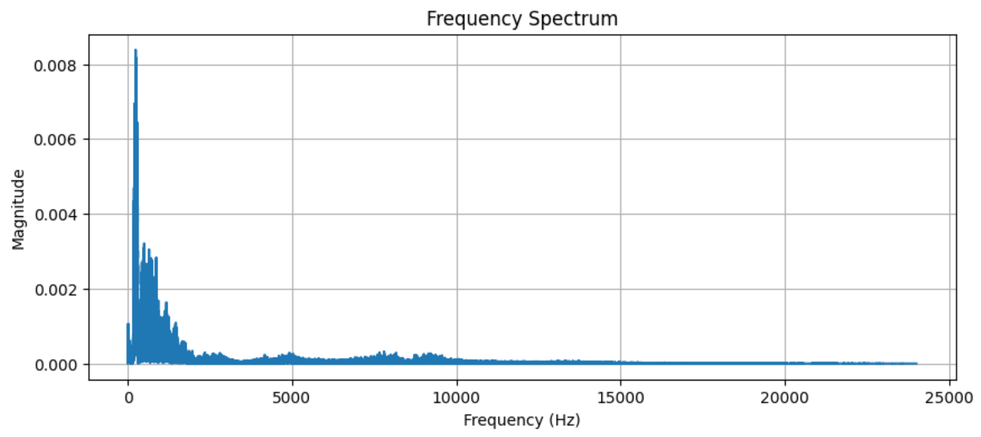

Figure 2.2: Frequency domain representation (spectrum) of the example file

audio aligns more closely with how humans actually hear sounds.

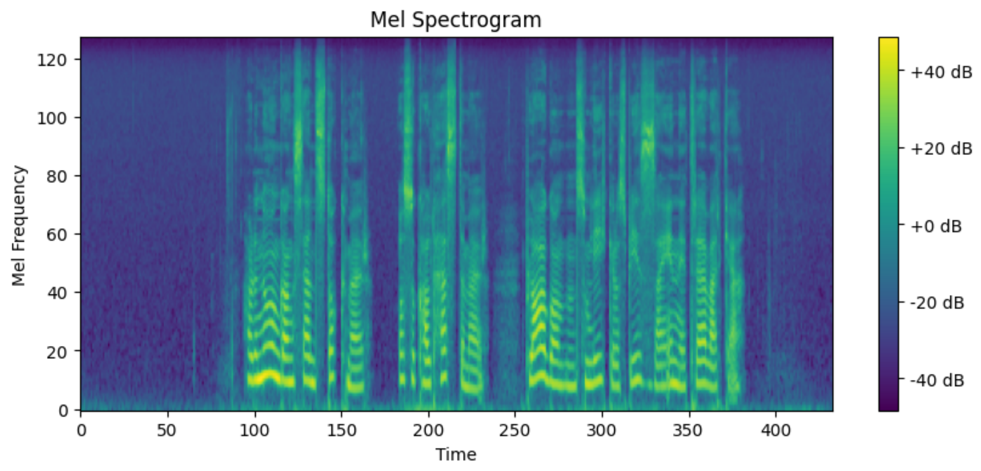

Figure 2.3: Mel spectrogram of the example file.

In a Mel spectrogram, the strength of sound at various frequencies is mapped over time, creating a visual, two-dimensional image of the sound. This method applies a Mel filterbank to frequency spectra derived from short audio segments via the STFT, effectively transforming linear frequency measurements to the Mel scale. This adjustment highlights the importance of lower frequencies and diminishes the relative significance of higher frequencies, which means that the Mel spectrogram outperforms standard spectrograms for tasks requiring nuanced sound analysis. As it captures features which aligns with human auditory perception, it is a great tool for diverse audio processing applications, e.g., speech recognition [7].

2.2 Audio forgeries and Audio Authenticity Analysis (AAA)

A file cannot be edited without generating some kind of traces. The examination of these traces, also referred to as artefacts, is essential to ensuring the authenticity of audio files. Disparities in background noise, speech flow, and sudden changes in acoustic properties are examples of common audio artefacts. Analysing an audio file for these artefacts to determine whether it is original or forged is known as Audio Authenticity Analysis (AAA) [3]. Audio forgeries can be classified according to which aspect of the file that has been modified: container-based forgeries, in which the file's metadata has been altered; and content-based forgeries, which involve edits to the actual audio content [3]. Within the category of content-based-forgeries, this project will primarily focus on copy-move forgery (CMF). Splicing and deletion forgeries are also presented, to give a more comprehensive view of the field.

Copy-move Forgeries which involve copying and pasting a segment of audio into another location of the same recording, as illustrated in Figure 2.4.

Splicing Forgeries in which two or more audio files are combined to create a single audio file, as illustrated in Figure 2.5.

Deletion Forgeries in which a segment of the audio has been removed.

These kinds of forgeries are the most common and, despite their simplicity, are very effective in changing the content or background of an audio file.To keep the project's scope within manageable bounds, other kinds of forgeries — deep fakes in particular — are excluded.

Adversaries who engage in audio forgery may also use various post-processing operations to obscure the traces of their tampering and make the forgeries more difficult to detect. These operations are designed to reduce the susceptibility of the tampered audio to detection techniques by altering the audio signal in different ways. Common post-processing techniques used in audio forgery include noise addition, filtering, compression, frequency scaling, pitch shifting, and resampling [3].

Along with manual inspection of audio files and searching for artefacts, Electric Network Frequency (ENF) signals can be analysed to detect audio forgeries by leveraging the unique signatures inherent in power grids [9] [10]. Smartphones and other recording devices are typically susceptible to picking up these ENF signals through their microphones when they record in environments where electrical devices are operating. This incidental capture of ENF signals in smartphone recordings embeds a time and location-specific marker into the audio. This frequency is typically stable (e.g. 50 Hz in many countries, 60 Hz in others) but can exhibit minor fluctuations due to various factors such as demand and supply imbalances. The presence of consistent ENF signals can validate the authenticity of the recording's timestamp and location, as the fluctuations in ENF at any given time are unique to specific geographical regions and time periods. The method

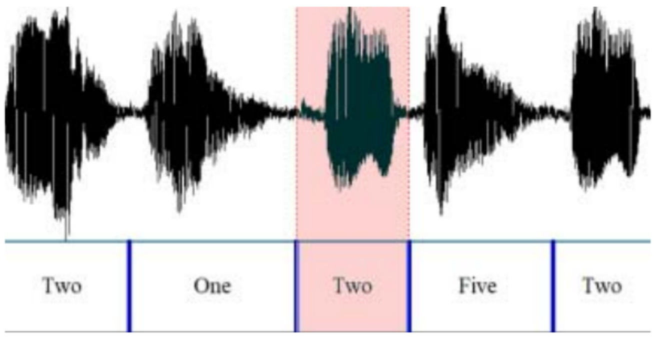

Figure 2.4: Copy-move forgery example. Reprinted with permission from *Audio Forgery Detection Techniques: Present and Past Review*, by Bevinamarad et. al [3] ©2020 IEEE.

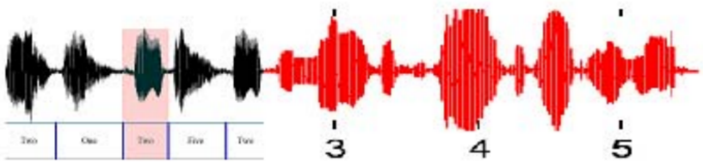

Figure 2.5: Splicing forgery example. Reprinted with permission from *Audio Forgery Detection Techniques: Present and Past Review*, by Bevinamarad et. al [3] ©2020 IEEE.

relies on extracting the ENF signal from audio files. The utility of ENF analysis in audio forensics stems from its ability to detect editing, splicing, and other modifications that may be imperceptible to the human ear. By analysing the continuity and consistency of ENF signals in audio files, forensic analysts can determine whether the recording has been tampered with. For example, sudden changes or discontinuities in ENF patterns may indicate splicing of segments from different recordings [9] [10]. Device trace analysis is another method in detecting audio splicing [11]. Splicing forgeries are, as mentioned, audio files where two or more recordings are combined to make a new audio file. These unique features can therefore be used for identifying changes in recording devices, which may be an indicator of splicing.

2.2.1 Best Practice Manual for Digital Audio Authenticity Analysis

A framework for evaluating the authenticity and integrity of digital audio recordings can be found in the European Network of Forensic Science Institutes' "Best Practice Manual for Digital Audio Authenticity Analysis" [2]. This subsection provides an overview of the framework, which was initially cited by the Oslo Police District. An overview of the techniques covered in the document is given in Table 2.1.

An initial examination of the contextual information in the audio recording is the first step in the process. This involves familiarising oneself with the recording environment, the hardware and software used, and any pre-processing, such as filters or speech enhancements, that was applied to the audio signal. The first stage is essential because it establishes the framework for the subsequent analysis and aids in the formulation of theories regarding the authenticity of the recording [2].

A thorough evaluation of the quantity and quality of the recording is completed after the contextual analysis. This stage is crucial for figuring out which forensic techniques is best suited, particularly in situations with difficult circumstances like low Signal-to-Noise Ratios (SNRs). The results of this evaluation are a list of suitable methods for additional investigation. Reference recordings are made whenever feasible, with equipment that is comparable to the ones used for the original recording. When recordings are made on removable media or when the original equipment is not available, this comparative analysis is especially helpful. Reference recordings offer a standard by which the recording in question can be assessed [2].

The retrieval, documentation, and analysis of all traces found within the audio recording are essential steps of the forensic process. This holistic method is crucial for confirming or disproving theories regarding the veracity of the recording. It entails closely examining the recording at different intervals to spot any irregularities or indications of manipulation [2].

Findings are presented in a way that ensures precision and clarity, particularly in legal contexts. A few strategies to make sure of this are outlined in the document, for instance, audio evidence needs to contain all pertinent evaluation and interpretation results, the style of presentation is customised to fit the demands of the legal process, written reports go through peer review, and the equipment used to playback audio in courtrooms needs to be able to do so in a clear and accurate way [2].

2.3 Deep learning-based audio forensics

In this section, the technical components of the automated deep learning-based AAA solution are presented and explained in detail.

2.3.1 Convolutional neural network (CNN)

Artificial Neural Networks (ANNs) are a subset of Machine Learning (ML) algorithms designed to mimic the learning process of the human brain. These systems are constructed as networks of neurons that communicate to collectively learn from the data they are trained on. During the learning phase, an ANN processes an input via an input layer, which then forwards the input to various hidden layers. Each hidden layer makes decisions about how to process the input based on information received from the preceding layer, optimising decisions that enhance the overall output. The learning can be either supervised, where the algorithm is trained with labelled datasets knowing the expected output for each sample, or unsupervised, where the algorithm attempts to identify patterns in unlabelled data. The choice between these methods depends on the available data and the specific objectives of the project, with supervised learning typically preferred for predictive modelling and unsupervised learning suited for data exploration. ANNs with several hidden layers are referred to as deep [12].

Convolutional Neural Networks (CNNs) are specifically designed for handling structured grid data, such as images. The architecture of CNNs includes layers organised in three dimensions: width, height, and depth. Unlike standard ANNs, the neurons in an ANN connect only to a small region of the preceding layer, optimising for spatial hierarchies in data processing. CNNs are particularly suited for this task because they can automatically learn hierarchical feature representations from raw data, making them effective for complex pattern recognition tasks. Figure 2.6 visualises how data is passed through the network from the input layer to the output layer via several key types of layers [12]:

1. **Convolutional Layers**: These layers apply a selection of learnable filters to the input. Each filter captures certain features by sliding across the input data and computing the dot product of the filter and local regions of the input, generating a feature map. This operation captures the local dependencies in the input data. The output of this computation passes through an activation function like the Rectified Linear Unit (ReLU) to introduce non-linearity.

Category	**Method**	**Description**
Continuity of time-variant traces	Auditory analysis	Focus on listening for inconsistencies in the signal, like unnatural fade-ins/outs or abrupt changes in sounds. Waveform and spectrogram analysis supplement findings.
	ENF analysis	Captures Electric Network Frequency signals in recordings, used to detect deletions or additions of content by analysing frequency and phase trajectories.
Invariability of time-invariant traces	Auditory and visual analysis	Looks for unnatural changes in reverberation, background sounds, encoding artefacts, and sound levels. Spectral characteristics and waveform analysis detect abrupt changes, such as in frequency or noise levels.
	DC-offset analysis	Examines variations in DC-offset to detect editing, like additions or deletions in the audio file.
	Frequency response analysis	Identifies additions of content recorded on a different device or comparisons with test recordings.
Invariability of periodic traces	Auditory and visual analysis	Focuses on sudden changes in traces from transmission devices or background sounds, like periodic dropouts or rate changes of a wall clock in the background.
	Codec frame analysis	Examines periodic traces from lossy encoding embedded in a file, using inverse decoding paradigm for analysis.
Detection of traces of post-processing	File structure and metadata analysis	Involves visualising file structure and searching metadata for signs of post-processing, like information from editing software.
	Double encoding traces	Detects when a lossy-encoded file is decoded and then re-encoded, identifying double encoding artefacts.
	Replicated time intervals	Identifies replicated portions in the audio using auditory and visual analysis, and signal subtraction techniques.
Comparison of recording traces with contextual information	Environment analysis	Compares acoustical properties of the recording to the known environment, but current methods are considered unreliable.
	Microphone analysis	Examines microphone frequency response in recordings to identify the likely used microphone or device.
	Encoding analysis	Checks for unexpected traces of lossy encoding in WAV files or verifies bitrate in compressed files.

