Master's thesis

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

Gard Eggen Hansen

Artificial Residual Noise in Machine Learning

Master's thesis in Information Security
Supervisor: Lasse Øverlier
Co-supervisor: Kyle Porter
June 2022

NTNU
Kunnskap for en bedre verden

Gard Eggen Hansen

Artificial Residual Noise in Machine Learning

Master's thesis in Information Security
Supervisor: Lasse Øverlier
Co-supervisor: Kyle Porter
June 2022

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

NTNU
Kunnskap for en bedre verden

Artificial Residual Noise in Machine Learning

Gard Eggen Hansen

Acknowledgment

First and foremost, I would like to thank my girlfriend and my family for the support given during the entire period. Without it would not have been possible to complete the thesis. I would like to express my sincere gratitude to my supervisor Lasse Øverlier for all the guidance, discussion, ideas, feedback, and support through the entire period. Further I want to thank my co-supervisor Kyle Porter for taking the time to provide valuable guidance, sharing knowledge, and feedback. The regular meetings that was held, all the discussions and feedback on email helped me reach the desired goals.

G.E.H

Abstract

In the area of audio forensics there has been some skepticism toward the application of noise suppression by machine learning models. Even if noise suppression using machine learning could add both efficiencies and help out handling the amount of data that an investigator would be facing, it has not been adopted as a go-to solution. One major reason for this is likely reliability. When working within the field of forensics, the importance of maintaining integrity and chain of custody related to the data at hand is essential. If these principles are violated, it can make the entire investigation collapse. An example of integrity violation is through the application of machine learning when performing noise suppression, and manifestation of artificial residual noise.

We will therefore investigate artificial residual noise with the goal of better understanding how it can be detected, and possible implications for digital forensics and investigations. Through experimentation we will replicate artificial residual noise in different machine learning models. Efficiency of current audio quality measurement methods and results from the experiments are analysed and discussed. Difference between the types of machine learning algorithms used in the experiment are also evaluated to determine how much artificial residual noise they produce based on type of audio noise. We present a method for detecting indicators for when artificial residual noise occur, and discusses the importance of discovering and mitigating artificial residual noise in order to avoid violating the integrity of the data at hand.

The ultimate goal of this thesis is to create awareness and understanding of artificial residual noise, and providing a novel method for detecting it and explaining what can go wrong from a digital forensic perspective. By directing the spotlight on this problem with current machine learning methods, the intention is to draw more attention towards research on adapting methods for audio forensics.

Sammendrag

Innen området for digital etterforskning av lyd har det vært en viss skepsis til bruken av maskinlæringsmodeller i forbindelse med fjerning av bakgrunnsstøy. Selv om støydemping ved hjelp av maskinlæring kan gi både effektivitet og hjelpe til med å håndtere mengden data som en etterforsker vil stå overfor, har det ikke blitt tatt i bruk som en standardløsning. En av hovedårsakene til dette er sannsynlig pålitelighet. Når man arbeider innen etterforskning, er viktigheten av å opprettholde integritet og sporbarhet knyttet til de tilgjengelige dataene avgjørende. Hvis disse prinsippene brytes, kan det få hele etterforskningen til å kollapse. Et eksempel på integritetsbrudd er når bruk av maskinlæring som utfører støydemping, tilfører kunstig reststøy.

Vi vil derfor undersøke kunstig reststøy med mål om å bedre forstå hvordan det kan oppdages, og mulige implikasjoner for digital etterforskning. Gjennom eksperimentering vil vi replikere kunstig reststøy i forskjellige maskinlæringsmodeller. Effektiviteten til gjeldende målemetoder for lydkvalitet og resultater fra eksperimentene blir analysert og diskutert. Forskjeller mellom typene maskinlæringsalgoritmer som brukes i eksperimentet blir også evaluert for å evaluere hvor mye kunstig reststøy de produserer basert på type lydstøy. Vi presenterer en metode for å avdekke indikatorer for når kunstig reststøy oppstår, og diskuterer viktigheten av å oppdage og håndtere kunstig reststøy for å unngå å krenke integriteten til de aktuelle dataene.

Det endelige målet med denne oppgaven er å skape bevissthet og forståelse for kunstig reststøy, og presentere en ny metode for å oppdage det og forklare hva som kan gå galt fra et digitalt etterforskningsperspektiv. Ved å rette søkelyset på dette problemet med gjeldende maskinlæringsmetoder, er hensikten å rette mer oppmerksomhet mot forskning på tilpasningsmetoder for lydetterforskning.

Contents

Figures

Tables

Chapter 1

Introduction

The first known audio recording was conducted in 1860 on the “Phonautograph” patented by Edouard-Leon Scott, and contained a 10 seconds long performance of a folk singer [1]. By etching the captured sound waves onto a piece of blackened paper it could be played back, but the quality was very distorted and had a high amount of noisy artifacts. The simple solution that could capture and store audio started what has become very commonly applied technology today. Capability of capturing and storing audio data holding information on an event occurring at a specific moment in time has been proven to have many use cases.

With all this audio information available, audio forensics has become an important field within the forensics area of work. An investigator can potentially extract information such as indicators of events happening, the presence of objects or persons, or the environmental surroundings. Information in audio recordings can be crucial when working with cases and take hours of work depending on the complexity and amount of information available.

With an increasing amount of information to process, new tools and methods that can help increase efficiency and handle large amounts of data are becoming more important [2]. Especially considering how technological advancements today are significantly increasing the number of devices that possess the capability to obtain audio recordings. The amount of digital media that is recorded and uploaded to the internet is enormous. Also, just considering how many smartphone devices that move around in the world, picking up sound in the background while recording for social media or making a phone call, is of such a proportions that it is probably not possible to process it all. To add context to the magnitude, Gartner reported 1,433,859,400 smartphone units sold on a global scale in 2021 which is a 6% increase compared to 2020 [3]. Statista estimated 6,259 million smartphone users world wide in 2021, and estimate continous growth towards 2027 where the number could reach 7,690 million [4].

In order to better address some of these volume of data, machine learning has been adopted in many cases of data science. It has also quickly become a tool that has proven itself to be very well suited for conducting different types of audio analysis. One of these applications is noise suppression. The capability of removing unwanted background noises and enhancing speech. This is a very useful method when cleaning up heavily distorted and noisy audio samples in order to enhance the quality of a conversation for example. Then one can ask, why has it not been adopted into audio forensics? There are many reasons for this, but one of them is reliability in terms of preserved data integrity. The integrity is violated especially when artificial residual noise is introduced into the audio recording at hand, and this problem is what we are focusing on in this thesis.

1.1 Keywords

Keywords covered: Audio analysis, Audio forensics, Residual noise, Artificial noise, Machine learning

1.2 Problem description

In the field of digital forensics, it is important to present evidence “as-is” without introducing any bias or in other ways altering or manipulating the data so that it may change the outcome of an investigation. It is a principle that also applies in audio forensics, where the investigator makes use of different filters, and methods of adjustment in order to enhance and process the audio recording being analyzed. A lot of this is performed as manual work due to the need of maintaining control of the process.

Although there are available automated solutions for audio analysis, they are built around altering data in order to achieve better quality, which in relation to forensics equals to not being considered as an option for handling evidence [5]. This has through history often proven to be correct when automate systems have been set up against manual methods, working on the same case data [6, 7]. Looking at how machine learning has entered the field and quickly has been put into play in modern-day data analysis, comparison of methods will happen more often in the years to come because of how this technology lends it well to handle big complex datasets.

For machine learning to become accepted as a method of forensic analysis, what is considered as major issues with the approach needs to be solved first. In audio forensics, there exist a problem that is inherited from audio noise suppression called artificial residual noise. This is a combination of residual noise [8] and artificial noise [9], which are both products of processing audio during analysis. This phenomenon has also been proven to occur when applying machine learning to conduct the audio analysis [10, 11].

Artificial residual noise has the potential of causing a severe impact if not handled properly. Anomalies that manifest can be misinterpreted as additional audio sources in the audio recordings when analyzing [7]. There is a risk of causing confusion and misleading the forensic investigator, as they suddenly find themselves in a situation where more information has been artificially added to the data, that analysis up until that point has not previously recorded.

Even if machine learning in audio analysis has come a long way and will continue to evolve as new methods are presented, this problem has persisted through multiple iterations and variants. Understanding what artificial residual noise is, how it can be identified, if it appears under certain conditions, and how it impacts forensic investigation are the goals we seek to answer in this thesis.

We will focus on understanding the foundation of the artificial residual noise, in order to provide a platform of knowledge to continue work. There has not been a lot of concrete research on audio noise directly related to machine learning and artificial residual noise from a forensic perspective. Most of the available research material are written to present a machine learning algorithm that mentions the problem but does not fully address it [10–12].

1.3 Justification, motivation, and benefits

When conducting forensic data analysis it is important to maintain the integrity and chain of custody of the data at hand [7]. Introducing machine learning to the process puts an ethical challenge into the equation. First and foremost it is difficult to explain how an algorithm arrived at a provided conclusion [13]. Second, machine learning algorithms are programmed by humans and therefore is prone to introduce biased results [14]. These two problems violates integrity and chain of custody, and proves the importance of having the capability to uncover if one or both issues has in some way influenced the result.

The importance of maintaining integrity and chain of custody, is the reason why a forensic copy is worked on instead of the original audio recording during an investigation [7]. Processing the data will to some degree always alter information stored, but this must be a well-documented and controlled process. If audio processing results in the introduction of unwanted information that the investigator is not aware of, it can have a severe impact and will most certainly breach the integrity of the evidence[7].

In a worst-case scenario, there can be an outcome where the analyst makes a wrong decision [6, 15], which can have severe consequences for others. An example of this can be audio recording processed and presented as evidence in a criminal trial. A wrong decision will likely influence a ruling and in turn make someone become sentenced for a crime they did not commit, or have a guilty person walk free.

Therefore, gathering knowledge on artificial residual noise is important in order to understand what it is, how it might occur, and why it should be mitigated when machine learning is used for audio processing. Digital forensics is a field where the amount of information can be enormous, and the available resources and manpower to handle it limited. Machine learning has proven to be a very valuable solution for assisting in data analysis. But at the same time, it is equally important to have a thorough understanding of the limitations in capability that comes with the tool at hand. This thesis focuses on artificial residual noise which is one of the hurdles that needs to be addressed in order to make proper use of machine learning for audio denoising in the digital forensics field.

Achieving a better understanding of artificial residual noise will create a foundation for developing methods of controlling and mitigating the occurrence of this anomaly in audio recordings. We will address a problem in digital forensic investigation tied to reliability, were processing large amounts of audio information with the use of machine learning could be considered the most efficient way. Most important, it can ensure that the outcome of a forensic investigation results in the correct decision because precautions were taken that ensure the integrity of the data. This last point is most essential in this project as it is considered a fundamental problem of artificial residual noise.

1.4 Research questions

This thesis answers the following research questions:

- **Research Question 1:** How can artificial residual noise appear in audio recordings enhanced with machine learning?
- **Research Question 2:** Is there a difference in amount of artificial residual noise produced related to difference in types of noise sources? If this is the case how much of a difference is there and which types are more prone to make it manifest?
- **Research Question 3:** How large is the difference is there between different types of machine learning algorithms and occurrence of artificial residual noise?
- **Research Question 4:** Why is it important to detect and/or mitigating artificial residual noise, and what are the consequence of not handling it properly in the context of audio forensics?