Table 2.1: Categorisation and description of methods described in *Best Practice Manual for Digital Audio Authenticity Analysis* by Bartle, et.al. (2022), reprinted from the project planning report by Skogseth (2023)

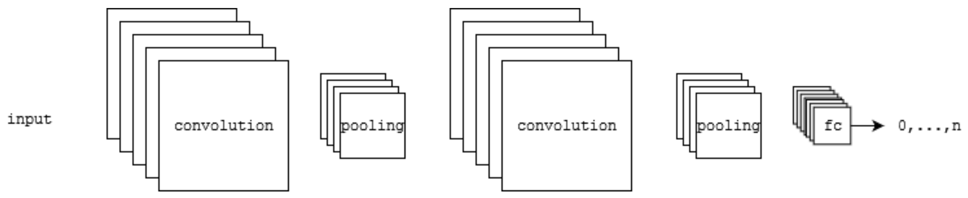

Figure 2.6: Example CNN architecture with two convolution layers, pooling layers and a fully connected (fc) layer. n represents the total number of classes.

2. **Pooling Layers**: Also known as subsampling or downsampling, this layer decreases the dimensionality of each feature map yet maintains the most relevant information. Pooling aids in identifying features despite changes in scale and orientation and also minimises the computational cost for the upcoming layers.
3. **Fully Connected Layers**: Neurons in a fully connected layer are connected to all activations in the preceding layer. These layers aggregate all the information learned by the previous layers across the image to recognise the broader patterns effectively. The final fully connected layer integrates the features to classify images into various categories based on the training dataset.

Rectified Linear Unit and He initialisation

The Rectified Linear Unit (ReLU) is a widely used activation function in deep learning due to its simplicity and effectiveness. However, proper initialisation of network weights is crucial for stable gradients and efficient training. He et al. [13] proposed a robust initialisation method specifically designed for ReLU and its variants, which has become a standard practice in training deep neural networks.

The ReLU activation function is defined as:

$$f(x) = \max(0, x)$$

While simple, ReLU has the advantageous property of not saturating in the positive domain, which helps in mitigating the vanishing gradient problem common in deep networks. However, inappropriate initialisation of weights can still lead to issues such as dead neurons or slow convergence.

He initialisation addresses this by scaling the variance of the initial weights based on the number of input units. Specifically, the weights are initialised from a zero-mean Gaussian distribution with variance:

$$\mathrm{Var}(w) = \frac{2}{n_{\mathrm{in}}}$$

where n_{in} is the number of input neurons to a given layer. This initialisation method ensures that the variance of the activations remains approximately constant across layers, preventing the gradients from exploding or vanishing.

The following equations illustrate the He initialisation:

$$w_{i,j} \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{\text{in}}}}\right) \tag{2.1}$$

This initialisation technique is derived from considerations of both forward and backward propagation. For forward propagation, it ensures that the variance of the output remains stable:

$$\text{Var}(y) = \frac{2}{n_{\text{in}}}\text{Var}(x) \tag{2.2}$$

For backward propagation, it prevents the gradient's variance from diminishing:

$$\text{Var}(\Delta x) = \frac{2}{n_{\text{out}}}\text{Var}(\Delta y) \tag{2.3}$$

In practice, He initialisation has been shown to significantly improve the convergence speed and performance of deep networks, particularly those employing ReLU or its variants. The results in He et al.'s study demonstrate that networks initialised with this method outperform those using traditional initialisation schemes, leading to state-of-the-art performance on various benchmarks [13].

This initialisation strategy has thus become an integral part of training modern deep neural networks, ensuring that even very deep architectures can be trained efficiently from scratch.

Batch Normalisation

Batch Normalisation (BN) is a technique introduced by Ioffe and Szegedy [14] to reduce internal covariate shift, which refers to the change in the distribution of layer inputs during training. This shift can slow down training because each layer must continuously adapt to the new distribution of inputs. BN addresses this by normalising the inputs of each layer so that they have a mean of zero and a variance of one. This normalisation is done on each mini-batch rather than the entire dataset, which ensures that the normalisation process is efficient and suitable for stochastic gradient descent. This technique allows for higher learning rates, reduces the need for careful initialisation, and acts as a regulariser, often reducing the need for dropout. BN ensures that the input to each layer maintains a stable distribution, which significantly accelerates the training of deep networks by allowing for faster convergence and more robust learning.

The following equations are involved in the BN algorithm and are from the article by Ioffe and Szegedy [14]. Equation (2.4) computes the mean of the mini-batch, and Equation (2.5) computes the variance. The normalisation step in Equation (2.6) ensures that the inputs to each layer have a mean of zero and a variance of one. Finally, the scale and shift step in Equation (2.7) adjusts the normalised inputs using learnable parameters γ and β, preserving the network's capacity to represent the data accurately.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.4}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{2.5}$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2.6}$$

$$y_i = \gamma \hat{x}_i + \beta \tag{2.7}$$

Cross-entropy loss

When training a CNN, the data is passed through the model to produce predictions. The model's predictions are then compared to the actual labels using a loss function to measure the error. The cross-entropy loss function is commonly used in classification tasks to measure the performance of a model where the output is represented as a probability between 0 and 1 [15].

The following functions and the related equations are from the |pytorch| documentation [16]. The model outputs raw, unnormalised scores known as logits, which are then converted into probabilities using the softmax function. The softmax function (2.8) calculates the probability distribution over C classes, where P_i is the probability of class i given the logit z_i. The cross-entropy loss for a single data point (2.9) measures the difference between the true label distribution y_i and the predicted probability distribution P_i. To provide a single scalar loss value for optimisation, the loss is often averaged over a batch of data points (2.10). This batch loss considers the true label $y_i^{(n)}$ and the predicted probability $P_i^{(n)}$ for class i of data point n in a batch of size N. These equations collectively help in evaluating how well the predicted probability distribution matches the true distribution of the labels, making the cross-entropy loss an effective tool for training classification models.

$$P_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{2.8}$$

$$\text{Loss} = -\sum_{i=1}^{C} y_i \log(P_i) \tag{2.9}$$

$$\text{Loss}_{\text{batch}} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{C} y_i^{(n)} \log(P_i^{(n)}) \tag{2.10}$$

Adam optimiser

The Adam optimiser, defined by Kingma and Ba [17], is known for its adaptive learning rates, which are computed for individual parameters based on estimates of first and second moments of the gradients. The first moment estimate captures the average of the gradients, while the second moment estimate captures the uncentered variance, or the squared gradients. Adam includes bias-correction steps to counteract the bias introduced in these moment estimates, which are initialised at zero. This ensures that the estimates are more accurate, especially in the initial stages of training. Overall, Adam is computationally cheap, has low memory requirements, and is a good fit for problems with a lot of data and parameters. Its ability to adapt learning rates and handle noisy and sparse gradients makes it a robust and versatile choice for optimisation in machine learning tasks.

The order of computation for Adam and their related equations are from the article by Kingma and Ba [17]. Before beginning, the first moment vector m_0, the second moment vector v_0, and the time step t are initialised at 0. First (2.11), compute the gradient g_t of the objective function with respect to the parameters at time step t. Then (2.12), using the exponential decay rate β_1, it updates the first moment estimate before (2.13) computing the second raw moment estimate using the exponential decay rate β_2. The next step (2.14) is to compute the bias-corrected first moment estimate, and then (2.15) compute the bias-corrected second raw moment estimate. Finally, the parameters are updated using the bias-corrected moment estimates and a learning rate α 2.16. Here, ϵ is a small constant to prevent division by zero.

$$g_t = \nabla_\theta f_t(\theta_{t-1}) \tag{2.11}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{2.12}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{2.13}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{2.14}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2.15}$$

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.16}$$

Applciation in audio classification

Although traditionally used for image data, CNNs can also be effectively applied to audio classification tasks. By transforming audio signals into Mel spectrograms, which represent how humans perceive sound, audio data can be structured into a format analogous to image data. Mel spectrograms display the spectrum of frequencies of audio signals over time, presenting an opportunity for CNNs to process and analyse audio similarly to how they process images [5] [18].

By training a CNN on a large dataset of labelled Mel spectrograms, the network learns to recognise and differentiate between categories based on unique pattern variations within the spectrograms. This technique can be used as a tool for automated audio analysis for classifying diverse types of audio information - for example, forged and original audio files [5] [18].

Machine learning metrics

The performance evaluation of machine learning models is commonly done by calculating precision, recall, and the F1 score [19].

- Precision indicates the accuracy of the model in predicting true positives. This score is computed by dividing the number of true positives by the sum of true positives and false positives. A high precision score signifies that the model's predictions are highly relevant.
- Recall measures the model's sensitivity, showing how effectively it identifies true positives out of all actual positive cases.
- The F1 score, which is the harmonic mean of precision and recall, provides a balanced measure of the model's performance, especially useful when dealing with imbalanced datasets.

In the article by Powers [19], they argue that there are some key limitations to these metrics. Precision, recall, and F1 score focus solely on the positive cases and predictions. They do not account for the model's performance on negative examples, which can lead to an incomplete assessment of the model's overall effectiveness. So although a model might have high precision and recall, it could still be performing poorly on negative cases, which leads to a biased understanding of the model's performance. Precision, recall, and F1 score are also sensitive to the underlying class distribution and the model's bias towards certain predictions. This sensitivity can cause misleading evaluations, especially in datasets with imbalanced class distributions.

Despite their inherent biases identified by Powers [19], precision, recall, and F1 score remain the standard for evaluating binary classification problems due to their straightforward interpretation and ease of calculation.

2.3.2 Overfitting and underfitting

The concepts of overfitting and underfitting are important in understanding ML performance and generalisation. Overfitting occurs when a model learns not only the patterns in the training data but also noise and outliers. This results in a model that performs well on training data but fails to generalise to unseen test data. An overfitted model essentially memorises the training data instead of learning their discernible features [20]. Underfitting happens when a model is too simple to capture the patterns in the data. This results in poor performance on both the training data as well as new data [20].

An overfitted model might recognise specific noise patterns in the training data, mistaking them for meaningful features, while an underfitted model might fail to capture essential patterns altogether.

Chapter 3

Related work

In this chapter, the state-of-the-art methods for Audio Authenticity Analysis (AAA) and dataset generation techniques are presented. The goal of the chapter is to provide a comprehensive overview of the current methodologies and technologies used in the field of audio forgery detection. The effectiveness of these methods is presented based on precision, accuracy, and other relevant metrics, as reported in the literature. This chapter also explores various strategies for creating datasets of copy-move forgeries.

3.1 Automatic audio authenticity analysis

3.1.1 Traditional automated methods

There is potential for automated techniques for AAA to be sigificantly more efficient than current approaches. The state-of-the-art methods from the literature are covered in this subsection. Table 3.1 provides an overview of the related work, the forgery type they focus on, the methods they used, as well as the performance they were able to achieve.

In order to identify audio forgeries, Hua et al. [9] present a novel method of matching the Absolute Error Map (AEM) of the ENF to the ENF database. Then, this method is applied to two algorithms that not only identify deletion and splicing forgeries but also validate the audio file timestamps. Another approach related to ENF is presented by Hsu et al. [10]. Their approach involves creating a Chinese ENF database with diverse ENF signals extracted from audio recordings. The methodology is comprised of three main steps: extracting the ENF signal from audio files, refining the signal through wavelet decomposition, and analysing it using an autoregressive model. The latter quantifies the influence of previous values in a time series on its current value, helping to predict future points based on its own past observations. The autoregressive coefficients are then utilised to train three types of machine learning models—Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Deep Neural Networks (DNN)—to distinguish between authentic and tampered audio files. The experimental evaluation, con-

Ref.	Year	Forgery-type	Method	Result (%)
[9]	2016	Insertion, splicing, deletion	Absolute-Error-Map of ENF signals	-
[21]	2017	Copy-move	Voice activity detection and Local Binary Point	A = 96.59
[22]	2019	Copy-move	Dynamic time warping	P = 95.05
[23]	2021	Synthetic speech	Light CNN and transformer	A = 98.83
[18]	2023	Copy-move	Mel spectrogram feature matching	A = 83.17
[10]	2023	Copy-move, deletion	ENF analysis, K-NN, SVM, DNN	A = 92.16
[11]	2023	Splicing	CNN based on acquisition device traces	A = 95
[5]	2023	Copy-move	Mel spectrogram-based CNN	A = 95.8
[24]	2023	Copy-move	Super-resolution spectrogram and keypoint-based clustering	P = 99
[25]	2023	Synthetic speech	Spectrogram, multi-task neural network	AUC = 0.90
[1]	2024	Copy-move	MFCC, ΔMFCC and LPC	A = 76.48

Table 3.1: Table of reviewed methods. P = precision, A = accuracy, AUC = Area Under Curve. The results are averaged where multiple tests were done

ducted on a specially recorded Chinese speech database within an anechoic chamber, demonstrates the effectiveness of this method in identifying audio tampering, with accuracy rates ranging from 91% to 93%.