1.5 Contributions

There is scarce research related to artificial residual noise available, as focus related to noise suppression often comes down to presenting a new machine learning model and how it performs compared to an existing alternatives [8, 10, 11]. The potential value of the contribution that machine learning can have in the forensic field is huge, but there are requirements that need to be met before this become possible. Part of this gap is what this thesis seeks to address, by providing context from a audio forensic perspective and conducting experiments related to identifying artificial residual noise in audio recordings.

The results will build a foundation for explaining which indicators to identify, and how to look for them when artificial residual noise manifests in processed audio recordings. Through a better understanding of this problem, it is possible to discuss and suggest solutions on how to better approach machine application. Especially from an audio forensics perspective, with the application of machine learning as a tool or supplement. The main contribution of this thesis is to provide knowledge on how to detect the presence of artificial residual noise, and understand what impact it has in a digital forensic setting. In addition we will:

- Compare different neural networks for denoising audio files, wherein focusing on residual and artificial noise.
- Identify difference in performance between the neural networks when conducting noise suppression. As an example, a Convolutional Neural Network has less complexity compared to the Recursive Network with Dynamic Attention noise, which is reflected in the results presented in chapter 5.
- Identify which kind of audio sources that causes most artificial residual noise after being processed. Out of the 10 different audio sources there are variance in complexity. The air conditioner is an almost constant sound while street music has a lot of variation in both frequencies and amplitude.
- Identify if conventional audio quality measurement methods detects indicators of artificial residual noise. Perceptual evaluation of speech quality, Segmental SNR, and Short-time objective intelligibility is commonly used for quality measurement of audio, but was not created to handle artificial residual noise.
- Present a method for identifying indicators of artificial residual noise in denoised audio recordings, by analyzing the short-time Fourier transform array.

1.6 Thesis outline

The rest of the thesis consists of the following chapters:

- **Background:** In this chapter the reader is introduced briefly to background of audio forensics and relevant knowledge on artificial and residual noise. The goal is to provide the reader with an understanding of the fundamentals for the rest of this thesis.
- **Theory:** The chapter on theory aims to giver a deeper and more detailed understanding of theory behind audio analysis and machine learning methods. The goal is to provide the reader with knowledge on why and how audio and machine learning is applied to handle noise suppression.
- **Methodology:** This chapter contain the methods used in this thesis in order to answer the research questions and conduct the experiments.
- **Results:** This chapter presents the result and output gathered during the experimentation. Information in this chapter makes up the foundation for the next chapter.
- **Discussion:** Analysis of the experiments and results acquired in the previous two chapters will be covered in this chapter. The analysis is then mapped to the research questions by illustrating how the experiments can be used to answer them. It also covers some limitations that were encountered during the thesis.

- **Conclusion:** This chapter presents a conclusion based on the content discussed in the discussion chapter.
- **Future work:** Some potential areas of research, improvements, and future work is presented in this chapter.

Chapter 2

Background

This chapter contains an introduction to the concept of audio forensics and artificial residual noise. The intention is to provide the reader with enough information about these two topics to understand the rest of this thesis.

2.1 Introduction to audio forensics

First of all, it is important to understand what audio forensics refers to. In this thesis it goes by the definition of Robert C. Maher:

"Audio forensics refers to the acquisition, analysis, and evaluation of audio recordings that may ultimately be presented as admissible evidence in a court of law or some other official venue." [7, p.84]

When obtaining forensic audio evidence, it is most often a product of an ongoing investigation. This can be either in the context of a civil or law enforcement case, but also in official inquiry might there be evidence in form of audio recordings. Audio forensics also comes with some concerns which are [7]:

- Establish authenticity of the audio evidence.
- Perform enhancement of audio recordings in order to improve speech intelligibility and audibility of low level sounds.
- Interpreting and documenting sonic evidence. Examples of this are identifying sources of speech, transcription of dialog, and reconstruction of crime scenes or scenes of accidents and timelines.

When performing these steps, it is essential to remember that the methods used must previously have been proven to be unbiased, be statistically reliable, non-destructive, and widely accepted by experts within the field of audio forensics if it is to be presented in a court of law [7]. This is essential when moving into the application of new methods like machine learning algorithms for processing audio.

It can be understandably tempting to make use of machine learning algorithms for speech enhancement in order to clean up several hours of seized audio recordings. But as long as this method is not previously accepted it will potentially damage the case severely by being denied as evidence based on the lack of admissibility. Besides, there might also be concerns that through processing the evidence presented might have been altered and as such become biased[7]. An example of this could be that during noise removal, a statement spoken during the audio recording can be interpreted differently. Someone saying what sounded like “I didn’t do it” could suddenly be interpreted as “I did do it”, which would be complete opposites of each other and of course have a large impact.

Because of these kinds of strict conduct and regulations that need to be followed, audio forensics is a field that requires a lot more than what is considered standards within an academic setting of audio processing. The demand for maintaining a strict chain of custody and authenticity of the audio recording is a good example of how different audio forensics is compared to the academic field [7]. To better understand why it has come to this strict regime, it is useful to take a look at some historical defining events that have dictated how audio forensics are done today.

2.1.1 Audio forensics history

A major paradigm shift in audio forensics came in 1974 as a consequence of the investigation of a White House conversation between President Richard M. Nixon and Chief of Staff H. R. Haldeman [7, 16]. The recording itself was made in the Executive Office Building in 1972. While examining the audio recording, investigators uncovered an unexplained section that ran for 18,5 minutes. During that timespan only the presence of a buzz sound could be heard, and no audible speech.

This lead to appointment of an advisory panel of technical experts to study the audio recordings. These experts represented engineers and acoustics communities such as AT&T Bell Laboratories, Magnetic Reference Laboratory, University of Utah and the Federal Scientific Corp [16]. They performed a series of analyses including observations the audio signals themselves as well as studies of the magnetic development of domain patterns and head signatures. The conclusion of the investigation was the the gap was caused by multiple overlapping erasures of the original recording, using a different type of recording device than the one that originally made the audio recording [7, 16]. The conclusion itself was made based mainly on the characteristics of the magnetic signatures on the original tape, caused by starting and stopping the recording.

Their approach was quickly adopted as the standard for audio forensics when checking authenticity of forensic audio recordings. A 5 step approach was established as guidelines for audio forensics [7]:

1. Physically observe the entire length of the tape.
2. Document the total length of the mechanical integrity of the tape, reel, and housing.
3. Verify that the recording is continuous with no unexplained stop/start sequences or erasures.
4. Preform critical listening of the entire tape.

5. Use nondestructive signal processing as needed for intelligibility enhancement.

Another defining event occurred some years prior to the Watergate case [16], the assassination of President John F. Kennedy. As part of this investigation, audio forensics became a central part of it to try to determine the placement of the shooter [15]. One of the most important pieces of evidence was an audio recording from one of the officers presumably riding a motorcycle in the area of interest. Normally these microphones were voice-activated, but this officer's microphone was stuck in active mode at what they estimated to be the time span of shots being fired.

The conclusion of the investigation has been changed a couple of times regarding the audio recordings. At first, analysts came under the assumption that a total of four indicators pointed toward shots being fired, where the third one was placed at the grassy knoll area of Dealey Plaza [15]. This led to the belief that there were two shooters.

Later more thorough investigation rectified this as forensic analysis involved attempts to reconstruct the soundscape at the time of the assassination. The reason for this was based on the original recording from the officers' microphone being subject to heavy distortion and noise. In fact, the report published by the Federal Bureau of Investigation (FBI) in 1980 dismissed the audio recording for the reason being *"...did not scientifically prove that the Dictabelt recording channel 1... contains the sound of gunshots or any other sounds originating in Dealey Plaza..."* [15] This conclusion was later confirmed by the Committee on Ballistic Acoustics in 1981 [15].

The committee commended the novel approach of analysing audio recording by trying to estimate shooter location based on how sound patterns were perceived based on the environment, but at the same time underlined the importance of questioning the conclusion based on the quality of the audio recording and identification of the audio source is based on subjective opinions, and not scientifically proven. The case is an example of the importance of having scientifically objective means to back up a hypothesis.

One last example of forensic audio recording analysis being conducted in a more modern setting happened in November 2002 [6]. An audio recording of a phone call, from what was stated to be former al Qaeda leader Osama bin Laden, was broadcasted by the independent Arabic station al Jazeera. This triggered an immediate response from security agencies starting to analyze the recording in order to verify the identity of the speaker. Even if it is not known which tools the CIA and National Security Agency have at their disposal, the conclusion was that this was indeed bin Laden speaking.

At the same time, Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP) decided to conduct an experiment testing their, at the time, state-of-the-art speaker authentication system. As opposed to the conclusion of the United States agencies, their system concluded that this was an impostor with a 55 to 60 percent certainty [6].

Benchmarking the biometric system of IDIAP had demonstrated precision in the performance of 97 percent [6]. Tests had been made with 15 authentic recordings and 16 imitations of bin Laden. The system had correctly identified all 16 imitations, and wrongly classified only 1 of the authentic audio samples. In other words, these numbers added some credibility to the system. This was not well received by some as it implied that traditional methods for conducting forensic audio analysis were out of date.

Based on what happened several years later when a military operation in 2011 resulted in the elimination of bin Laden, it adds credibility to the established approaches of forensic audio analysis. As was stated by the director of IDIAP, there are certain aspects of how speech has characteristics that algorithms do not necessarily have the capabilities to catch. The director of IDIAP also stated that systems such as what they are developing should be used as a supplement, and not be looked upon as a replacement[6]. This underlines the importance of understanding the limitations that comes with automated audio processing systems built for analysis.

What is important to learn from these examples is how historical events have shown that audio forensics is an essential area of research. Some form of audio will almost always be present where humans are involved, and as such creates potential evidence that can be used in further explanation of what has happened. Another important point is to understand how there are limitations to the technology at hand. As demonstrated by both the Kennedy case and the bin Laden event, information can be misinterpreted and lead to wrong assumptions and conclusions.

But history also shows us how important it is to question and not adopt methods that have not been established as a standard. Making use of new technology like machine learning should be used as a supplement, to begin with, and continue to be used as such until it is proven to be reliable. Quick and Choo pointed out that although machine learning has many advantages, the lack of control the can lead to information not being included, which underlines the importance of human supervision [17].

2.1.2 Audio forensics and the law

Another very important part of audio forensics is to understand some of the legal aspect that needs to be considered when handling and processing audio recordings. As will be presented in this section there are differences in how the legal framework is built which also has an impact for practitioners within the forensic field.

In the United States, foundations for how audio recordings were to be treated as evidence was laid down in 1958 [7]. The evidence in question was a tape-recorded conversation that involved the defendant, in the case of the United States versus McKeever. The tape itself was never played in court, as a transcript of it was delivered instead, but the judge had to establish some requirements, that are still in use today with some variations. These 7 requirements revolves around authenticity of audio recordings [7]:

1. That the recording device was capable of taking the conversation now offered as evidence.
2. That the operator of the device was competent to operate the device.
3. That the recording is authentic and correct.
4. That changes, additions, or deletions have not been made in the recording.
5. That the recording has been preserved in a manner that is shown to the court.
6. That the speaker are identified.
7. That the conversation elicited was made voluntarily and in good faith, without any kind of inducement.

Considered most important today are points 3 to 7 [7], where 3 and 4 can refer to a forensic audio expert being involved. 5 sets the requirement of chain of custody. Requirement 6 addresses the importance of being able to clearly identify the participants in the audio recording. This can be either by their voice or through participants addressing them directly by name for instance. The 7. requirement implies that the conversation should not be rehearsed or prepared, but be spontaneous.

But even when following these guidelines, there are concerns that can lead to a court rejecting an audio recording as evidence [7]. First of, the participants in the audio recording has not sworn to tell the truth, like they do in court. One might consider this a formality, but it is still reason enough for dismissing the evidence. Another issue is cross-examination of witnesses, that may or may not be available for this process.

In Norway there are some differences for how audio recordings are treated legally. Most important is the principle of freedom of evidence. This means that a person can bring any legally acquired evidence in front of a court, and then it is up to the court to decide whether the evidence is allowed or not, as outlined in the Norwegian Criminal Procedure Act §294 [18].

Audio recordings are considered to capture an overflow of information, such as a persons way of speaking and potentially their mood [19]. It is therefore considered as intrusive when it come to personal information and must adhere to the Personal Data Act which draws many similarities from the general data protection regulation (GDPR) [20]. There are other ways that are considered less intrusive, like for instance taking notes based on what is said in the recording, which would be aligned with the Personal Data Act principle of data minimisation.

Anyone that wants to record, store or in other ways make use of audio recordings containing information on other persons must therefore adhere to Norwegian law. In short this means that the following applies [19]:

- A fundamental principle is that in order to make an audio recording there need to be a legal basis for it.
- Those who are subject of being part of an audio recording must be notified

before recording begins.

- There must be a clear purpose, and the audio recording must be limited to what is strictly necessary.
- An audio recording can not be stored after it has served its purpose.
- Audio recordings can only take place if information security is considered to be properly applied to protect it.

There are situations where the Personal Data Act and these principles does not apply [19]. One example of this is when the police are using audio recordings as part of their ongoing investigation of criminal cases. In this case they adhere to The Police Databases Act and the The Criminal Procedure Act. Journalism are also considered partially excepted from this when operating within their line of work.

Understanding and being acquainted with both local and foreign legislation as a forensic investigator is essential as it will dictate how an investigation would need to be planned and conducted. Because of how a lot of infrastructures today is spread across geographical borders, an investigation is not necessarily bound to only one jurisdiction. As the examples in this section demonstrate there are differences that will have an influence one way or the other.

2.2 Previous work

When it comes to artificial residual noise there is little available research directly aimed at the field of digital forensic. The same applies to academic research as well, and most often what is presented is in context of a new machine learning algorithm and mainly focused on improving quality of speech [8–11, 21]. This indirectly touches upon the field of residual noise as it can be tied to suppressing natural background noise.

There is, to our knowledge, no information or research of how these machine learning algorithms specifically addresses the artificial residual noise problem. One of the reasons for this can be related to how they focus on achieving best possible result in quality measurement scores like perceptual evaluation of speech quality score, short-time objective intelligibility score, or segmental signal to noise ratio score [8]. But in none of the cases are artificial residual noise directly addressed as a problem.

Considering how machine learning has been available for some time and the amount of research that is being put into audio enhancement area of the technology, it is a bit strange that there has not been more focus on this from an audio forensic perspective. Especially when evaluating the potential beneficial gain it can yield in terms of processing speed and volume of information handling. The amount of potential audio data that a forensic investigator will face in the future will continue to grow when considering a constant introduction of new social media and sharing solutions, and how the number of devices like smartphones continue to increase [3]. In order to keep up with all of these sources of audio information, automation in form of machine learning has a big potential.

Chapter 3

Theory

This chapter presents additional theoretical background information on the purpose and functionalities of audio analysis, machine learning algorithm and quality measurement approaches made use of in this thesis. The aim is to grant more theoretical knowledge on each respective method.

3.1 Introduction to artificial residual noise

When working with audio signals there will always be some form of noise present. Noise is added to an audio signal as soon as the signal moves from the source holding it to the receiver. During the signal transmission, elements in the environment will influence the signal and degrade it by adding some static noise [22]. It is called additive noise, and applies even within a device as a signal has to move through components before it can be broadcasted from a speaker for instance. Additive noise is not always possible for a human to hear, as there are limitations to the sound that we are able to perceive. But measuring a raw signal before and after signal transmission will show that some noise has been added [22].

3.1.1 Artificial noise

Artificial noise is created in an artificial way by human devices. It can manifest in various ways being wave or vibration, audible, electromagnetic, or other signals to mention some examples. In some cases it can be intentional in order to test the audio cancellation of microphones in a laboratory, or it can be unintentional if a system or device malfunctions. Even if artificial noise has some use cases it is often not desired.

In this thesis, artificial noise is an area of interest because machine learning algorithms that perform audio processing might be prone to introduce artificial noise. The hypothesis of this is tied partially to the concept of additive noise. Similar to components in an audio playback device, a machine learning model is composed of components, that handle different operations during audio processing in this case. As pointed out earlier, processing an audio signal will alter it and will very likely degrade it to some degree. In previous research it has been proven that under certain conditions when attempting to control or remove noise, it did add artificial noise to the audio recording instead, which is not wanted behavior [9].

While working with audio the term "phase" is guaranteed to be encountered. When we talk about audio phase we are referring to a specific point within a audio wave cycle. As audio waves travels, the easiest way of visualizing it is through sine waves because of how they vary in amplitude over time. Audio phase contains some interesting information that can be applied when conducting audio analysis. As illustrated in Figure 3.1 a phase is represented by the green area, and contain information about a cycle of the blue signal. A phase spectrum, represented by the pink area, holds information about multiple phases for both blue and red signal in this case. A phase shift is is illustrated in Figure 3.1 as the difference between the blue and the red signal reaching the same amplitude at different times.

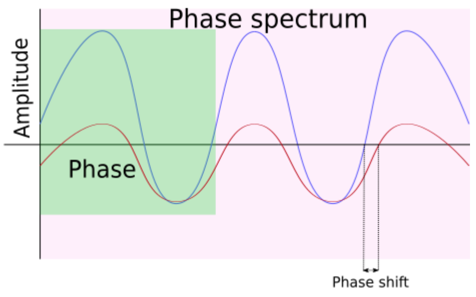

Figure 3.1: Example of audio phase terminology

Not including phase information in training of machine learning can cause the introduction of artificial noise. There were indications of this during a study made by Paliwal, Wójcicki and Shannon [9]. To elaborate on what their study uncovered, the way that phase spectrum display phase shifts between signals with different frequencies at a given time, allows for the estimation of noise versus clean signal phase. By applying this knowledge to adjust for changes and variations, yielded promising results especially when a clean speech spectrum is known beforehand [9]. Knowledge of the clean speech spectrum before processing also helps with the challenge of working with audio processing close to real-time, which can be very difficult if complex computation is needed.

Another technique within audio denoising, is application of ratio masks in an attempt to isolate clean speech [11]. The approach is often performed in combination with machine learning in order to try to estimate how the clean speech signal will appear [10, 11]. Based on these estimates, the machine learning algorithm makes an attempt at predicting how to best mask the noisy signal. Multiple methods exists for how to create a ratio mask, but the common challenge they all face is how to achieve perfect precision [10]. This is especially hard when the audio source or noise has a lot of variance in behaviour and amplitude. As a consequence, there can be noise bleeding through where the mask does not provide the proper coverage. In these cases, the machine learning algorithm might not handle noise removal properly [10, 11]. Additive noise is manifesting instead, resulting in artificial noise, as it processes parts of the noise into what is considered clean speech, which is not wanted behavior.

Variations in the amplitude of clean speech are another factor that can result in the creation of artificial noise [21]. The reason for this resembles that of the mask ratio, in that the human voice can be difficult to predict. If the amplitude of the audio source presenting clean speech goes soft, a machine learning algorithm might have a challenge distinguishing it from the background noise. It might lead to a situation where noise is interpreted as speech and vice versa, especially if the source of the noise has similarities with a human voice in terms of frequencies and variance in pitch and amplitude.

3.1.2 Residual noise

Within the area of audio signal theory, residual noise can be explained as noise that is still present after noise suppression has been conducted at a given position and situation [8]. It is not necessarily unwanted in all settings, as it can be used as a tool to block out other sounds. However, in the field of audio forensics where the aim is to achieve as high precision in the result as possible, it is not desired in most cases.

As this is a common concept there has been research made on residual noise and how to mitigate it, but there is no perfect way of handling it as it can be a complex problem. Variance in motion, amplitude, and frequency are all factors that contribute and makes it difficult to control. It is in all essence audio signal noise and has the same attributes.

Now that the basics of artificial and residual noise have been presented, an understanding of the building blocks of what makes up artificial residual noise should be easier to understand. When using machine learning algorithms for noise suppression, the process can add and leave the noise behind in the processed audio recording that was not present in the original recording. Considering how strict requirements are for evidence in the field of forensics, as presented earlier in this chapter, we understand that this is not accepted by a method used in an investigation.

3.2 Audio analysis

When working with machine learning algorithms it is important to understand how to best represent the data to the machine learning model. In our case, data consists of interpreting and handling audio. A human's perception through hearing that has been trained for multiple years can understand and single out individual sounds even if they are mixed together in an audio clip. When presenting the same challenge to a machine learning algorithm, it needs a lot of time and a lot of data in order to understand how it is supposed to interpret the information.

To further help the machine learning algorithm, it is useful to select features from the data that are distinct and as such makes it easier to distinguish sets of data from each other when compared. In order to further increase the performance of a machine learning algorithm, especially when working with large datasets, is selecting only the features that contribute most to the uniqueness of the data. By taking this approach there will be less data (features) to look at while they still make it easier to conduct data classification.

As the data, in this case, consists of audio, it needs to be converted in such a way that features can be extracted. Audio is a field where there is a lot of existing research, and looking within the scope of audio features for data analysis there are multiple possibilities.

One of the easiest features to understand is the chroma feature [23]. It is typically built up of twelve elements indicating how much energy of each respective pitch class is present in a signal. The different pitch classes is what we in music describe as “C”, “C#”, “D”, “D#”, “E”, and so on. These features are often used when comparing or classifying music pieces.

The Mel-frequency Cepstral Coefficients (MFCCs) [24] can be used to extract features when an audio signal is measured in the frequency-amplitude plane. By looking at the overall shape of the sine wave curve (spectral envelope) a signal makes in time one can extract anywhere from ten to twenty features from it based on its shape. An example of the application of the MFCCs is for modeling the characteristics of the human voice.

Another feature that can be used is the Zero-Crossing rate [25]. Considered a simple way to calculate the smoothness of a signal by counting the number of times it oscillates within a set segment. An example is how a human voice can oscillate slowly. A 100 Hz signal will cross zero 100 times per second. At the same time, it can oscillate fast as an unvoiced fricative can have 3000 zero crossings per second.