Imran et al. [21] identify matches indicating copy-move forgery using Local Binary Point (LBP) histograms after using Voice-Activity Detection (VAD) to identify the words and their corresponding boundaries.

Yan, Juang and Huang [22] detect copy-move forgeries by leveraging pitch and formant features within voiced speech segments. The method involves segmenting the speech into voiced and unvoiced parts, extracting the pitch and the first two formant sequences from the voiced segments, and then using dynamic time warping (DTW) to measure the similarities between these features. By setting a threshold for the DTW values, they can identify and locate copied segments that have been moved to different positions within the same recording. This approach is shown to be highly effective even in the presence of common post-processing operations such as noise addition, filtering, and compression, achieving high accuracy in various test scenarios. The extensive experiments validate the robustness of the method, making it a valuable tool for practical forensic applications.

3.1.2 Machine learning methods

The field of automatic audio forgery detection has seen massive transformation thanks to the application of machine learning, especially deep learning. The effectiveness of merging CNNs and vision transformers to improve detection capabilities is shown by Liu et al. [23]. This integration increases the accuracy of forgery detection and enables better handling of complex audio data.

Ustubiologu, Tahaoglu and Ulutas [18] introduce a novel feature matching approach using Mel spectrograms to detect and locate audio copy-move forgery. This method uses keypoints extracted from the Mel spectrogram representation of the audio, beginning with converting the input audio into a Mel spectrogram image and using Scale-Invariant Feature Transform (SIFT) keypoints for matching. These keypoints are matched to detect similar regions, and the matched keypoints are used to identify potential forgery regions. The proposed method includes a post-processing step that eliminates false positives and accurately marks forged areas in the spectrogram image. The forged segments in the audio file are identified using the positions of the forged regions in the spectrogram image.

A splicing detection technique based on the analysis of acquisition device traces is presented by Leonzio [11]. In order to apply this technique, a CNN is used to extract features, a K-means clustering algorithm is used to find traces from various models, and a custom distance measure is used to locate the forgery.

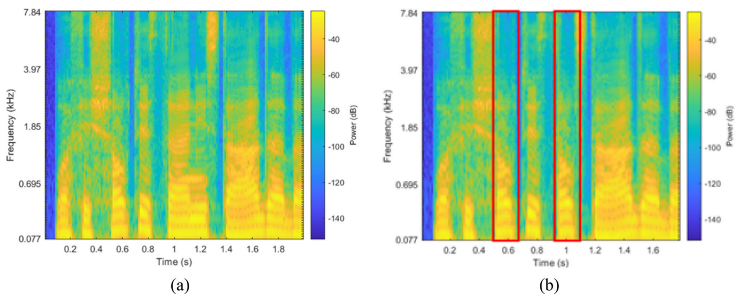

Figure 3.1: Mel spectrogram of (a) an original audio file and (b) its related copy-move forgery. Reprinted with permission from *Mel spectrogram-based audio forgery detection using CNN* by Ustubioglu, Ustubioglu, and Ulutas [5] ©2022 Springer

Mel spectrograms are used as features in a CNN in an approach proposed by Ustubioglu, Ustubioglu and Ulutas [5]. Mel spectrograms are used to train the CNN, which subsequently classifies Mel spectrograms as either original or forged. Their proposed method achieves the best performance in a comparative analysis included in the paper.

Ustubioglu et al. [24] provide an example of how machine learning can be used to locate and identify audio forgeries using keypoint-based clustering and high-resolution spectrogram images. The study highlights how machine learning can be used to detect complex and subtle artefacts in audio files that would be difficult to detect using more conventional techniques.

Akdeniz and Becerikli [1] introduce a novel method for detecting audio copy-move forgery using an ANN, leveraging Mel Frequency Cepstral Coefficient (MFCC), Δ MFCC, and Linear Prediction Coefficient (LPC) as features from the audio signal. The results demonstrate the method's effectiveness in identifying forgeries, with significant improvements in detection accuracy compared to traditional methods. This approach offers a new direction in the field of digital audio forensics by employing advanced feature extraction techniques and machine learning algorithms. Although the precision achieved is less than from other methods in the related work, they argue that this method could potentially be more robust to changes in a number of variables, such as the environment of the recording, noise level, and the speaker.

The classification model developed in this thesis is a Mel spectrogram-based CNN inspired by the method of Ustubioglu, Ustubioglu and Ulutas [5] and based on the guide by Doshi [26]. The CNN architecture consists of five convolutional blocks, each incorporating convolutional layers, ReLU activations, and batch normalisation. An adaptive average pooling layer follows these blocks, which prepares the features for the linear classification layer.

In addition, this thesis explores a method to classify longer audio files with the CNN. The model processes these files by segmenting their spectrograms into smaller, overlapping chunks. The final classification for the entire audio file is determined through a majority voting scheme over the smoothed segment predictions.

3.2 Database generation

Imran et al. [21] created a database for detecting fraudulent media files using recordings from various environments and types of microphones. The database includes recordings from a soundproof room, an office, and a cafeteria, each with different levels of background noise. The forged audio database was generated using a VAD module to accurately detect the boundaries of spoken words in the audio, simulating realistic forgeries. The locations for copy and move actions within the recordings were determined using Gingerbreadman chaos theory, so that the forgery could be at any point within the recording. Another point of doing this according to chaos theory, as opposed to doing it with a random number generator, is its deterministic nature. This makes it possible to replicate the database generation process if the initial parameters are known. The database consists of a total of 1350 forged audio recordings. These recordings were generated from 270 original audio recordings in Arabic, taken from 90 speakers from various countries and regions, in three different environments.

Ustubioglu, Ustubioglu and Ulutas [5] used audio samples from the Arabic Speech Corpus and TIMIT speech database to create forged audio files by segmenting the speech into voiced and unvoiced parts, then copying and pasting random segments. The resulting duration of each repeated segment formed is ap-

Paper	**Number of samples**	**Forgery**	**Database generation method**
[21]	1350 forgeries	Copy-move	VAD and Gingerbreadman chaos theory
[27][5]	368 + 1329 forgeries	Copy-move	Pitch based separation of segment, then replacing one voiced section with another
[24]	2208 + 160 forgeries	Copy-move	Randomly generated by separating voiced and unvoiced parts of the audio based on VAD, then replacing one voiced section with another
[1]	2189 forgeries	Copy-move	Automatically generated by copying the third second of the audio and inserting it after the first second

Table 3.2: An overview of copy-move forgery datasets and the methods used to generate them in the related work. The "+" represents that two different datasets was used as the source of audio files

proximately between 0.2 and 0.6 seconds. A total of 368 forged audio files were created from the TIMIT database, and 1329 forged audio files were generated from the Arabic Speech Corpus, amounting to a total of 1697 samples. This is the only dataset which is publicly available through a link to a Google drive folder in the paper.

Ustubioglu et al. [24] used a similar approach: the database consists of audio files created using the TIMIT speech database and the Arabic Speech Corpus. In the TIMIT-based dataset, each audio file is divided into segments, and random segments are copied and pasted in different places within the same audio file, with each forged segment ranging from 0.2 to 0.6 seconds. For the Arabic Speech Corpus dataset, 30 audio files underwent similar forgery and various post-processing attacks, such as adding white Gaussian noise, median filtering, and MP3 compression at different bitrates. The database, therefore, contains audio samples with plain copy-move forgeries and those subjected to several types of post-processing attacks, leading to a comprehensive set for testing forgery detection methods. The paper describes a dataset of 2208 forged audio files created from TIMIT, plus another set of 180 from the Arabic Speech Corpus, along with various versions of these files that were subjected to post-processing attacks to evaluate the robustness of their detection method.

Akdeniz and Becerikli [1] utilised a total of 4378 audio files, split evenly between original recordings and forgeries. The forged samples were generated by copying the segment from the third second of the audio and pasting it after the first second within the same recording.

For the purposes of this thesis, a novel dataset is generated using the NB Tale dataset from the National Library of Norway. NB tale encompasses manuscript read and spontaneous speech recordings. The forgeries are generated by extracting voiced segments using the YAAPT algorithm and randomly selecting a voiced segment to copy and paste at another random location in the audio file. The dataset includes 5361 original and 5361 forged samples of short audio recordings, along with 365 original and 365 forged samples of long audio recordings.

Chapter 4

Method

This chapter outlines the methodologies employed in this research to detect audio forgeries. It provides detailed descriptions of the literature review, as well as the processes involved in dataset creation, model development, training, and evaluation.

4.1 Literature review

A literature review focusing on audio forensics and audio authenticity analysis was conducted to obtain an overview of the field, including fundamental concepts and state-of-the-art techniques. Emphasising articles published in academic journals and books, the review aimed to identify the latest trends and methodologies in audio forgery detection. This initial review was conducted as part of the project planning report [4], and parts of the method are therefore also described there.

Google Scholar was the primary search engine used for sourcing relevant projects and theories due to its extensive database and ease of use. Keywords such as "audio forensics", "audio authenticity analysis", "audio forgery detection" and "machine learning" were used to find relevant literature. Google Scholar makes it easy to find relevant material from a range of different journals, conferences, and books.

The papers included provide a comprehensive view of the state-of-the art audio authenticity analysis techniques. The emphasis was placed on recent publications to understand current trends, but older works were also included to ensure a well-rounded perspective. The source of the literature spanned a variety of databases.

This is partly a development project of exploratory nature. The relevant literature offers too few details to properly replicate the steps to build the methods used in the project. Because of this, more searches for specific technologies, methods and subjects was done iteratively to find articles, guides and other webpages which explain the components needed for creating both the dataset and CNN model.

The literature review highlights the current state-of-the-art in audio forensics and authenticity analysis, emphasising the importance of robust detection techniques in identifying forgeries. Key findings from the review include various methodologies for creating and detecting audio forgeries, particularly copy-move forgeries, and the application of machine learning models in this domain. These insights form the foundation for the methods developed in this thesis, guiding the creation of a novel Norwegian dataset and the design of a CNN model to effectively detect audio forgeries. This chapter will build upon these findings to detail the specific techniques and processes employed in this research.

4.2 Dataset

To assess a model's ability to detect copy-move forgeries, it needs to be trained, evaluated and tested on a dataset of audio files where one section has been replaced by another. The only publicly available dataset that fit this description was one based on the Arabic Speech Corpus, found through the paper by Ustubioglu, Ustubioglu and Guzin Ulutas [5]. Manually creating the realistic forged audio files is a non-trivial and resource-heavy task; it would take a long time to create enough samples to effectively train a ML-model. The way this is solved in the related literature is through different ways of generating the forged audio files with an algorithm. This section presents the steps that were made to create the novel Norwegian dataset of copy-move forged and original audio files. The final dataset is available in Kaggle [28].

4.2.1 Source datasets

Modules 1 and 2 of the NB Tale dataset [6], a publicly available database for automatic speech recognition (ASR) by the National Library of Norway, were used to create the datasets for training, validation, and testing. Module 1 consists of 4,800 audio files with durations ranging from 4-14 seconds, recorded by 240 speakers reading 2,163 unique texts. Module 2 consists of 2,800 audio files with durations ranging from 4-24 seconds, recorded by 140 speakers reading 1,263 unique texts. Module 3, used for creating the test dataset with longer files, consists of spontaneous speech recordings from 380 speakers, with audio files up to several minutes long. There are two available versions of the three modules, which differs in the recording equipment used. For this project we chose the version recorded with a Sennheiser HS 2-5-1 headset microphone at a sampling rate of 48kHz, and a bitrate of 256kbps. The audio files were recorded at three different locations:

- Sound studio at NTNU.
- The Arctic University of Norway (UiT).
- University of Agder (UiA).

4.2.2 Dataset generation

The audio files from the source dataset were processed to create two distinct categories: copy-move forgeries and original audio files.

- **Copy-Move Forgery** A segment of the audio is duplicated and inserted back into the audio at another location. This simulates a scenario where a part of the audio file is copied and pasted elsewhere in the same file to, e.g., alter the meaning of a sentence.
- **Original** The original audio files are saved along with the forgeries as they are.

Two methods were explored for generating the forged audio files: a VAD-based method and a static copying method. The VAD-based method, which uses the YAAPT algorithm to extract voiced segments and create forgeries, was ultimately selected for the final dataset.