An efficient way of extracting features from audio signals is through Fourier analysis, because of how the method was developed to define periodic waveforms. Considering that audio consists of periodic sine waves as illustrated in Figure 3.1, The principle of Fourier analysis is to convert a signal from its original domain like time or space to a representation in the frequency domain and vice versa. There are some variants of Fourier transform:

- The Discrete Fourier transform (DFT) [26] is a method for converting a sequence of complex numbers to a new sequence of complex numbers. This approach is applied in order to find the coefficients of an approximation of the signal, so that different sound files can be compared by these coefficients.
- Another variant is the Fast Fourier transform (FFT) [26], which is considered to be a variant of the Discrete Fourier transform algorithm. It reduces the number of computations needed, by making use of a shorter sequence if the Discrete Fourier Transform equals the power of 2.
- Then there is the variant that was used during this project. The Short-time

Fourier transform (STFT) [26] converts signals to enable the amplitude of the given frequency at a given time to be identified. Using STFT we can determine the amplitude of various frequencies playing at a given time of an audio signal, and provides a very good foundation for data to be provided for the machine learning algorithms.

STFT is converted back to an audio signal by providing a sample rate. There exist multiple tools that can process the data into common formats like wav, mp3, flac, and so on. The STFT also lends itself well to being plotted as spectrograms, as it holds 3 parameters; time, frequency, and amplitude.

The Spectral Centroid indicates where the ”center of mass” for a sound is located and can be used as a value for measuring the robustness of an audio signal, like a weighted mean value [27].

Spectral Rolloff is another feature and is used as a way to measure the shape of the signal [28]. It represents the frequency at which high frequencies break and starts to decline towards 0.

The last common way to measure audio features is the Spectral Bandwidth. It is defined as the portion of a signal spectrum in the frequency domain which contains most of the energy of the signal.

3.3 Machine learning algorithms

3.3.1 Artificial Neural Networks

Conventional Artificial Neural Networks are known as feed-forward networks, or a multilayer perceptron, meaning each of the layers are fully connected to the subsequent one. This setup seeks to imitate how the neurons of the human brain are laid out and connected. A simplified presentation often used would be as demonstrated by Figure 3.2

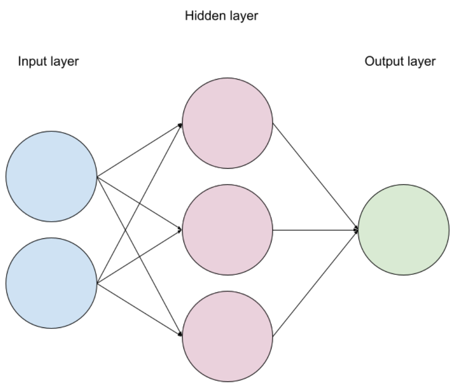

Figure 3.2: Example architecture Artificial Neural Network

In its simplest form, the neural network consists of an input layer where features of the data model are fed into the network. The following layer is known as a hidden layer, meaning that the model is not directly aware of its presence. It has the purpose of altering the values by applying additional weights as they are passed forward toward the output layer. This way the model has to repeatedly train itself to increase its precision, as it does not fully control the outcome [29].

As stated this is a neural network in one of its simplest forms. This project made use of several different types of neural networks with different degrees of complexity in order to compare how they handle the task of conducting audio noise removal, and the potential of generating and adding artificial noise to the result in that same process. Before the architecture of these neural networks are covered, there are some concepts that need to be explained and taken into account, as they can have a big influence on neural network performance.

One of two learning paradigms that can be applied by an Artificial Neural Network is supervised learning [29]. In this case, the input fed to the network is pre-labeled, meaning that for each input there exists a designated target output. The approach seeks to increase the precision of the model by constantly comparing and adjusting weights applied to the calculations, in order to minimize the classification error as much as possible.

The second learning paradigm is unsupervised learning [29]. As opposed to supervised learning, there are no labels present. Instead, the network tries to increase or decrease a cost function in order to achieve better precision. In this case, the model itself is given more freedom as the model learns directly from the data without having any form of previous knowledge. As this is a rather time-consuming process and demands both experience and good knowledge within the field of machine learning, supervised learning will be used when working with data as part of this thesis. A set of training and validation data will be used for pretraining the different models.

Another important factor to control is the concept of overfitting [30]. When a model learns the details of the training data too well, it will in fact have a negative impact on performance when new data is introduced. Because the model has learned all concepts of the initial data, it is not able to interpret the new dataset because it lacks the ability to generalize. It is therefore important to include methods and/or techniques for limiting and constraining how well the model learns the data presented. In the case of this thesis, an exhaustion mechanism is put in place. The purpose of this mechanism is to stop the training after a set amount of intervals where it no longer detects any increase in precision. This is a common and efficient way to avoid overfitting the model.

In order to control how a model learns, some form of adjustment needs to be applied to correct or align the output between layers. There are multiple approaches to do this and a sigmoid [31] or tanh [31] activation function has been a common way of handling nonlinearity. However, as the depth of a neural network increases the gradient signal of sigmoid and tanh functions will vanish.

To address the vanishing gradient, using Rectified Linear Unit (ReLU) [31] is recommended. It sustains a constant gradient, it will also provide a more sparse representation, and can reach a complete zero, as opposed to a sigmoid, and tanh that can only provide non-zero results. it leads to a more precise output compared to the older alternatives. ReLU will therefore be utilized when training models as part of this thesis unless other methods are recommended in the literature.

3.3.2 Convolutional Neural Network

The Convolution Neural Network has become a well-established alternative when making use of neural networks for applications related to pattern recognition [32]. One of the main advantages it has over conventional artificial neural networks is the reduced number of parameters needed to conduct classification. Because of this, it allows researchers to make use of larger and more complex datasets for solving problems that were not possible with previous algorithms.

Another advantage of this algorithm is spatial independence when mapping features. A good example of this is how an image is analyzed. If the aim of an application is to identify cars in images, the spatial independence makes it irrelevant where the car is located in the image. The algorithm is simply looking for a pattern that corresponds to the car itself [32]. This feature is why the method also suits well for audio analysis as in this project. The main aim is to identify patterns in the audio clip that can be mapped and processed for denoising.

Derived from this ability to handle details, convolutional neural networks are also able to map abstract features [32]. Input can propagate as the model moves towards deeper layers, granting more granular data. In relation to audio analysis, this means that the presence of an audio source versus silent frames can be detected in the first layer, differences in the beat of a rythm on the second finer layer, and tonal differences in the third more detailed layer. This is made possible by how the algorithm processes data on the different layers.

A layer in the convolutional neural network defines a subset of data [33], also known as a filter or kernels, of the complete dataset that it examines. The output is often referred to as a feature map. As demonstrated in Figure 3.3 a subset of the full matrix is extracted into a pooled vector. This is then transformed into a filter/ kernel where a scalar product is calculated for each value it holds. The values provided from this operation are learned by the network, so that the next time this feature is observed in a spatial position, the kernel will activate.

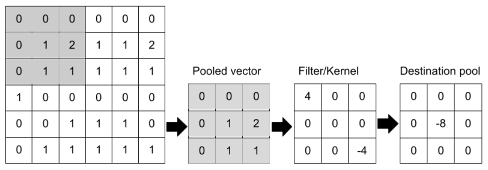

Figure 3.3: Example Convolution layer

One of the main advantages of this approach is granted the ability to make use of a very high number of features compared to a conventional Artificial Neural Network. As an example, if the classification of images was the task the most suitable features would be the individual pixels. If this data were to be fed into an Artificial Neural Network, there would be a need for one input perceptron per pixel. Considering how an image consists of x times y number of pixels, the number requires such a high number of perceptrons that the entire network would not operate in a proper way.

Because of how Convolutional Neural Networks naturally break down chunks of data into segments [32, 33], it is much more suitable for handling such a task. This also applies when handling multidimensional data like images consisting of an x and y parameter along with color values for each pixel at the x and y location. Another example that applies to this project is audio analysis, where an audio signal consists of data in the dimensions of different frequencies over time and the amplitude of the audio at a given time in a given frequency.

There are some drawbacks to this algorithm that needs to be considered. One is how it is very common to make use of pooling layers in order to reduce the dimension of feature maps in the neural network. By doing so the number of parameters to learn and computational power demanded is also reduced. From a performance perspective, this is a wanted feature, but one has to take into account how removing parameters will impact the uniqueness and potentially valuable information.

To give an example we can consider a human face. If we break it down there are features that will help us identify a human face such as the presence of two eyes, a nose, a mouth, and an oval shape of the face itself. We, as humans, also know that in order to classify this as a human face, these components need to be placed in a certain way. An algorithm that makes use of pooling and discarding information might not take any heed to the placement of the facial features. It simply identifies them through pattern recognition and concludes that they are present. Therefore it must be a human face.

Another challenge is how backpropagation [30] is used as a learning method in a convolutional neural network. The size of the dataset needed to efficiently benefit from backpropagation to find gradients is a factor that needs to be accounted for. Also, backpropagation in itself is very much dependent on the input data and can be sensitive to noise. Because of this, it raises the need to do some preprocessing on the data before it is handled by the machine learning algorithm.

As the convolutional neural network grows deeper [32], it is also important to be aware of problems tied to vanishing gradients. To mitigate this, Batch Normalization is an important method where the input is normalized to be kept within an acceptable range so that the gradient does not become too small. But normalization in itself is not enough to address vanishing gradients. Therefore an activation method such as the Rectified Linear Unit (ReLU) [31] is used in combination with Batch Normalization. The benefit is that ReLU does not saturate and has a constant and bigger gradient compared to legacy activation methods such as sigmoid

[31] and tanh [31] functions. In practice, ReLU treats a gradient that is greater than 0 as 1, and otherwise, it is zero. This binary approach makes it so that the result is either true or false instead of some value. Although this is good, there might be situations where the input is negative. In these cases, ReLU will set the value to 0, but the side effect is that the network cannot make backpropagation because of this. The phenomenon is known as the “dying ReLU problem”.

In order to mitigate this problem another variant of ReLU, called Leaky ReLU [31] has been presented, capable of handling negative values. Although it solves the problem with negative values, it adds additional performance requirements as computational work needs to be performed when calculating negative values. Also, when learning rates become very small, dead neurons will remain dead and will not participate in training.

3.3.3 Convolutional Recurrent Neural Network

Drawing from the same benefits as the convolutional neural network makes up the foundation of the more advanced convolutional recurrent neural network [34]. In addition, it makes use of capability from another type of machine learning algorithm known as the recurrent neural network. The recurrent neural network is an established method that was developed during the 1990s. It is an algorithm designed to learn patterns in a sequential manner [34]. The main feature of a recurrent neural network is the ability to handle feedback in form of possessing a short-term memory. This makes it so that recurrent neural networks are designed with the vanishing gradient problem in mind, able to keep track of arbitrary long-term dependencies.