VAD-based copy-move

This section outlines the process used to generate a dataset of forged audio files using the YAAPT (Yet Another Algorithm for Pitch Tracking) algorithm, proposed by Kasi and Zahorian [29]. The audio files are first processed to extract voiced segments using the YAAPT algorithm. The voiced segments are then used to create forgeries by copying and inserting them into different locations within the same audio file. The following is a high-level overview of how the algorithm works [1] [5] [5] [30]:

The audio signal undergoes pre-processing to enhance the pitch information. This includes generating multiple versions of the signal using nonlinear processing techniques.

YAAPT predicts an approximate pitch track by analysing the spectral harmonic content of the signal. This involves calculating the Spectral Harmonic Correlation (SHC), which measures the harmonic structure of the spectrum.

Pitch candidates are generated from both the original and non-linearly processed signals using the Normalized Cross-Correlation Function (NCCF). This function measures the similarity of the signal with itself at various lag values to estimate the pitch period. The algorithm differentiates between voiced and unvoiced frames using the Normalized Low Frequency Energy Ratio (NLFER). This helps in identifying frames where pitch is present.

The final pitch track is determined using dynamic programming, to integrate information from both spectral and time-domain analysis.

The process of creating a forgery starts by loading a file and resampling it to a sampling rate of 16kHz. Then, the YAAPT pitch tracking algorithm is applied to the resampled audio, to extract the pitch. These pitch values are then used to identify voiced segments. The voiced segments are identified as regions where the pitch values are greater than zero. Voiced segments are only extracted if they are of a minimum of 0.4 seconds. If no voiced segments over the minimum length is found the file is not included in the dataset, which results in 5361 audio files included from modules 1 and 2 and 365 samples included from module 3.

The voiced segments are used as the basis for selecting a segment to copy in the files. The forgeries are generated by copying a segment of voiced audio and inserting it into another location within the same audio file. The voiced V segments are labelled as $V_1 \dots V_i$ where i represents the total number of voiced segments in the audio file. A unique random number a is chosen within the range $[1, 2 \dots i]$. The rule for selecting this number can be mathematically summarised as follows:

$$Let\, a \in \{1, 2, \dots, i\}.$$

The voiced segment V_a is then copied and pasted to a new random location L within the audio file, ensuring that the new location does not overlap with the original segment. The new location L is determined by:

$$Let\, L \in \{0, 1, \dots, T - (end_a - start_a)\}\, such\, that\, L \notin [start_a, end_a]$$

where T is the total length of the audio file, and $start_a$ and end_a are the start and end points of the segment V_a. This approach retains the original length of the audio file ensuring that there are no common features for forged vs original files, other than the forgery itself.

Static copy-move

To create the static training dataset, the audio files from NB tale module 1 were first filtered on duration. This was done by loading the source dataset using the `librosa` library, which provides functions for loading and analysing audio data. The duration of each file was checked using `librosa.get_duration()`. Files inside the 7 to 9 second range was saved for further processing. The filtering resulted in a total of 2913 files being included. To create the test dataset, the audio files from NB tale module 2 were also first filtered on duration, in the same manner as the training dataset. The filtering resulted in a total of 1193 files being included. During this step, all audio files were downsampled to a sampling rate of 16kHz to reduce the files to a more manageable size to reduce computational complexity during processing.

The static copy-move forgery method, based on the method used by Akdeniz and Becerikli [1], involves copying a specific segment of the audio file and inserting it at a predefined location. This method was used for initial testing, while the final dataset employs the VAD-based method for creating forgeries.

The method was adjusted for the characteristics of the filtered datasets. For example, one adjustment that was made was that it was not fitting to paste the section after the first second as many of the files in the dataset have 1-2 seconds of leading silence. This would result in that the original and forged files could be classified based on the length of the leading silence, instead of the identical patterns.

For each audio file in the training dataset, the sixth second was copied and inserted after the second second. Let:

- $x[n]$ be the original audio file, where n is the sample index.
- f_s be the sampling rate (in samples per second).

For an audio file with a sampling rate of f_s, the relevant segments are defined as:

- segment1 $= x[0:2f_s]$ (samples from 0 to $2f_s - 1$)
- segment2 $= x[5f_s:6f_s]$ (samples from $5f_s$ to $6f_s - 1$)
- remaining $= x[2f_s:5f_s] \cup x[6f_s:]$ (samples from $2f_s$ to $5f_s - 1$ and from $6f_s$ to the end)

The forged audio file $y[n]$ is created by concatenating these segments:

$$y = [\text{segment1}, \text{segment2}, \text{remaining}]$$

where:

$$\begin{aligned}\text{segment1} &= x[0:2f_s]\\ \text{segment2} &= x[5f_s:6f_s]\\ \text{remaining} &= x[2f_s:5f_s] \cup x[6f_s:]\end{aligned}$$

4.3 Experimental setup

The experiments were conducted on a Windows 11 computer equipped with an Intel Core i7-10700 CPU, featuring a base clock speed of 2.90 GHz and a maximum boost speed of 4.80 GHz. The system has 32 GB of DDR4 RAM and an NVIDIA GeForce RTX 2070 Super GPU with 8 GB of dedicated VRAM. The development of data processing scripts and model programming were done in Visual Studio Code using Python 3.12.0. The following packages were used: os, librosa, pathlib, pandas, torch, torchaudio, torchvision, torchviz, matplotlib, shutil, numpy, soundfile, and collections.

4.3.1 Pre-processing and loading data

The data pre-processing steps were adapted from methods detailed in Doshi's guide [26]. For pre-processing the python library "torchaudio" was used in a three step process. These steps are represented as methods within the `AudioUtil` class in the script.

1. **Loading Audio Files**: The `open` method loads an audio file and returns the audio signal (sig) and its sample rate (sr). torchaudio.load directly fetches these two components, which are used for further processing. The sample rate indicates how many samples of the audio are taken per second.
2. **Creating Mel Spectrogram**: The `spectro_gram` method converts the audio signal into a Mel Spectrogram. This transformation is performed by applying a Mel scale filter to a Short Time Fourier Transform (STFT) of the

signal, configured by parameters such as the number of Mel frequency bins (`n_mels`), the window size (`n_fft`), and the hop length (`hop_len`). The variables used to create the spectrograms were `n_mels = 128`, `n_fft = 2048`, and `hop_len = 512`. The output is then converted from amplitude to decibels, which normalises the loudness and improves the visibility of various features in the spectrogram, making it more suitable for analysis and model training.

The Mel spectrograms are randomly divided into three subsets, also referred to as datasplits: 70% for training, 20% for validation, and 10% for testing. These subsets are then exported as ".pt" files for use during the training and validation processes. The training data is used to train the model, while the held-out validation data provides an indication of the model's performance on unseen data during the training phase. However, because the validation data is utilised during training, there is a potential for bias towards these samples. To mitigate this, a separate test split is reserved to objectively assess the model's performance after training, ensuring that the evaluation reflects the model's ability to generalise to completely new and unseen data.

4.4 Convolutional Neural Network model

This subsection details the CNN model architecture as well as its related training and evaluation process.The model architecture was inspired by Doshi's guide [26] and adapted for this study.

Model architecture

The CNN audio classification model is defined as a subclass of `nn.Module` within the PyTorch library. The model consists of five sequential convolutional blocks followed by a linear classification layer, optimised for an audio signal processing task.

The primary feature extraction in the model is accomplished through five convolutional blocks, each comprising a convolutional layer, a ReLU activation function, and a batch normalisation. He initialisation in each convolutional block is used with a non-linearity coefficient of 0.1 to ensure effective weight initialisation of ReLU activations. Each block also uses batch normalisation and a dropout rate of 0.3 to prevent overfitting.

- **Block 1**: 8 filters, size = 5x5, stride = 2x2, padding = 2x2.
- **Block 2**: 16 filters, size = 3x3, stride = 2x2, padding = 1x1.
- **Block 3**: 32 filters, size = 3x3, stride = 2x2, padding = 1x1.
- **Block 4**: 64 filters, size = 3x3, stride = 2x2, padding = 1x1.
- **Block 5**: 128 filters, size = 3x3, stride = 2x2, padding = 1x1.

The output of the final convolutional block is passed through an adaptive average pooling layer (`nn.AdaptiveAvgPool2d`) that transforms the output of the convolutional blocks to a fixed size (1x1), ensuring a consistent input size for the subsequent linear layer regardless of the input dimensions.

The pooled output is then flattened and passed to the fully connected linear layer (`nn.Linear`) which maps the extracted features to the final classification outputs. The linear layer consists of 128 input features, corresponding to the number of output channels from the last convolutional block, and 2 output features, which represent the class identifiers 0 and 1.

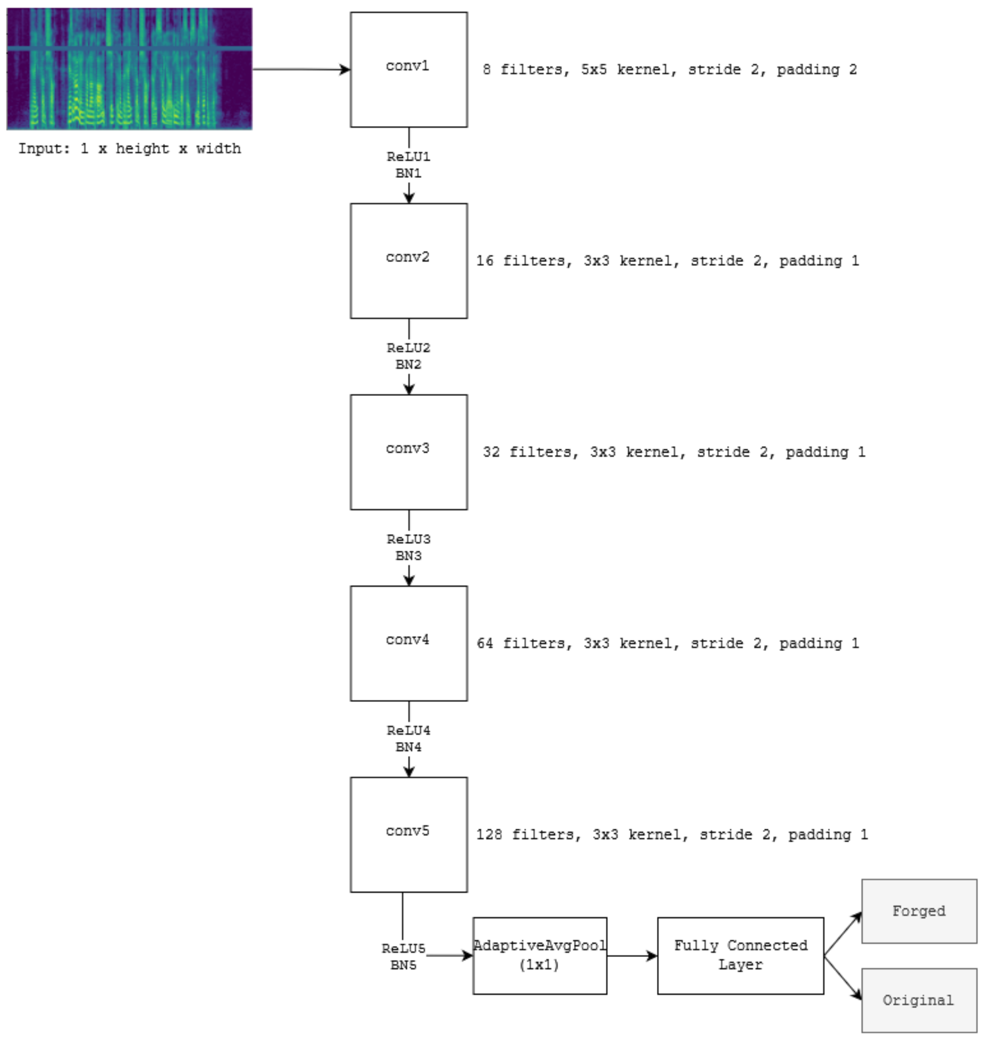
Input: 1 x height x width
conv1
8 filters, 5x5 kernel, stride 2, padding 2
ReLU1
BN1
conv2
16 filters, 3x3 kernel, stride 2, padding 1
ReLU2
BN2
conv3
32 filters, 3x3 kernel, stride 2, padding 1
ReLU3
BN3
conv4
64 filters, 3x3 kernel, stride 2, padding 1
ReLU4
BN4
conv5
128 filters, 3x3 kernel, stride 2, padding 1
ReLU5
BN5
AdaptiveAvgPool (1x1)
Fully Connected Layer
Forged
Original

Figure 4.1: CNN architecture

Training and evaluation

The process of training the CNN models involved several steps. The datasets were divided into training, validation, and test sets. The training set was used to train the model, while the validation set provided feedback on the model's performance during training. This feedback was used for tuning hyperparameters and preventing overfitting. The test set was reserved for final evaluation to assess the model's generalisation ability.