To mitigate the long-term gradients from either vanishing or exploding into infinite value, the algorithm makes use of feedback connections as illustrated in Figure 3.4. There are some variants of these, the Long short-term memory (LSTM) being one of the more known ones [35]. This connection is composed of three components. Input is taken by the input gate, which decides if the information is to be stored in the long-term memory or not. It operates with the current information and the information from the last step. The input gate has two options. It can either forward information as output, feeding it further into the network and at the same time update the short-term memory with the new value. The second option is the forget gate, which results in discarding the information as it is considered to not contribute to improvement. The main goal of this method is to constantly drive improvement forward, and leave out information that does not serve this purpose. convolutional recurrent neural network used in the experiment.

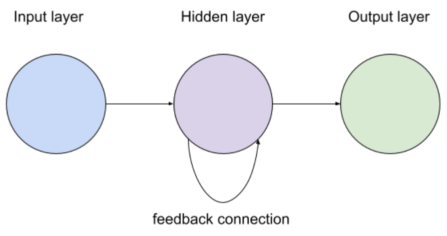

Figure 3.4: Example of Recurrent Neural Network feedback connection

An alternative to LSTM is the gated recurrent unit (GRU) [35]. Compared to LSTM, it has only two gates and as such is simpler. The update gate decides how much information is needed to pass to the next state. If needed, it can copy all of the past information in order to eliminate the risk of vanishing gradient. The second component of GRU is the reset gate. As opposed to update gate, it decides how much information to discard, meaning whether the previous state was important or not.

Comparing GRU versus LSTM [35] it is important to take note of the differences. As the gated recurrent unit is simpler in design and operation than long short-term memory, it is also faster when it comes to performance. However, it does not contain an internal memory. This means that long short-term memory handles the data internally making decisions based on the information it possesses itself. A Gated recurrent unit, on the other hand, applies the reset gate directly to a previous hidden state.

Deciding on which option to use will depend on the size of data to be handled. The short gated recurrent unit is better for smaller datasets, while a larger dataset would benefit more from long short-term memory, and the internal memory capacity that comes with it. Considering how this project was operating on a relatively small dataset, the gated recurrent unit was applied to the convolutional recurrent neural network used in the experiment.

Even if the recurrent neural network handles sequential data [34], varying outputs, and possesses the ability to memorize historical information, it comes with some shortcomings that one needs to be aware of. First of all the added step of handling historical information can impact the performance making it slow. Second, the network does not have the capability of taking future inputs into account when making decisions. Third, even if methods like long short-term memory and gated recurrent unit are tools for addressing the vanishing gradients, it does not completely mitigate the problem. When using gradients for updating the weights of a network move close to zero, it can prevent the network from learning new weights. This problem is more likely to occur the deeper the network becomes. It is therefore important to take this into account when building the network model [34].

3.3.4 Densely Connected Convolutional Networks

Just like the convolutional recurrent neural network, a densely connected convolutional network draws on much of the same capability as a conventional convolutional neural network. What sets this variant apart is how layers are connected in order to handle the flow of information [36]. Each layer, including the input layer, is connected and shares its information with all subsequent layers through the network until a transition layer consolidates and prepares the final result which is forwarded to the output layer as illustrated in Figure 3.5.

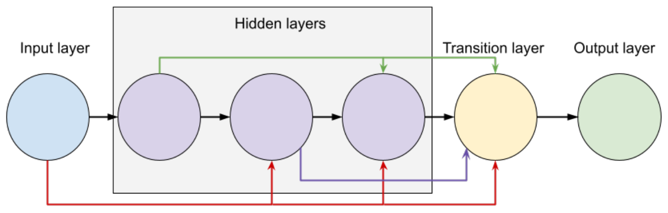

Figure 3.5: Example of Densely connected layers

As described by Huang et al.[36] the concept is to ensure a maximum flow of information when feeding forward in the network and preserving data from all previous layers. Each layer consists of a convolutional block so that the data forwarded will be much the same as in a convolutional neural network.

Another difference in the densely connected convolutional networks, is the number of features needed. Because of how information is shared between layers, a lot fewer are needed as relearning already processed features would be considered a redundant operation. This makes the model a bit more performant and faster to train compared to the conventional variant of the algorithm.

3.3.5 Recursive Network with Dynamic Attention

This algorithm was developed especially for handling speech processing and background noise [12]. It highlights how other algorithms often become restricted because of how a number of parameters are kept to a minimum in order to increase performance, which in turn restricts the dept of the network. Secondly, it also points to the vanishing gradient problem which is an ever-present factor tied to the depth of the neural network. Because of this, the recursive network with dynamic attention approaches a progressive way of learning, subdividing the information mapping process into multiple smaller stages. The network is then reused for training these smaller stages by linking a memory mechanism. It allows for deeper network architecture without introducing extra parameters.

Another core aspect of this model is how it seeks to emulate human behavior when it comes to the capability of focusing attention on audio in a noisy environment [12]. The basic principle is that when focusing on a specific audio source, more neuron capacity is applied in order to enhance attention toward the target source.

The result of this concept is two networks running in parallel. The main network operates as the noise reduction module, and the sub-network cooperates with functions as an attention generator [12]. The workflow of this collaboration begins at each intermediate stage where the noisy feature and the estimation from the previous stage are combined. The result serves as the input for the next stage. The attention generator sub-network creates an attention set which is then applied to the noise removal network as pointwise convolution. It makes use of the Attention generator function as a perceptron module that can adjust weight distribution in a flexible way. Figure 3.6 below is an illustration of one stage. Several of these stages are linked together into a sequence when training.

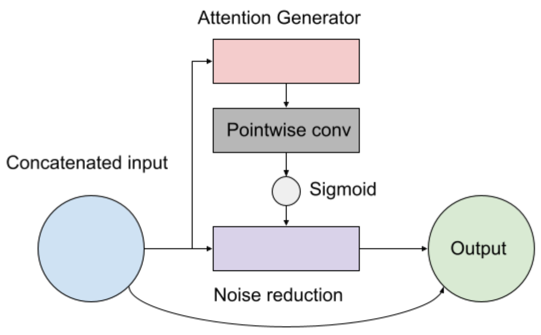

Figure 3.6: Example of one stage in processing data

To better understand the purpose of the attention generator it can be broken down into smaller components. There can be some variations to the usage of building blocks, but in this project, it makes use of a variant that consists of an encoder and a decoder.

For audio analysis, the encoder is set up with successive two-dimensional convolutional layers [12, 32]. Like previously explained algorithms it utilizes batch normalization to normalize the output from the convolutional layer. Then instead of Rectified Linear Units as an activation method, it makes use of Exponential Linear Unit (ELU). This method is very similar to the rectified version except but has a smoother slope in transition compared to the sharp approach that ReLU has. The advantage is that ELU this way has higher precision.

As opposed to the encoder, the decoder is a mirrored version. All the convolutional layers are replaced by deconvolution layers to enlarge the mapping size back, step by step until it reaches its original state. When reverting back it is recommended to compensate for information loss that can occur during the encoding process, by having a mechanism in place like a skip connection. The processed result from the decoder is then fed back into the noise reduction part of the network.

The noise reduction part of the network consists of the convolution and deconvolution blocks as explained when looking at the encoders and decoders. Between the encoders and decoders, there exists a section containing a sequence of multiple Gated Linear Units (GLU) [12].

A Gated Linear Unit [12] is derived from the long short-term memory [35] method but is different in the way that it is built to be applied to convolutions and linear layers. The purpose of GLU in this neural network is to look at contextual correlation efficiency. Multiple GLU segments operate in a sequence and the result is concatenated before being passed on to the decoder section of the neural network in order to extract features for further processing.

3.3.6 Gated Residual Networks With Dilated Convolutions

This is also a model building on the basis of convolutional neural networks. In order to increase the accuracy of a model, there are multiple options available. Previously the most common approaches were to either increase the depth or kernel size of a model [32]. However, a known drawback of increased depth comes in the form of decreased computational efficiency, and also results in vanishing gradients. Another known issue related to the conventional way is related to increasing the kernel size. It also comes with a potentially high cost related to increased computation and training time for the model.

Dilated convolutions are an alternative method to increase the depth of the neural network [37, 38]. The idea behind this technique is to increase the number of receptive fields exponentially, as compared to a linear increase in conventional convolution layers. The benefit of the method is increased accuracy through a more granular network. Dilation makes use of space between the convolutions by widening the kernel. Figure 3.7 below illustrates dilated convolution.

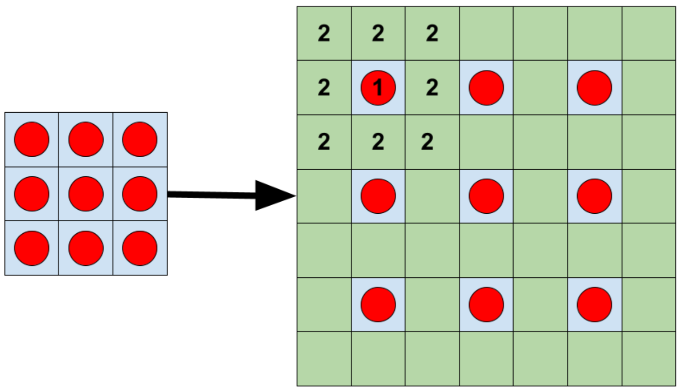

Figure 3.7: Example of 2 dilated convolution

As demonstrated in the example the receptive field is increased to a size of 3 x 3. The exponential increment applies for each additional level added when it comes to receptive fields, but the number of parameters will grow linearly.

Gating mechanisms are a method applied in order to control the gradients of a network [38]. As an example, there is a state called “exploding gradients” that can cause unstable networks unable to improve from the provided training data. When an error gradient accumulates over time, it can result in very large gradients. As a result, very large updates of the network weights can in turn cause instability, or in the worst case cause an overflow resulting in unreadable values.

By applying gating mechanisms to the model one is able to control and thereby avoid this kind of behavior.

Another aspect of this algorithm is residual learning. It is known as a bottleneck design that utilizes a gated linear unit and dilated convolution to create a residual block, with the aim to reduce network depth and maintain performance, while at the same time increasing receptive fields (kernels) to handle more information.

The drawback of this algorithm is the increased requirement of computational power due to the number of features it has to process. This is something to be aware of especially if the dataset has a high number of features in the first place which can often be the case when operating with image or audio data.

3.4 Audio denoising

Processing audio for noise removal is a problem that has been around for some time. The main goal of this process is to filter away as much of the background noise as possible from the noisy audio sample. A good example of this would be a conversation recorded in a high-traffic city environment. Ideally, post-process, only the speech would be left as an audio source. A conventional method to this has been through means of filtering or statistical approaches.

An example of a statistical method was suggested by [39]. Time-frequency audio denoising is the process where the short-time Fourier transform of the noisy audio sample is calculated. It is a way of uncovering the time-frequency signal structure that can be separated from the noise by applying a statistical approach. This converts audio into matrixes and runs computational algorithms in an attempt to identify and average out unwanted data.

Instead of the statistical approach, there is the usage of filtering. This method is a more direct attempt at removing unwanted sources from the audio sample. The target in this case is frequencies operating in different channels when taking spatial dimension into account [40]. It allows for locating audio sources that by combining temporal frequencies and spatial dimensions occur isolated from the rest of the audio mix.