The models were trained using the Adam optimizer with a cross-entropy loss function. Various parameters were adjusted, including the learning rate, batch size, and the introduction of dropout and weight decay, to optimise performance. Training was conducted over multiple epochs, utilising checkpoints to save the best model state as it improves based on the validation loss. The choices made in regards to the parameters of the model during training were informed by preliminary experiments and literature reviews, aiming to achieve a balance between training efficiency and model robustness. The steps in the training loop are:

- **Normalisation and optimisation** At the start of each epoch, the input data is normalised based on the global mean and standard deviation of the training data. This step ensures that the input features have similar scales and contribute equally to the learning process. The model then performs a forward pass, a backward pass for error backpropagation, and an optimisation step where the model parameters are updated using the Adam optimiser.
- **Loss computation and backpropagation** During the training phase, the loss for each batch is determined using the cross-entropy loss function, which assesses the difference between the predicted probability and the true class label. This loss is used to guide the learning process through backpropagation.
- **Learning rate adjustment** After each batch, the learning rate scheduler updates the learning rate, facilitating the model's adaptation to the dataset's features more effectively, as training progresses.

Parallel to the training process, the model is periodically evaluated on the validation data after each training epoch:

- **Validation Phase**: In the validation phase, the model is set to evaluation mode, which disables certain operations that behave differently during training versus testing. The validation input data is also normalised by using the same mean and standard deviation from the training data. The model processes the entire validation dataset without computing gradients (to save memory and computation time), calculates the loss using cross-entropy loss, and determines the accuracy.
- **Monitoring Validation Loss**: The validation loss is used to evaluate the model's performance on unseen data. This gives an indication of how well the model is likely to perform on general data outside of the training set.
- **Early Stopping Implementation**: If the validation loss does not improve (decrease) for 10 consecutive epochs, there is an option for the training process to be stopped early. This mechanism prevents overfitting by stopping the training when the model no longer shows improvement on the validation data.

- **Saving checkpoints**: During the training, if a new minimum validation loss is observed, the model's state is saved as a set of weights. This ensures that even if the model's performance degrades in subsequent epochs due to overfitting or other issues, the best-performing state of the model is saved and can be used for future predictions.

Training was done with and without data augmentation of the training data. With data augmentation, multiple transformed versions of the spectrograms are created on-the-fly. This means that each time a spectrogram is fetched during training, a different version might be generated depending on the random transformations applied. This helps increase the variability of the training data without actually increasing the size of the dataset stored on disk. The augmentations which were applied were the same as utilised in the paper by Ustubioglu, Ustubioglu and Ulutas [5], adapted to the RandomAffine function from the torchvision.v2 library with the following parameters:

- degrees = 0: No rotation of the spectrogram
- translate = (0.1, 0.1): Translates the image by up to 10% of the width and height, effectively shifting the height and or width of the image.
- shear = 0.3: applies shear transformation with a shear range of 0.3. Shearing involves shifting parts of the spectrogram along one axis.
- fill = 0: specifies the fill values as 0 for areas outside of the boundaries of the input after the translate and/or shear transform is applied.
- interpolation = InterpolationMode.NEAREST: Uses nearest-neighbor interpolation to match the fill mode used in [5].

4.5 Tests with post-processed samples

To assess the robustness of the model under various adversarial conditions, multiple test datasets were created with different levels of noise and compression applied to the audio files. The augmentations applied include the addition of SNR adjusted noise and audio compression at different bitrates. By comparing the model's performance on these processed test datasets to its performance on unprocessed test datasets, we can evaluate how well the model generalises to real-world scenarios where audio quality may be compromised, degraded or manipulated in some form. The augmentations applied include the addition of Signal-to-Noise Ratio (SNR) adjusted noise and audio compression at different bitrates.

Signal-to-Noise Ratio (SNR) Noise

To simulate environmental noise and signal degradation, SNR adjusted noise was added to the audio files at 20 dB and 30 dB. This was done to assess the model's performance when audio files are subjected to varying levels of noise in post-processing.

SNR is a measure of the level of the desired signal relative to the level of background noise. The ratio is expressed in decibels (dB), and a higher SNR indicates a cleaner signal with less noise, while a lower SNR indicates a noisier signal. Specifically, a SNR of 20 dB means that the signal level is 20 dB higher than the noise level, and a SNR of 30 dB means that the signal level is 30 dB higher than the noise level. Thus, a lower SNR value (e.g., 20 dB) corresponds to a higher amount of noise in the audio file, making it more challenging to discern the original signal from the noise [31].

The noise was added using the `torchaudio.functional.add_noise` function from the torchaudio library. The function modifies the audio file by incorporating noise at a specified SNR level [32].

Audio Compression

Audio compression is a common post-processing technique used to reduce the size of audio files. The effects of lossy compression on the classification model were explored by compressing audio files to 32 kbps and 64 kbps. Compression at lower bitrates typically results in a loss of audio quality, which can serve as a method for adversaries to mask forgery artefacts.

The compression was done using `AudioSegment` from the pydub library [33], which reduce the file size while attempting to maintain as much quality as possible. In this context, lower bitrates represent a higher degree of compression, potentially affecting the integrity of the audio file and its detectable features.

4.6 Method for detecting copy-move forgeries in long audio files using CNN

Given that the forged and original Mel spectrograms of the audio files in NB Tale Part 3 are significantly longer than the samples used for training, validation, and testing, they were segmented into smaller chunks which were more comparable in length to the other data. The segmentation was performed based on a fixed duration of 9 seconds, corresponding to a specific number of time frames in the Mel spectrogram, determined by the sampling rate and hop length. The segmentation process split each long spectrogram into overlapping chunks, with an overlap of 50% to capture sufficient temporal context in each chunk. If the spectrogram was shorter than the chunk length, it was kept as a single segment.

For each spectrogram, each chunk was normalised using the mean and standard deviation derived from the training data and processed individually by the CNN to extract predictions. The probability of each chunk being forged was evaluated against a predefined threshold, and the predictions were smoothed using a windowed approach. The final classification for the full audio file was determined by the majority voting of the smoothed segment predictions.

Chapter 5

Results

This chapter presents the results from training and testing the Convolutional Neural Network (CNN) model. This section focuses on the presentation of the results without analysis, which will be provided in Chapter 6, along with the discussion of the results.

5.1 Static dataset

This section presents the initial training and testing results on the static copy-move dataset. This experiment validated the CNN model's ability to learn features and generalise to data it was not trained on. The results showed high accuracy for the training, validation, and test datasets, with respective values of 0.985, 0.937, and 0.890. Training was done using a learning rate of 0.001 with a target of 100 epochs, stopping early after 36 epochs due to the early stopping criterion being met at epoch 26. The model's weights at epoch 26 were saved as a checkpoint. Figure 5.1 shows the plot of training vs. validation loss over the epochs, and the plot of training vs. validation accuracy over the epochs.

The best model state was saved as a checkpoint during training. This checkpoint of weights was then loaded into the model which did a classification of a new, unseen test dataset. The result produced from the test is a classification report, which is detailed in table 5.1.

	Precision	Recall	F1-Score	Support
Original	0.88	0.89	0.89	1194
Forged	0.89	0.88	0.88	1194
Accuracy			0.89	2388
Macro avg	0.89	0.89	0.89	2388
Weighted avg	0.89	0.89	0.89	2388

Table 5.1: Classification report generated by testing the trained CNN1 on the datset of long audio files.

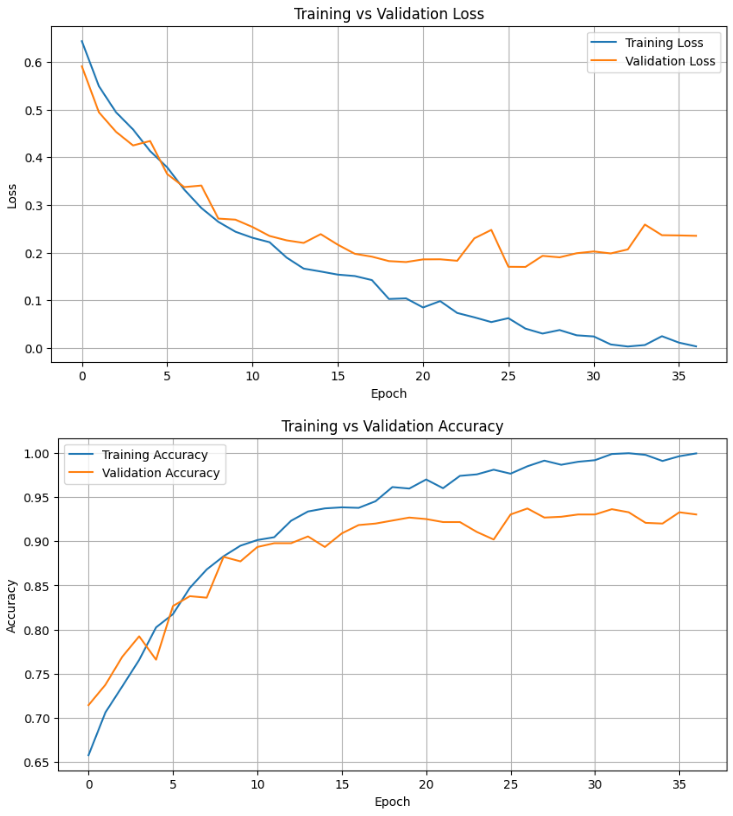

Figure 5.1: Training loss vs validation loss and training vs. validation accuracy of CNN1 on the static dataset.

5.2 VAD-based dataset

This section presents the results of training and testing on the final VAD-based dataset. It includes subsections that detail the results on the extended dataset with augmented test files, as well as the test results on the dataset with longer audio files.

5.2.1 Training and validation results

For testing on the VAD-based dataset, two models were used: the CNN described in the method chapter (CNN1), as well as a CNN-model (CNN2) based on the one proposed by Ustubioglu, Ustubioglu and Ulutas [27] [18]. The results from this testing determined which model was used for the extended testing.

During training, accuracy and loss on the training and validation splits were logged. Table 5.2 shows these values at the epoch where the validation loss stopped improving. Both models were trained with and without data augmentation, where random augmentations were applied to the Mel spectrograms of the training files to increase the model's robustness and its ability to generalise to unseen samples.

CNN1 was first trained without data augmentation, using a learning rate of 0.001 and a batch size of 64 over 75 epochs. The best model state was achieved at epoch 19, with a training loss of 0.6068, training accuracy of 0.6564, validation loss of 0.6444, and validation accuracy of 0.6117. Without data augmentation, the training took approximately 3 seconds. Figure 5.2 visualises the loss and accuracy curves over the epochs. The plot shows that the validation loss starts to rapidly increase around epoch 20 as the training loss decreases, indicating overfitting.

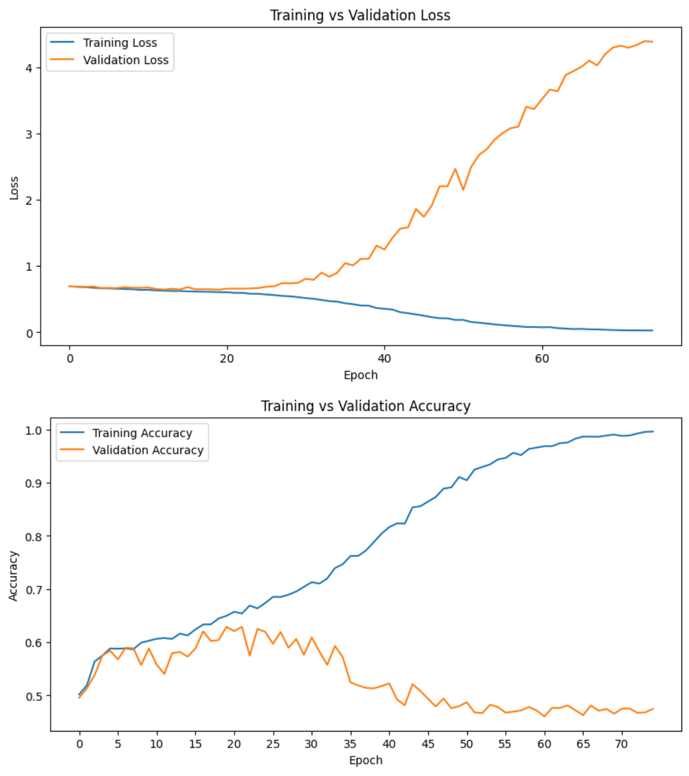

Figure 5.2: Training vs validation loss and training vs validation accuracy during training CNN1 without data augmentation.

CNN1 was also trained with data augmentation, using a learning rate of 0.002 and a batch size of 64 for 400 epochs. The best model state was achieved at epoch 371, with a training loss of 0.5000, training accuracy of 0.7402, validation loss of 0.5304, and validation accuracy of 0.7189. With data augmentation, the training took approximately 14 seconds per epoch. Figure 5.3 visualises the loss and accuracy curves over the epochs, showing how the model reaches convergence, indicated by stability in the metrics, around epoch 380.

CNN2 reached its best validation accuracy of 68.8% and validation loss of 0.5784 after 18 epochs without data augmentation.

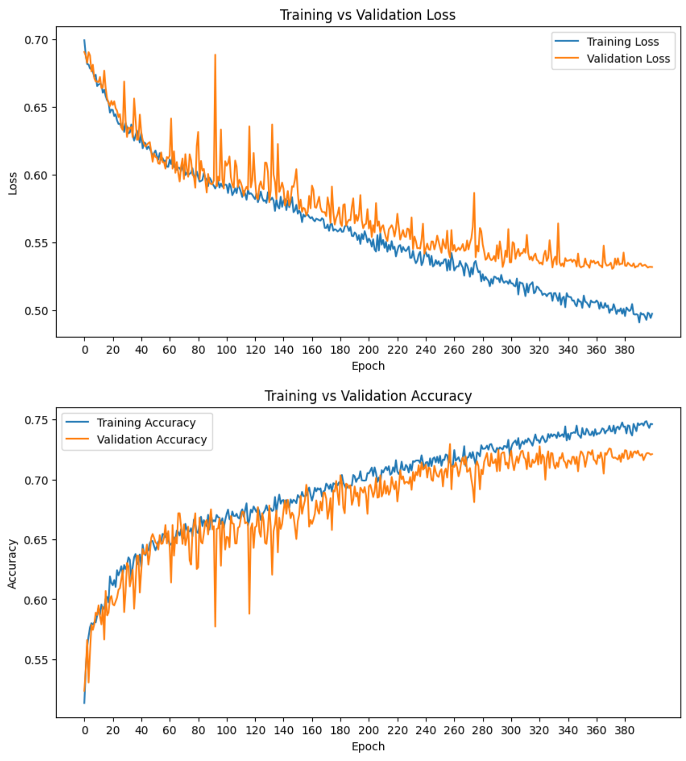

Figure 5.3: Training vs validation loss and training vs validation accuracy during training of CNN1 with data augmentation.

Model	Epochs	T loss	T acc	V loss	V acc	Aug
CNN1	19	0.6068	0.6564	0.6444	0.6117	No
CNN1	371	0.5000	0.7402	0.5304	0.7189	Yes
CNN2	18	0.5573	0.7030	0.5784	0.6880	No
CNN2	185	0.6161	0.6415	0.5890	0.6800	Yes

Table 5.2: Training and validation results of the two CNN models on the VAD-based dataset. T = training, V = Validation, Aug. indicates if the training data was augmented during training.

5.2.2 Test results

The baseline classification performance of the final trained CNN1 model was evaluated on the test split of the dataset, using precision, recall, F1-score, and support metrics for both the original and forged classes. The results are summarised in Table 5.3. The model achieved a precision of 0.74 for the original class and 0.70 for the forged class. The recall values were 0.71 and 0.73 for the original and forged classes, respectively. This indicates that the model correctly identified 71% of the original samples and 73% of the forged samples. The F1-scores, which

	Precision	Recall	F1-Score	Support
Original	0.74	0.71	0.72	556
Forged	0.70	0.73	0.71	516
Accuracy			0.72	1072
Macro avg	0.72	0.72	0.72	1072
Weighted avg	0.72	0.72	0.72	1072

Table 5.3: Classification Report generated by testing the trained CNN1 on the test split.

balance precision and recall, were 0.72 and 0.71 respectively for the original and forged samples.

Overall, the model achieved an accuracy of 0.72 across 1072 samples. The macro average and weighted average metrics also reflected a balanced performance, each scoring 0.72 for precision, recall, and F1-score.

These results indicate that the model performed similarly on both classes, with a slight variation in recall between the original and forged categories. The balanced f1-scores suggest a consistent performance in handling both classes of data.

The model's performance on the test data is further illustrated by the confusion matrix shown in Figure 5.4. The confusion matrix provides a detailed breakdown of the model's classification results, indicating the number of correct and incorrect predictions for each class.

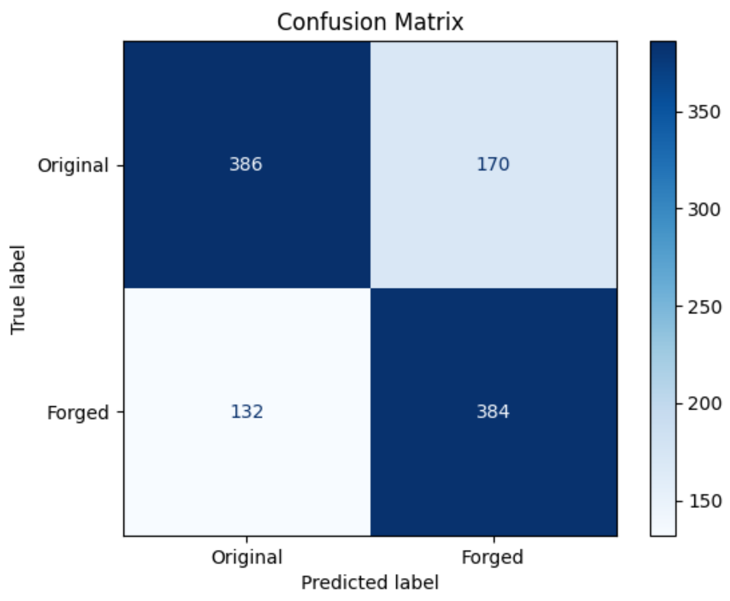

Figure 5.4: Confusion matrix from testing the trained CNN1 on the test split.

The confusion matrix shows that out of 534 original samples, 370 were correctly classified. Similarly, out of 538 forged samples, 386 were correctly classified.

These results reflect the model's performance in distinguishing between original and forged samples, showing a relatively balanced distribution of correct and incorrect classifications for both classes. The confusion matrix highlights the areas where the model performed well and where there is room for improvement, particularly in reducing the number of false positives for the forged class.

Post-processed test samples

The classification performance of the model was evaluated on test samples with different post-processing operations applied to the samples, to simulate scenarios where an adversary might process forged audio files to conceal the forgery. Table 5.4 presents the results of these tests, including accuracy, precision, recall, and F1-score for various augmentation methods.

Augmentation	Accuracy	Precision	Recall	F1
20 dB SNR noise	0.5606	0.6658	0.5606	0.4936
30 dB SNR noise	0.7024	0.7029	0.7024	0.7025
32 kbps bitrate	0.7164	0.7172	0.7164	0.7165
64 kbps bitrate	0.7164	0.7172	0.7164	0.7165

Table 5.4: Results from testing the trained CNN1 with additional post-processing.

- With 20 dB SNR noise the model achieved an accuracy of 0.5606, precision of 0.6658, recall of 0.5606, and an F1-score of 0.4936.
- With 30 dB SNR noise the model achieved an accuracy of 0.7024, precision of 0.7029, recall of 0.7024, and an F1-score of 0.7025.
- Compression to 32 kbps bitrate resulted in an accuracy of 0.7164, precision of 0.7172, recall of 0.7164, and an F1-score of 0.7165.
- Compression to 64 kbps bitrate yielded an accuracy of 0.7164, precision of 0.7172, recall of 0.7164, and an F1-score of 0.7165

These results indicate how different types of noise and compression affect the model's ability to classify forged and original audio files. The model's performance is generally robust, but notably worse with the 20 dB SNR noise addition, indicating a weakness to high levels of noise.

5.3 Long audio files

The performance of CNN1 was further evaluated on longer audio files to understand its ability to classify original and forged audio under extended durations. The results are summarised in the confusion matrix shown in Figure 5.5 and the classification report in Table 5.5.

The confusion matrix in Figure 5.5 illustrate the following results:

	Precision	**Recall**	**F1-Score**	**Support**
Original	0.52	0.38	0.44	365
Forged	0.51	0.65	0.57	365
Accuracy			0.52	730
Macro avg	0.52	0.52	0.51	730
Weighted avg	0.52	0.52	0.51	730

Table 5.5: Classification report generated by testing the trained CNN1 on the datset of long audio files.

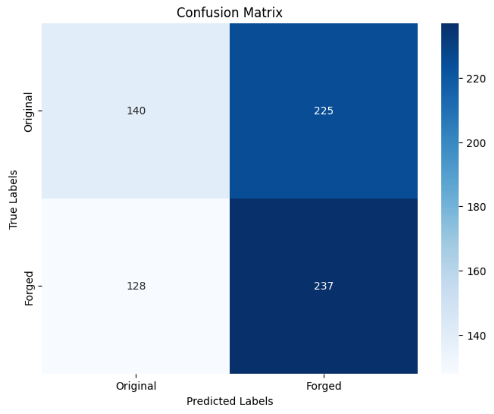

Figure 5.5: Confusion matrix of testing the trained CNN1 on the dataset of long audio files.

- **Original class**: Out of 365 original audio samples, 140 were correctly classified while 225 were misclassified as forged.
- **Forged class**: Out of 365 forged audio samples, 237 were correctly classified, while 128 were misclassified as original.

Table 5.5 provides detailed metrics for the classification performance. The precision for the original class was 0.52, indicating that 52% of the samples identified as original were correctly classified. The precision for the forged class was 0.51, meaning that 51% of the samples identified as forged were correctly classified. The recall of the original class was 0.38, for the forged class the recall was 0.65. The F1-score was 0.44 for the original class and 0.57 for the forged class. The overall accuracy of the model on the longer audio files was 0.52, meaning that 52% of the total samples were correctly classified. The macro average precision, recall, and F1-scores were 0.52, 0.52, and 0.51, respectively, while the weighted average metrics were 0.52 for precision and recall, and 0.51 for F1-score.

These results show that the model do not generalise to the longer samples. The precision and recall values suggest that while the model is relatively better at identifying forged audio, it struggles more with correctly classifying original audio files. This may suggest that the model is biased towards classifying samples as forged.

Chapter 6

Discussion

In this chapter, the various aspects of the thesis project are discussed in detail with the goal of addressing the identified research questions. This includes an examination of the processes involved, the choices made, the challenges encountered, and the lessons learned throughout the project. Each section addresses specific components of the research, providing insight into the advantages and limitations of the methodologies employed. The discussion also explores potential improvements and the broader implications of the research, discussing how improved audio forgery detection can benefit forensic investigations and judicial processes. Based on the findings in the results and discussion, the research questions are answered as follows.

6.1 Research questions

(RQ1) To what extent can Mel spectrogram-based CNN models be applied to copy-move forgery detection in realistic forensic scenarios?

Mel spectrogram-based CNN models can be applied to copy-move forgery detection in realistic forensic scenarios, although their effectiveness varies with the conditions of the audio samples. The results from CNN1, trained with data augmentation on the VAD-based dataset, demonstrate that this model can perform well in more realistic forensic scenarios than what is demonstrated in the related work. This dataset has significant variance, particularly in terms of audio length, making it more representative of real-world scenarios than datasets used in the literature. The model achieved high accuracy on held-out test datas, indicating its ability to generalise well to unseen samples

Extended tests show that the model is robust to common post-processing operations, namely noise addition and compression, which are techniques that might be used to obscure a forgery. The model maintained a high accuracy when subjected to these conditions, with the exception of very high noise levels, under which accuracy dropped significantly. This indicates that the model can detect forgeries even when there have been attempts to hide them through these specific audio manipulations. Mel spectrogram-based CNN models are thus promising for the application of forensic tools, particularly in detecting manipulated audio where the forgeries are not immediately obvious by ear.

RQ2 What are the limitations of Mel spectrogram-based CNN models for copy-move forgery detection in forensic scenarios?"

While the Mel spectrogram-based CNN-model show potential in detecting CMF, it also has several limitations that impact its effectiveness in forensic scenarios. A significant limitation is its inability to generalise to longer audio files. The test results on the dataset created from NB-tale part 3, despite being tailored to favour the model's performance, showed poor classification performance on forgeries in lengthy audio files. Another limitation is the model's susceptibility to high levels of noise. While the model performs well under typical conditions, with compression, and with low levels of noise, its accuracy significantly decreases when faced with higher levels of noise. This can be a considerable drawback in forensic scenarios, in which audio quality can vary widely.

Currently Mel spectrogram-based CNN models lack the capability to localise forgeries within an audio file. They can potentially identify whether a file is forged but cannot pinpoint the exact segments that have been manipulated. Although not a deal breaker, this limitation negatively impacts their utility in forensics scenarios.

6.2 Generating datasets

Despite automatic audio CMFD being an active field of research, there is a lack of publicly available datasets of audio copy-move forgeries. In the specific case of the Oslo Police Department, the audio files they process are often in Norwegian, and there are no Norwegian CMF datasets available. The NB Tale dataset was chosen as the source for generating the dataset for three main reasons: (1) The language of the recordings is Norwegian, spanning a large number of dialects and ethnolects; (2) Module 1 and 2 contain short recordings (3 to 24 seconds), making them suitable for creating forgeries similar to previous studies; and (3) Module 3 consists of longer audio files, facilitating forgeries that simulate more realistic scenarios for evaluating the model's performance on extended recordings.

To facilitate the testing and evaluation of the CMFD-method, two approaches from the literature for generating the forgeries were explored: static copy-move and VAD-based copy-move.