One of the main challenges when trying to statistically handle noise, is computational strain due to the size of the matrix needed. Another problem when it comes to approaching by filtering noise is that these methods often add distortion. These methods work better when the complexity is low and audio sources remain more stationary.

This is why machine learning has quickly gained a foothold as a very good tool for audio denoising. The more sophisticated algorithms allow for higher precision in terms of removing noise and at the same time maintaining as much quality as possible of the target audio source.

3.5 Audio quality measurement

3.5.1 Perceptual evaluation of speech quality

Perceptual evaluation of speech quality is a model that initiates its operation by level aligning an input reference signal with a degraded signal, (we will use a denoised signal), to what is considered listening standard. A filter is applied to make use of the Fast Fourier transform. A new alignment of the signals in the time domain is then done, including equalizing linear filtering for gain variations [41].

This time alignment assumes that delay in a system is constant, which is valid in most systems including packet based transmission systems. The model has tolerance for some delays as long as they happen during silent periods and when speech is present. The following steps are involved in the time alignment process[41]:

- Narrowband filtering of both signals for what is considered as perceptual important parts.
- Estimation based on envelope delay.
- Signaldivision of the reference signal into utterance.
- Estimation based on envelope delay for each utterance.
- Fine correlation based on histogram delay identification for each utterance.
- Splitting utterance and re-aligning to test for changes in delay during the occurrence of utterance.

Part of the time-aligned processing involves mapping the signal into a model called auditory transform. The purpose is to unveil what is perceived as loudness in time and frequency. This process is solved through four steps [41].

First, the calculation is done using the fast Fourier transform and Hamming window to find the instantaneous power spectrum in each frame for 50% overlapping frames of 32 milliseconds duration. The result is then grouped into 42 bins equally spaced in the perceptual frequency on a scale named “Bark scale”. The step is known as the “Bark spectrum”.

The second step is Frequency equalization which calculates the mean Bark spectrum for active speech frames. The difference between the spectrum of the reference signal and the degraded signal gives an estimate of the transfer function, assuming the system being tested has a constant frequency response. Reference is then equalized to the degraded signal using this estimate along with a tolerance threshold of approximately 20 decibels.

The third step is the equalization of gain variation. Making use of the ratio between the audible power of the reference and the degraded in each frame is used as means of identifying variations in gain. A first-order low-pass filter is then applied and bound before the degraded signal is equalized to the reference signal.

The final step is mapping loudness using the Bark spectrum [41] from the first step. It includes a threshold depending on frequency and exponent. The result from this is a perceived loudness in each time-frequency cell.

Another aspect of the model is handling disturbance. Audible error is measured by calculating the absolute difference between the degraded signal and the reference signal [41]. This process is completed over multiple steps before calculating a non-linear average over time and frequency.

One of these steps is to delete a negative delay change. This leaves a section which overlaps with the degraded signal [41]. If the deleted part is longer than half a frame, the entire overlapping section is discarded. Another step is applying masking with a threshold that is set to the lesser of the loudness of the reference and degraded signal divided by four. This threshold is then subtracted from the absolute loudness difference with values less than zero set to zero. Masks are applied in each time-frequency cell.

The final step computes asymmetry [41]. It is performed by taking a stable ratio of the Bark spectral density of the degraded signal to the reference signal in every time-frequency cell. The result is raised to the power of 1.2 and bound with an upper limit of 12.0. If a value is less than 3.0 it is set to 0. Because of how the asymmetrically weighted disturbance is obtained through multiplication, only additive distortion are measured.

This method is complicated because of all the parameters that needs to be calculated as part of the process. In Figure 3.8 a diagram of the Perceptual evaluation of speech quality model is presented that illustrates how both clean and denoised signal flows through the model when applied as part of evaluation in the experiments we conduct in this thesis.

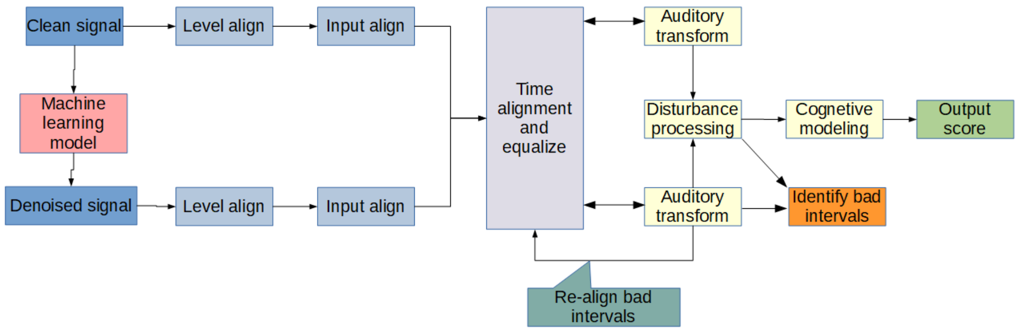

Figure 3.8: Simple examples of Perceptual evaluation of speech qualit flow

3.5.2 Segmental SNR

Signal to noise ratio is a well-established technique for measuring signal quality, audio being one of them. Measuring the relation between the desired signal level and amplitude of what is considered background noise yields a quality metric. Another way to describe the signal-to-noise ratio is the signal power to the noise power. Measurement is typically done in decibels, where a ratio greater than 0 or a higher than 1:1 indicates more signal than noise. Ideally, it is desired to have as large a gap as possible. the following scale is used for the classification of signal transmission in wireless networks but provides a good guideline for level classification.

- **5 to 10 decibels::** considered below the minimum level of what is accepted. In this case, signal and noise level is nearly indistinguishable from each other.
- **10 to 15 decibels:** is looked upon as barely acceptable but it can still be hard to distinguish the two signals from each other unless they are static.
- **15 to 25 decibels:** classified as a minimum for poor quality. Not ideal as analysis and processing might be a bit harder if signal and noise are dynamic leading to variance.
- **25 to 40 decibels:** considered as a good range to operate within. Signal and noise are easily distinguished from each other which makes the analysis and processing easier.
- **41 decibels or higher:** in this case, the ratio is considered excellent and does not cause any problems.

With the basic concept of signal-to-noise ratio explained it is important to emphasize that this in itself is not the best approach for measuring signal-to-noise ratio when operating with audio samples that contain highly dynamic sources that have variance in both presence and amplitude as is the case in this thesis. Therefore a slightly more advanced version is recommended with the capability of breaking up and looking at smaller segments of the audio sequence.

This approach allows for a better analysis as some audio sources are not present all the time, and when they are, might vary in amplitude. A suggested method to measure signal-to-noise quality in such a scenario is the segmental SNR. It is defined as the average of signal-to-noise values over segments, and calculates an arithmetic average of linear signal-to-noise followed by logarithm calculation for the respective segment where speech is considered active [42, 43]. This way it is possible to get a more correct result as only part where speech is present are calculated instead of the entire audio sample as a whole. Because of this, segmental SNR lends itself well to provide an objective quality measurement in experiments conducted as part of this thesis, as the target signal consist of speech.

Figure 3.9 illustrates how segmental SNR is measured compared to conventional signal-to-noise ratio. The blue line in the diagram represents a clean signal, and the red line noise. Normal signal-to-noise ratio is averaged to a value for the entire sample as illustrated by the dotted green line, while the dotted pink lines show how segmental SNR is calculated and varies within its respective segment.

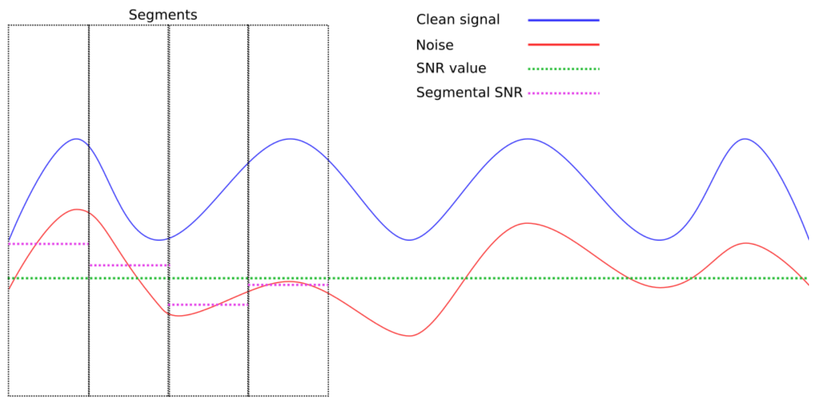

Figure 3.9: Illustration of segmental SNR

3.5.3 Short-time objective intelligibility

The short-time objective intelligibility was developed as a means of alleviating the need for time and resources spent to conduct listening experiments. The developers also had the process of noisy unprocessed speech in their minds, giving it the capability of assisting in providing answers for improving intelligibility in these scenarios. The core of short-time objective intelligibility is to create a function out of the clean and degraded speech signal. The output of this function is a scalar value where an expected monotonic relation to the percentage amount of correctly understood words averaged across a group of users [44].

The first part of the process is to segment both audio signals into smaller frames [44]. These frames are further analyzed to identify silent regions that do not contain speech, and as such do not contribute to speech intelligibility. Frames that do not contribute are therefore discarded. Then both audio signals are reconstructed without the removed frames where the energy of clean speech falls below a threshold of 40 decibels in relation to the maximum energy recorded in the clean speech frame.

The next step is to conduct a one-third octave band analysis. This is done by gathering discrete Fourier transform-bins into groups. Two thresholds are then used where a lower center frequency is set to 150 hertz, and an upper center frequency is set at 4.3 kilohertz. A total of 15 one-third bins are then distributed between these two thresholds. what happens next is that the changes in amplitude and frequency, also known as the short-time temporal envelope, are measured and compared between the clean and noisy speech signals by means of the correlation coefficient [44, 45]. The output of this analysis is a score between 0 and 1, where 1 is considered the better result. Just as with segmented SNR, short-time objective intelligibility suits this project well as a form of measuring as the target audio signal is speech in relation to noise which this method is specially developed to process.

In Figure 3.10 an illustration of the Short-time objective intelligibility model is presented. When compared to the Perceptual evaluation of speech quality model it has is a lot simpler.

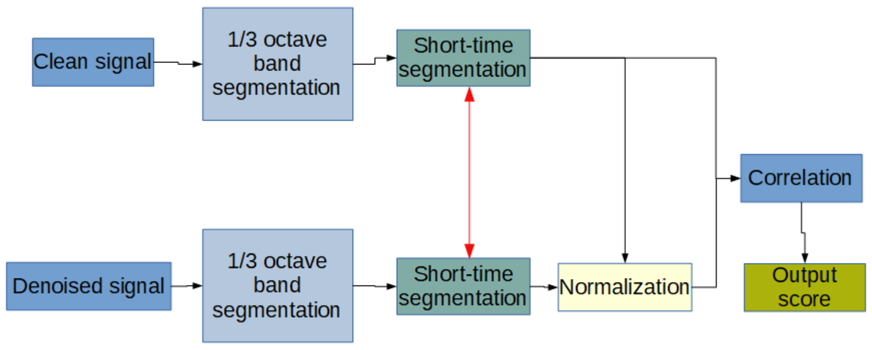

Figure 3.10: Illustration of Short-time objective intelligibility model

Chapter 4

Methodology

This chapter describes the methodology used to examine and address the research questions raised in this thesis. We will describe how datasets were selected and pre-processed to prepare for experimentation with machine learning models. We also describe both the physical and logical environment of the experimentation.