The static copy-move dataset was created by copying a static one second segment of the audio from a specific location in the file and pasting it at another specific location. An advantage of this method is that it was much easier to effectively replicate than the VAD-based methods described in the literature. While the studies using VAD describes how the datasets were generated, it is hard to have a general approach work for detecting segments of speech in short audio files across different dataset sources. The static method does not require any parameter tuning other than choosing an appropriate place to copy the audio from, depending on the length of the audio file.

However, as the static copy-move forgery creates a file by copying and *inserting* the forged audio at the other location, one concern with using this method for dataset creation is that the trained model gets biased towards classifying longer spectrograms as forged and shorter spectrograms as original. Another issue is the static nature of the location of the forgeries, being restricted to two specific seconds in the audio file. This introduces bias in the model's learning process, as it may overfit to the specific temporal patterns of forgery rather than generalising to different potential locations and variations of forgeries. Consequently, the model might fail to detect forgeries that deviate from these predefined patterns, reducing its effectiveness in real-world scenarios where the location and nature of forgeries can vary widely.

To address these limitations, a more dynamic dataset was created that includes forgeries at random positions and varying lengths within the audio files. Additionally, employing data augmentation techniques during training to create a diverse set of forgeries can help the model learn more generalised features that are not dependent on specific forgery patterns.

The VAD-based copy-move dataset was created using YAAPT, identifying the voiced segments of the file, choosing one of them at random with a minimum length of 0.4 seconds and pasting it at another random location in the file. Initially, SileroVAD was used as the VAD-method, but it struggled to single out words and short sounds as individual segments of speech, and would often lead to very large parts of the audio being used as the basis for the forgery, which made the classification overly simple. The YAAPT-algorithm was to a greater extent able to identify short voiced segments in the audio files consistently. By creating the dataset with this method, we are able to simulate more realistic forgeries by ensuring that:

- The copied segments are of variable length.
- The copied segment can be anywhere in the recording.
- The location of the pasted segment can be anywhere in the file.
- The forged and original version of the same file retains the same length.
- The forged files have no common features which differentiate them from the original files, other than the forgery itself.

Because of the variance in the dataset, it becomes substantially more difficult to train the CNN to learn features which are applicable to the entire dataset. This is evident when comparing the results of training the models on this dataset to those from the static dataset. Initially, when switching over to training on the VAD-based dataset, several adjustments had to be made to the data pre-processing, CNN architecture, and training strategy.

In pre-processing, the data augmentation strategy had to be changed. Previously, each spectrogram was pre-made with a randomly generated frequency mask, which covered up a part of the image along the x-axis, simulating missing information for a small range of frequencies. However, this technique led to the model overfitting to the training data quickly. One theory for this overfitting is that the static nature of the augmentation did not effectively extend the dataset. Although the frequency masks were applied at a random amplitude for each spectrogram, the pre-loaded masks did not provide enough variability.

The way this was solved was to apply another form of image augmentations on the fly, using the "RandomAffine" transformation from the torchvision library. This method randomly generates a new version of each training sample for each epoch, making it harder for the model to memorise the training data, forcing it to learn the general features despite these augmentations. One trade-off for doing data augmentation on the fly is the increase in computational complexity during training and epochs needed for the model to converge. The average time used per epoch for training CNN1 without data augmentation was 3 seconds, while it took 14 seconds per epoch with the data augmentation. The epochs needed with and without data augmentation for the model to reach its lowest validation accuracy was 19 and 371, respectively.

In addition, dropout was introduced into the CNN architecture. Dropout is a regularisation technique that helps prevent overfitting by randomly "dropping out" a fraction of the neurons during training. This forces the model to learn redundant representations and makes it more robust. With the increased complexity and variability of the VAD-based dataset, adding dropout layers helped to further reduce overfitting. Specifically, dropout was applied after each convolutional block with a dropout rate of 0.5, ensuring that the model did not rely too heavily on any single neuron and instead learned more generalised features across the dataset.

Furthermore, weight decay, also known as L2-regularisation, was added to the optimiser during training to penalise large weights and further reduce the risk of overfitting. This regularisation technique adds a term to the loss function proportional to the squared value of the weights, encouraging the model to maintain smaller and more generalised weights. By integrating these three changes to the pre-processing, model architecture and training, the model's ability to generalise from the training data to unseen samples significantly improved.

To summarise, the static method was straightforward and easy to implement but introduced biases due to its predictable forgery patterns. In contrast, the VAD-based method created more realistic forgeries but was more complex to implement. The dynamic nature of the VAD-based forgeries made it harder for the model to overfit, improving its ability to generalise. One of the key challenges encountered was ensuring the dynamic nature of the VAD-based forgeries did not introduce unintended biases. This was addressed by varying the lengths and insertion points of the copied segments. Another learning was the importance of dynamic data augmentation, which facilitated convergence during training, although at a slower pace. Future iterations of the dataset creation process could involve incorporating additional sources of audio to reduce dataset-specific biases. Moreover, manually creating forgeries with even more varied patterns could further enhance the dataset's realism. Utilising more sophisticated data augmentation techniques could also improve the robustness of the training process. The creation of a Norwegian CMF dataset enhances the capability of forensic audio analysis in Norway. It provides a resource for training models to detect forgeries in Norwegian audio files, which can improve the accuracy and reliability of forensic investigations.

6.3 Training and testing the CNN model

The initial training and testing of the CNN-model (CNN1) on the static dataset achieved good results on the training and validation data in terms of accuracy and loss, as well as good results on the held-out test set using a different source dataset. These results are the ones that align most with the results in previous research. It validates that CNN1 is able to learn features from data. Because of the inherent biases and limitations associated with the static dataset discussed in the previous section, it was only used as a theoretical baseline for the CNN classifier performance, proving that the model is able to learn features from spectrograms.

The results from training and doing continuous validation with the VAD-based dataset highlight the importance of augmenting the training data for this task to reduce overfitting and improve the model's ability to generalise to unseen data. Without it, the validation loss stops decreasing and starts to increase at a relatively early epoch while the training loss keeps improving, signifying overfitting to the training data. The data augmentation steps makes the training data much harder to "memorise" for the model by introducing some random variance to the samples per epoch. One negative consequence to this technique is that the learning of the model takes a lot more time, but in return we get a model that is significantly better able to generalise its learned features, which is demonstrated in the increased performance on the validation split. Selecting parameters and model architecture involved careful consideration of trade-offs between training efficiency and model robustness. For instance, increasing dropout rates and weight decay improved generalisability but also extended training times.

As mentioned earlier, the VAD-dataset was created with a goal of ensuring that the only discernible difference between the forged and original version of an audio file is the forgery. Together with a lot more variance in terms of audio length of the source data and location of the forgeries, this better simulates real-life cases and results in a hard classification task for the CNN. The classification difficulty of the dataset is further validated by attempting to train the other CNN-model (CNN2). CNN2 showed significantly worse results on this dataset when compared to the reported accuracy of 99% from the article [5].

Training and validating the CNN models revealed notable differences between CNN1 and CNN2. CNN1 showed a relatively strong performance, which could be attributed to its deep architecture with more filters than CNN2. CNN2, while simpler and less resource-intensive, struggled to achieve the same accuracy on the VAD-based dataset, indicating its limitations in handling more complex and variable data.

A broader range of data augmentation techniques could potentially be applied during training. But given the resource heavy nature of doing data augmentations on the fly, it was not deemed feasible for this work. Additionally, experimenting with alternatives such as transformers or hybrid models combining CNNs and Recurrent Neural Networks (RNNs) could offer improved performance on extended-length audio data. A trained model that is able to efficiently generalise to longer audio files could potentially improve forensic audio analysis by providing a tool currently not available for detecting audio forgeries.

6.4 Testing on post-processed samples

Forensic analysts encounter audio files with a high variance in their condition. Skilled adversaries looking to compromise the integrity of audio files with copy-move forgery will likely make an attempt to hide the forgery so that it is not so easy to detect by ear.

One way of doing this may be to add noise or compressing the audio to reduce the quality of the audio. To test the model's robustness under these conditions tests were conducted on four different datasets with varying grades of compression or noise addition. The results indicate that the model's accuracy decreases as the quality of audio is compromised, highlighting the importance of training with diverse augmentations to enhance robustness.

The classification performance of the model was evaluated on test samples which had been compressed, and noise added to them to simulate scenarios where an adversary might process forged audio files to hide the forgery. The results of these tests indicate how different types of noise and compression affect the model's ability to classify the audio files.

The results indicated that the model's accuracy decreases as the quality of audio is compromised, highlighting the importance of training with diverse augmentations to enhance robustness. The model performed notably worse with the 20 dB SNR noise addition, indicating a weakness to high levels of noise.

These findings underscore the need for further research and development to improve the model's resilience to adversarial tactics, ensuring robust performance across a variety of challenging audio conditions. Future work could involve training with a wider range of augmentations and testing under more varied post-processing conditions.

6.5 Application to real-world forensic settings

As no literature describes how automatic copy-move forgery methods would be implemented in practice with longer audio files, this area required significant trial and error.

The general idea was to divide the audio file into smaller chunks comparable in length to the training dataset, classify each chunk, and aggregate the results. However, this approach presented concerns, such as the model's inability to recognise forgeries when the copied and pasted segments are far apart in time. Since the training dataset consists of files where both the copied and the pasted segment is present within a short time frame, limited by the section of speech in the recording, one concern is that if the copied and pasted segment in the long audio file are far apart in time, the model will not be able to recognise the forged section as it is trained to look for identical or similar segments of speech rather than indications of forgery.

One method which was tested to combat this was to generate a dataset of forgeries for training where a segment is copied, deleted then pasted elsewhere in the file. This method was scrapped quickly as testing with extensive parameter tuning proved that under no circumstances did the CNN-model successfully generalise to the validation data-split. The problem is likely that since the forged segment originates from the same audio file, there are no discernible features of the forged segment which the model is able to learn - making it a splicing forgery without the indications of forgery typically associated with splicing like difference in background noise or changes in the quality or characteristics of the audio in the spliced segment. One idea to solve this could be to slightly alter some properties of the pasted segment like its pitch or volume in an attempt to make the forgeries stand out more from the rest of the file, but then again this would simulate detection of splicing forgeries, which ultimately is not the goal of the model.

Since this did not work, the forgeries used for testing on long audio files were made in a way that ensures that both the copied and pasted segments are within a limited range based on the chunk size. This looks to make sure that the two identical segments are "seen" by the model simultaneously when using overlapping segments. This condition for the forgeries is advantageous in favour of the model's chance of success and is not a good way to represent realistic forgeries. This was however done initially as a test to see if classifying chunks and aggregation the results would work at all. Despite this advantageous condition, the results indicated that even with this artificial setup, the model struggled to accurately detect forgeries in longer audio files.

The results, as shown in Table 5.5 and Figure 5.5, reveal a significant number of false positives, where original files were incorrectly classified as forged. Specifically, the model exhibited a precision of 0.52 and a recall of 0.38 for original files, indicating a substantial rate of misclassification. This suggests that the model is overly sensitive to patterns that it interprets as indicative of forgery, even in authentic audio files. The high false positive rate can undermine the reliability of the model in forensic applications, where accurate identification of authentic files is just as crucial as detecting forgeries. This issue highlights several challenges and areas for future improvement. Developing methods that allow the model to consider broader context within the audio files could improve its ability to accurately classify long audio recordings. Techniques such as attention mechanisms or hybrid models that combine CNNs with Recurrent Neural Networks (RNNs) might help maintain contextual awareness over an extended duration. Attention mechanisms can potentially enhance the model's ability to focus on relevant parts of the audio when making predictions, effectively enabling the model to "pay attention" to important features that are dispersed throughout a long audio file. While CNNs are effective at extracting spatial features from Mel spectrograms, RNNs are adept at handling sequential data and capturing temporal dependencies. A hybrid model could first use the CNN to extract features from the spectrogram, and then use RNNs to analyse the sequence of these features, potentially improving the model's ability to detect forgeries over extended timeframes.

The poor results of classifying the longer audio files signify that training a CNN-model for classification of CMF on short audio files is not a sufficient methodology for training the model for real-life scenarios. This study identifies this as the primary challenges and limitation in the field of automatic CMFD and highlights how hard it is for a CNN-based model to successfully detect forgeries where the original audio content is reused to create forgeries over extended durations.

The field of automatic AAA in general, and automatic CMF-detection specifically, would greatly benefit from having a publicly available dataset of realistic copy-move forgeries with the type of variance forensic analysts encounter in their work. There are several problems with the current methods for generating copy-move forgeries which do not translate to realistic cases. The most obvious factor being the length of the files. In the samples used to train theCMFD it is relatively easy to detect the forgeries by ear when the copied and pasted segment are so close together. This questions the usefulness of an automaticCMFD system if it is not able to transfer the features it learns from these short samples to longer audio files. So while the existing methods are proofs of concept, the question still remains if applying them in the daily operations of forensic analysts is feasible. The lack of any datasets with realistic CMF samples makes it difficult to assess how any other automatic ML-based methods would perform when applied in real-world settings from the literature alone. It is therefore hard to say how useful any of them are in reality.