The following experiments are performed:

- Five different machine learning models are applied to conduct noise suppression in audio recordings.
- Audio analysis for comparing noise suppressed audio recordings with clean speech.
- Comparison analysis of noise suppressed audio data between machine learning models.
- Comparison of machine learning models and difference in noise suppression between types of audio noise.
- Dataset analysis looking for patterns in noise suppressed audio data.

Figure 4.1 presents an overview of the entire experiment process form a bird's-eye-view. Each step in the flowchart will be explained in more detail throughout this chapter.

4.1 Datasets

It is decided early in the process to build a synthetic dataset for training the machine learning models made use of in the experiments. The major reason for this is to avoid spending time sanitizing recorded audio out of privacy concerns when handling data containing human speech. Because of how this project revolves around audio denoising, the data needed is split into two categories. One would be clean speech acting as the target audio, and the other category is a variety of different audio sources to synthetically mix with the speech audio acting as noise. There are multiple audio datasets available for research related to machine learning. But many of them are not maintained, have restrictions related to data privacy, or include ambiguous data that may introduce a bias or other problems

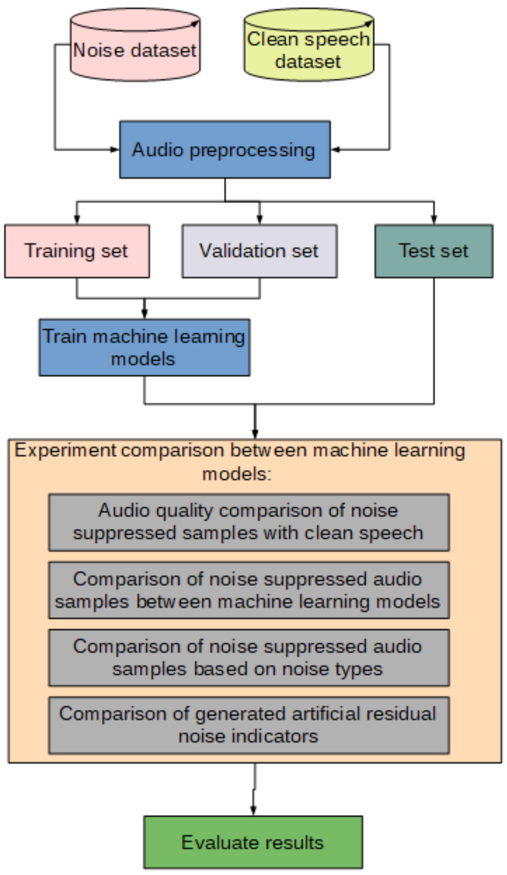
Noise dataset
Clean speech dataset
Audio preprocessing
Training set
Validation set
Test set
Train machine learning models
Experiment comparison between machine learning models:
Audio quality comparison of noise suppressed samples with clean speech
Comparison of noise suppressed audio samples between machine learning models
Comparison of noise suppressed audio samples based on noise types
Comparison of generated artificial residual noise indicators
Evaluate results

Figure 4.1: Experiment workflow overview

in the test results.

For the clean speech dataset, the decision fell upon Mozilla Common Voice[46]. It is a publicly available voice dataset licensed under creative commons, made up of voices from voluntary contributors. Another benefit is that it is maintained on a regular basis by receiving updates every third month. The common voice dataset also has multilanguage support if needed, but only the English samples are used as part of this project as they covered the requirement. It currently contains 81 085 voice samples and each approximately 3 to 5 seconds in length. This dataset also comes with a predefined splits into training, validation, and test data, which needs to be adhered to in order to ensure validation of the results.

Also, one criterion for the dataset in this project is to contain a wide selection of audio sources that could represent noise when mixed with speech data. It should also not consist of static noise like constant white or brown noise. The reasoning for this is that constant and/or stationary noise is easier to handle, and does not put the same challenge on machine learning algorithms in a real world environment setting because of a lack of variance[47]. It also means that manifestation of artificial residual noise is less likely to occur compared to non-stationary audio sources[21].

The decision, therefore, landed on UrbanSound8k[48] as noise data, licensed under creative commons. It is a dataset made up of labeled audio samples. Each clip is approximately 4 seconds in length, taken from the urban environment. It is divided into 10 different classes, and also has very clear instructions on how training should be conducted in terms of file usage, e.g. how to split the dataset and perform cross-validation to increase the validity of the end result.

There is also a need for a third temporary dataset consisting of mixed audio clips where clean speech and noise were used to create samples that the trained algorithms would attempt to process for denoising. As this data was only needed during testing, it is created on the run by mixing a set amount of speech samples with a set amount of noise samples from the respective test samples of each category. The same temporary dataset would then run against each machine learning algorithm for performance comparison of the same samples. An illustration of how the temporary dataset is used, is presented in Figure 4.2.

4.2 The Experimental phase

The experiment consisted of an environment split in two. It is described in detail in the following sections, along with how data is sampled, processed, and results extracted.

4.2.1 Physical environment

In this section a description of the physical resources that were utilized for computation and analysis for this thesis. As the training of machine learning algorithms benefits from running on a powerful Graphics Processing Unit and has a high demand for memory and disc space. To meet these requirements, a virtual machine dedicated to machine learning was set up with the following specifications:

- Type: Virtual machine
- OS: Ubuntu 20.04.3 LTS 64-bit
- CPU: Intel Xeon Processor (Cascadelake)
- RAM: 90 GB
- Storage: 40 GB
- GPU: Tesla V100 8GB GPU RAM (¼ of GPU = 1 core)

For script development, prototyping, audio analysis, audio processing, audio editing, and other relevant lab work, a virtual lab was set up on a personal laptop:

- Type: Virtual machine
- OS: Ubuntu 20.04.3 LTS 64-bit
- CPU: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
- RAM: 4GB
- Storage: 60GB

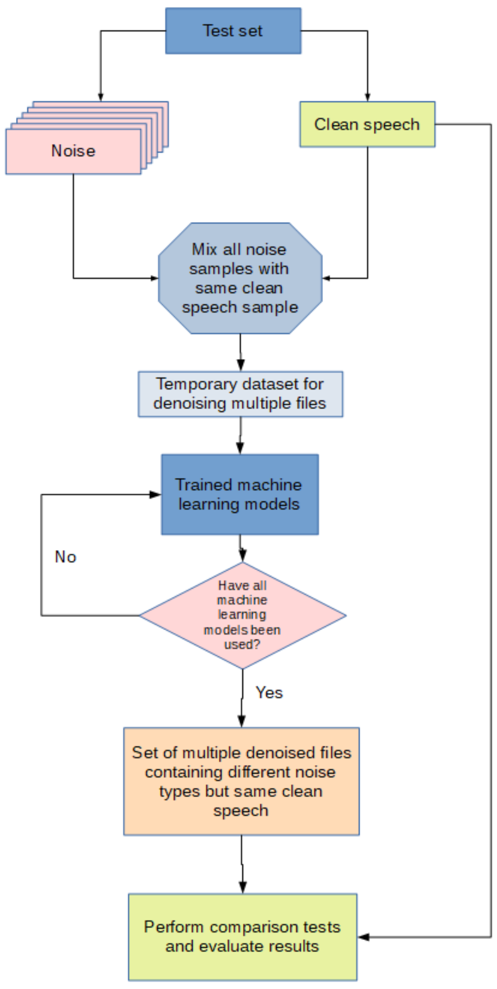
Test set
Noise
Clean speech
Mix all noise samples with same clean speech sample
Temporary dataset for denoising multiple files
Trained machine learning models
No
Have all machine learning models been used?
Yes
Set of multiple denoised files containing different noise types but same clean speech
Perform comparison tests and evaluate results

Figure 4.2: Temporary dataset flow

- GPU: VMWare SVGA controller

4.2.2 Logical environment

This list contains the major applications that made up the virtual laboratory when conducting the experiment along with their respective version number:

- Audacity 3.1.0 - Used for sound processing, analysis, and editing.
- Python 3.8.10 - Programming machine learning algorithms and other utilities.
- PuTTY - SSH and Telnet client for remote connection between lab machines.
- Rsync - used for taking backup of folder and files over ssh
- Sublime Text 3 - Text editor used for Python programming
- Vi IMproved (VIM) - Text editor used for Python programming.

- WinSCP - SFTP/FTP client used for file transfer between Windows client and virtual hosts over SFTP.

4.2.3 Audio database and machine learning algorithms

This phase involves creating an audio database consisting of audio noise, and clean speech samples from the datasets selected. Variants of deep neural network algorithms with different capabilities for conducting experiments and tests are also evaluated. Five different algorithms are chosen based on their foundational similarity, but at the same time having variations in capability. These machine learning algorithms and their main features was introduced in section 3.3.2 Convolutional Neural Network, 3.3.3 Convolutional Recurrent Neural Network, 3.3.4 Densely Connected Convolutional Networks, 3.3.5 Recursive Network with Dynamic Attention, and 3.3.6 Gated Residual Network With Dialated Convolutions.

Each model is programmed in Python 3.8.10 [49] using API, example code and documentation as provided by Tensorflow [50] in order to conform to the frameworks layered standard. This also made application of Tensorflows storage system easier as it is natively supported out of the box. The Tensorflow API has been in development since 2015, and all functionality needed for building the models part of the experiments were available as integrated modules.

To provide an example of how these modules work in Tensorflow, we can start by looking at the convolutional block in the convolutional neural network. The convolution layer is available in several variants depending on the need. Audio data that we use in this experiment is evaluated in two dimension (time and frequency), and the convolutional 2 dimensional layer, Conv2D [50], is considered the best option. The layer can be configured by specifying filters, kernels size, and stride in order to optimize prediction, and some experimentation is needed in order to find the optimal values.

After the convolution layer, an activation layer is set that takes the output from the convolutional layer as an input. The role of an activation layer is to serve as a helper function which in this case applies Rectified Linear Unit (ReLU) to the network, as discussed in chapter 3, section 3.3.2. The activation layer is followed by a batch normalization layer taking in the output of the activation layer and normalizing the data. By combining activation layer and batch normalization like this the vanishing gradient problem is addressed as discussed in chapter 3, section 3.3.2. The output from the batch normalization layer is forwarded in the network to the next convolution block and the process repeated until every block has been traversed, and output in form of a prediction is compared against the validation target.

All of the machine learning models used in the experiments make use of the Tensorflow modules as building blocks to ensure that the functionality is standardized across all 5 of them. The structure of each machine learning model used in the experiments are based of the following papers:

- Convolutional Neural Network was based on the paper made by Park and

Lee [51]
- Convolutional Recurrent Neural Network Was built using Tan and Wangs suggested method [34].
- Densely Connected Convolutional Networks is based of Huang et al. [36]
- Recursive Network with Dynamic Attention was built based on the paper presented by Li et al. [12].
- Gated Residual Networks With Dilated Convolutions are built from two papers by K. Tan, J. Chen and D. Wang [37, 38]

Every model used during experimentation with a detailed diagram description of its modules, their respective output shapes, number of paramters and connections, are presented in Appendix A, B, C, D, and E for reference.