6.6 Model interpretability

Classification of an audio file is useful in forensic contexts to get an indication whether a file has been forged. However, adding a method for model interpretability could provide more detailed insights into why the model classified the input as forged. One potential solution is the implementation of Gradient-weighted Class Activation Mapping (Grad-CAM) [34]. It creates a heat map which signifies which parts of the input were most influential on the classification decision, and thus signifying where the classified forgeries are in the file. This can be implemented after classification by providing the predicted label as target label to increase the interpretability of the classification. This has just been explored through some initial tests, but it could be an area to look into for developing a fully operational

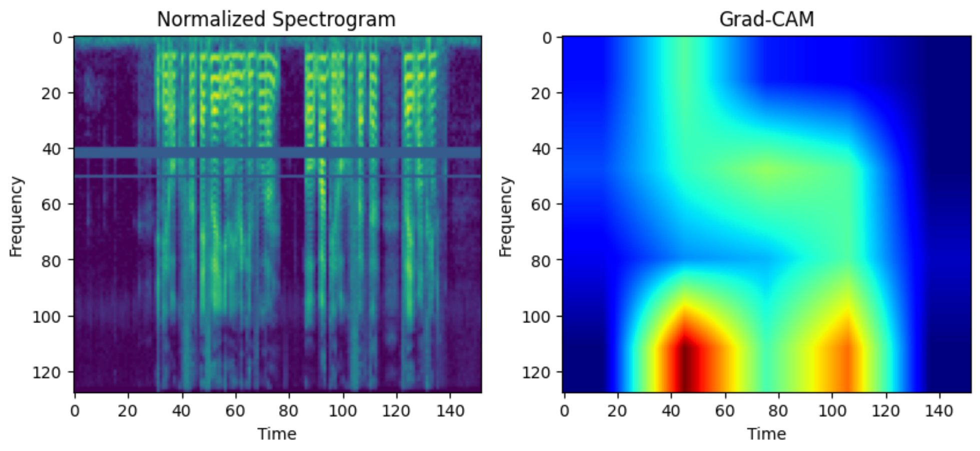

Figure 6.1: Example of Grad-CAM for a copy-move forgery from the test dataset which the model correctly classified as forged.

automatic CMFD-solution with a CNN as the backbone, if the other issues are resolved. Table 6.1 visualises a Mel spectrogram as well as its associated Grad-CAM plot, where the more intense colours signify which parts of the spectrogram was most influential on the decision.

6.7 Limitations and future work

Despite some promising results, several limitations and concerns were identified in this thesis work. The reliance on a single source datasbase (NB Tale) may introduce biases related to recording conditions and speaker demographics. Additionally, the lack of publicly available CMF datasets limits the comparison of the findings to other methods from previous work. The CNN model demonstrated good performance on short audio files, but it struggled with longer recordings, indicating the need for more advanced techniques to handle diverse and extended audio inputs.

A significant challenge of copy-move forgery detection using current Mel-spectrogram based CNNs is their dependence on "seeing" both the copied and the pasted segments simultaneously. This limitation arises because these models are typically trained on relatively short audio clips where the copied and pasted segments are in close proximity, allowing the CNN to detect the repetitive patterns within a single spectrogram. However, in real-world scenarios, especially with longer audio files, the copied and pasted segments may be separated by significant time intervals. As a result, the model's inability to process long sequences in their entirety means it often fails to identify forgeries when the copied segment and its duplicate are far apart. This gap in detection capability highlights the need for advanced techniques that can maintain context over longer durations or alternative methods that can bridge the temporal distance between forgeries within long audio files. Without addressing this limitation, the effectiveness of Mel-spectrogram based CNNs in practical forensic applications remains compromised, necessitating further research and development to enhance their robustness in diverse and realistic scenarios.

Future work should focus on creating a comprehensive dataset with realistic forgeries, diverse recording conditions, and longer audio files to better simulate forensic scenarios. Furthermore, researching new methods and algorithms that can effectively handle long audio files should be the primary focus of the field moving forward. Also, implementing interpretability methods like Grad-CAM to provide forensic analysts with insight into the model's decision making process, should be part of a fully operational automatic audio authenticity analysis solution.

Also, doing extended tests on different post-processing operations like applying median filtering, pitch shifting, or frequency scaling to the test samples, in addition to noise and compression, should be considered to further evaluate resilience to different types of audio manipulations.

The development of robust forgery detection methods has significant implications. By enhancing the ability to detect manipulated audio recordings, these methods can improve the integrity of audio evidence in legal and forensic contexts. This, in turn, can contribute to more accurate and fair juridical outcomes. Additionally, the technology can be applied in media and journalism to verify the authenticity of audio content, thereby combating misinformation.

Even if this necessitates manually creating the dataset, future work could benefit from incorporating a wider range of audio sources and even more realistic forging techniques to create a more comprehensive dataset. Even though it would be an extensive and time-consuming process, the field would profit immensely from it. To this end, research in the field going forward should concentrate on developing techniques that can successfully classify and handle full-length audio files. The existing methods are designed to work well on short audio files; while this shows that the approaches can be applied in theory, it does not guarantee that the results will hold up in practical situations.

Chapter 7

Conclusion

In this thesis, methods for detecting copy-move forgeries in audio files using a Convolutional Neural Network (CNN) model based on Mel-spectrograms has been developed and evaluated. The project created a novel dataset of Norwegian audio files with both static and VAD-based forgeries, providing a valuable resource for training and testing audio forgery detection models. Through extensive experimentation, it was demonstrated that CNN models could effectively learn to identify forgery patterns in short audio clips. The introduction of dynamic data augmentation and regularisation techniques such as dropout and weight decay significantly improved the model's generalisation capabilities, addressing overfitting issues encountered during training.

Despite these promising results, the project identified critical limitations, particularly the model's deficiencies to detect forgeries in longer audio files where copied and pasted segments are far apart. This highlighted the need for future research in developing techniques that can handle extended audio sequences and maintain context over longer durations. The model's performance on test samples with various post-processing operations applied underscored its robustness against common audio manipulations. However, the notable decline in accuracy with high noise levels (20 dB SNR) highlighted a vulnerability that needs to be addressed for reliable forensic applications. Additionally, the project explored initial steps towards model interpretability with Grad-CAM, which showed promise in providing forensic analysts with insights into the model's decision-making process.

Looking forward, future work should focus on creating more comprehensive datasets with realistic forgeries and diverse recording conditions to better simulate forensic scenarios. Further research is also needed to develop frameworks for handling long audio files and enhancing the robustness of forgery detection methods. By addressing these challenges, the field of automatic audio authenticity analysis can significantly improve, providing more reliable tools for forensic investigations.

Bibliography

[1] F. Akdeniz and Y. Becerikli, 'Detecting audio copy-move forgery with an artificial neural network,' *Signal, Image and Video Processing*, pp. 1–17, 2024.

[2] A. Bartle, D. Boss, A. G. Boyarov, L. Cuccovillo, C. Grigoras, M. Michałek and D. Nyberg, 'Best practice manual for digital audio authenticity analysis,' 2022.

[3] P. R. Bevinamarad and M. Shirldonkar, 'Audio forgery detection techniques: Present and past review,' in *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, IEEE, 2020, pp. 613–618.

[4] T. Skogseth, 'Detection of edited media files based on sound,' Department of Information Security, Communication NTNU – Norwegian University of Science and Technology, Project report in IMT4205, Dec. 2023.

[5] A. Ustubioglu, B. Ustubioglu and G. Ulutas, 'Mel spectrogram-based audio forgery detection using cnn,' *Signal, Image and Video Processing*, vol. 17, no. 5, pp. 2211–2219, 2023.

[6] N. L. of Norway, *Nb tale - speech database for norwegian - språkbanken*, Dec. 2015. [Online]. Available: `https://www.nb.no/sprakbanken/en/resource-catalogue/oai-nb-no-sbr-31/`.

[7] H. Face, *Introduction to audio data - hugging face audio course*. [Online]. Available: `https://huggingface.co/learn/audio-course/chapter1/audio_data`.

[8] S. S. Stevens, J. Volkmann and E. B. Newman, 'A scale for the measurement of the psychological magnitude pitch,' *The journal of the acoustical society of america*, vol. 8, no. 3, pp. 185–190, 1937.

[9] G. Hua, Y. Zhang, J. Goh and V. L. Thing, 'Audio authentication by exploring the absolute-error-map of enf signals,' *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 1003–1016, 2016.

[10] H.-P. Hsu, Z.-R. Jiang, L.-Y. Li, T.-C. Tsai, C.-H. Hung, S.-C. Chang, S.-S. Wang and S.-H. Fang, 'Detection of audio tampering based on electric network frequency signal,' *Sensors*, vol. 23, no. 16, p. 7029, 2023.

[11] D. U. Leonzio, L. Cuccovillo, P. Bestagini, M. Marcon, P. Aichroth and S. Tubaro, 'Audio splicing detection and localization based on acquisition device traces,' *IEEE Transactions on Information Forensics and Security*, 2023.

[12] K. O'Shea and R. Nash, 'An introduction to convolutional neural networks,' *arXiv preprint arXiv:1511.08458*, 2015.

[13] K. He, X. Zhang, S. Ren and J. Sun, 'Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,' in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[14] S. Ioffe and C. Szegedy, 'Batch normalization: Accelerating deep network training by reducing internal covariate shift,' in *International conference on machine learning*, pmlr, 2015, pp. 448–456.

[15] Y. Ho and S. Wookey, 'The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling,' *IEEE access*, vol. 8, pp. 4806–4813, 2019.

[16] P. Contributors, *Crossentropyloss*. [Online]. Available: `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html`.

[17] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization,' *arXiv preprint arXiv:1412.6980*, 2014.

[18] B. Ustubioglu, G. Tahaoglu and G. Ulutas, 'Detection of audio copy-move-forgery with novel feature matching on mel spectrogram,' *Expert Systems with Applications*, vol. 213, p. 118 963, 2023.

[19] D. M. Powers, 'Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation,' *arXiv preprint arXiv:2010.16061*, 2020.

[20] A. C. Müller and S. Guido, *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.

[21] M. Imran, Z. Ali, S. T. Bakhsh and S. Akram, 'Blind detection of copy-move forgery in digital audio forensics,' *IEEE Access*, vol. 5, pp. 12 843–12 855, 2017.

[22] Q. Yan, R. Yang and J. Huang, 'Robust copy–move detection of speech recording using similarities of pitch and formant,' *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2331–2341, 2019.

[23] C. Liu, J. Li, J. Duan, H. Shen and H. Huang, 'Lightcvt: Audio forgery detection via fusion of light cnn and transformer,' in *Proceedings of the 2021 10th International Conference on Computing and Pattern Recognition*, 2021, pp. 99–105.

[24] B. Ustubioglu, G. Tahaoglu, G. Ulutas, A. Ustubioglu and M. Kilic, 'Audio forgery detection and localization with super-resolution spectrogram and keypoint-based clustering approach,' *The Journal of Supercomputing*, pp. 1–33, 2023.

[25] L. Cuccovillo, M. Gerhardt and P. Aichroth, 'Audio spectrogram transformer for synthetic speech detection via speech formant analysis,' in *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, 2023, pp. 1–6.

[26] K. Doshi, 'Audio deep learning made simple: Sound classification, step-by-step,' Jan. 2022. [Online]. Available: https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5.

[27] A. Ustubioglu, B. Ustubioglu and G. Ulutuas, 'Mel spectrogram-based audio forgery detection using cnn,' in *Signal Image Video Process*, Research Square, vol. 1, 2022. DOI: 10.21203/rs.3.rs-1828771/v1.

[28] T. Skogseth, *Norwegian audio copy move forgery datset*, 2024. DOI: 10.34740/KAGGLE/DS/5167635. [Online]. Available: https://www.kaggle.com/ds/5167635.

[29] K. Kasi and S. A. Zahorian, 'Yet another algorithm for pitch tracking,' in *2002 ieee international conference on acoustics, speech, and signal processing*, IEEE, vol. 1, 2002, pp. I–361.

[30] Q. Yan, R. Yang and J. Huang, 'Copy-move detection of audio recording with pitch similarity,' in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 1782–1786.

[31] dpwe, *Icsi speech faq - how is the snr of a speech example defined?* Accessed: 2024-06-01, 2000. [Online]. Available: https://www1.icsi.berkeley.edu/Speech/faq/speechSNR.html.

[32] P. Contributors, *Torchaudio.functional*. [Online]. Available: https://pytorch.org/audio/main/generated/torchaudio.functional.add_noise.html.

[33] Jiaaro, *Github - jiaaro/pydub: Manipulate audio with a simple and easy high level interface*. [Online]. Available: https://github.com/jiaaro/pydub/.

[34] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, 'Grad-cam: Visual explanations from deep networks via gradient-based localization,' in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.