4.2.4 Feature extraction

The main part of the experiment goes into data preprocessing and preperation. The clean speech data of the Mozilla Common Voice dataset[46] is first split into three respective parts, training, validation, and test. This is done according to the documentation of the dataset and using the three tsv-files supplied along when downloaded. The tsv-files contain a pre-determined split of the data based on analysis and classification deciding on which files are suited for each of the three categories. The result is 864 450 audio files in the training set, 1 530 387 files in the validation set, and 16 328 files in the test set.

The next step is to sort out the binary mp3 files that correspond to the files in the three different tsv-files. The audio files are converted to wav format in the same process as they are sorted and moved by category. The format of the noise data represented by UrbanSound8k[48] dataset comes in wav format which is preferred over mp3. Compressing UrbanSound8k down to mp3 format that the Mozilla Common Voice comes in, is not wanted because of how it can degrade the audio quality in the process. Converting from mp3 to wav does not involve additional loss to the audio quality which is desired behavior, and therefore the method used in this experimental setup.

After splitting the data into three different sets, each file in its respective dataset is broken down into logical information by extracting a spectral magnitude vector from the short-time Fourier transform consisting of 256-points. In order to calculate this, window size is set that defines how big of a segment will be analyzed at a time. As this window needs to move across the audio sample in order to analyze it in its entirety, a hop size value of 64 is also set.

The defined window then slides across the audio sample based on the hop size and calculates the Discrete Fourier Transform (DFT)[26] of the data within each segment. It is used to build the short-time Fourier transform by simply combining the acquired segments of the discrete Fourier transform. Then, the magnitude vectors of the signal are extracted from the 256-point short-time Fourier Transform, but only the first 129 points are needed so the symmetric half is discarded.

As machine learning algorithms make use of features for training, these need to be extracted from the short-time Fourier transform. By using the 256 points set in the short-time Fourier transform and the hop size of 64, the short-time Fourier transform vectors will overlap by 75% to avoid loss of data when a function moves towards zero at the boundaries. Then 8 consecutive noisy short-time Fourier transform vectors are concatenated together that will be used as input for the algorithm with a shape of (129, 8). The input vector consists of the current short-time Fourier transform vector, along with 7 previous ones. Figure 4.3 illustrates the feature creation process.

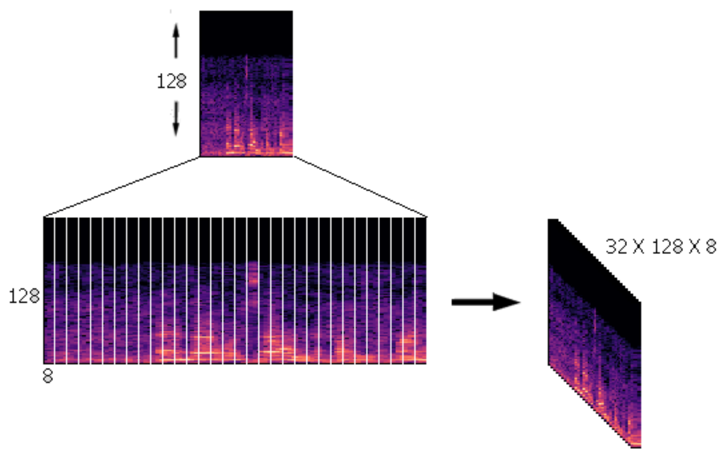

Figure 4.3: STFT feature creation

This makes it so that the machine learning algorithm can read the profile of the audio sample. Each short-time Fourier transform vector is stored in a corresponding entry in their respective Tensorflow record file representing the datasets. Figure 4.4 illustrates the flow of the feature selection process of the Mozilla clean speech data.

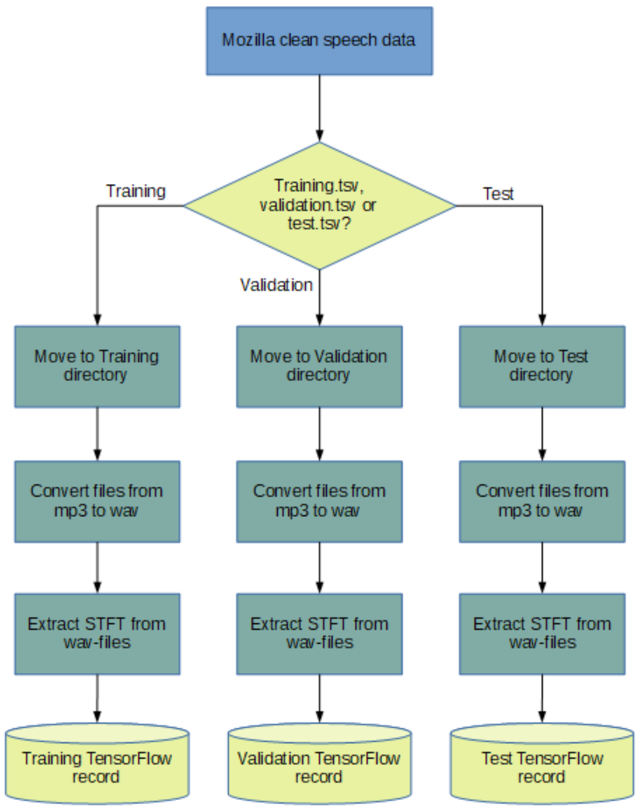
Mozilla clean speech data
Training.tsv, validation.tsv or test.tsv?
Training
Validation
Test
Move to Training directory
Move to Validation directory
Move to Test directory
Convert files from mp3 to wav
Convert files from mp3 to wav
Convert files from mp3 to wav
Extract STFT from wav-files
Extract STFT from wav-files
Extract STFT from wav-files
Training TensorFlow record
Validation TensorFlow record
Test TensorFlow record

Figure 4.4: Clean speech feature extraction flow

In parallel, pre-processing of the noise files is conducted. This data is structured into training, validation, and test data like the Mozilla Common Voice dataset[46]. The approach to the split is however a bit different from the clean speech dataset. UrbanSound8k[48] comes with 8732 audio files in wav format split across 10 folders. One folder is dedicated to testing data as stated in the documentation for the dataset. The 9 other folders are split into training and test datasets. As pointed out by the documentation, no shuffling of the noise files is done in order to not mix audio samples meant for testing with the training and validation set, and vice versa.

When the dataset containing noise data is split, the same conversion process from audio to short-time Fourier transform and feature extraction as with the clean speech data is conducted. The output is stored as part of the training and validation TensorFlow record[50]. The training and validation variant is then uploaded to the physical machine responsible for handling the training of the machine learning models. At the same time, the test records are stored on the machine dedicated to testing, analysis, and other tasks. The binary wav version of the test data files is also uploaded to the laboratory machine that will make use of these files when conducting experiments at a later stage. The flow of this process is presented in Figure 4.5.

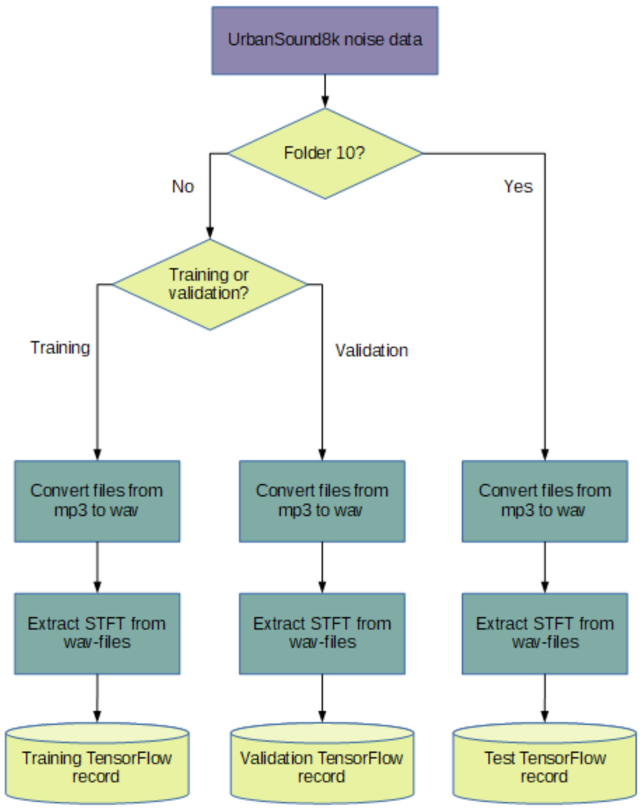
UrbanSound8k noise data
Folder 10?
No
Yes
Training or validation?
Training
Validation
Convert files from mp3 to wav
Convert files from mp3 to wav
Convert files from mp3 to wav
Extract STFT from wav-files
Extract STFT from wav-files
Extract STFT from wav-files
Training TensorFlow record
Validation TensorFlow record
Test TensorFlow record

Figure 4.5: Noise feature extraction flow

4.2.5 Training and validation of machine learning algorithms

With data prepossessed and prepared the next stage is training and validation of the machine learning algorithms. This is done by making use of the created training and validation record files. Because of the size of both training and validation data available, a subset is created by randomly selecting from each respective training and validation pool. A total of 4000 samples equally divided providing 2000 noise samples and 2000 clean speech samples makes up the active training set for the algorithms, and the validation set is made up of 1000 clean speech samples and 200 noise samples. The reason for differentiation in ratio in validation samples is because of wider variation in in clean speech samples making them more difficult for the algorithm to distinguish, and as such more sample are needed.

When the training and validation sets are provided to the model, the first step is to shuffle and mix random clean speech and noise samples from the training set. These mixed noise samples are then processed by the machine learning model who makes a prediction that is compared against the clean speech validation target. The difference between the prediction and the validation target is measured and optimized using mean square error for each step in an epoch. The result of the mean square error calculation is stored in a variable for the next epoch if one of the following two conditions are met:

1. Current epoch is the first one in the training cycle, or
2. Result of current mean square error calculation is better than the previous one.

A new epoch is then initiated, and the process of making prediction is repeated until either the max number of epochs are reached, or no further improvement in precision is detected for a set number of tries (patience value). A simple illustration of this process is presented in Figure 4.6

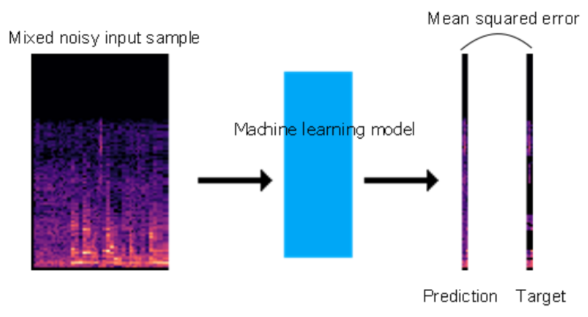

Figure 4.6: Machine learning model training example

Multiple tests were done for each machine learning model in order to try to find an optimized setting for training and validation in order to maximize the precision score. This resulted in a setup that worked well for all 5 models used as part of this experiment. Max number of allowed epochs is set to 400, with 400 steps per epoch using Mean Square Error (MSE) [52] as the loss function and “Adam” [53] as the optimizer function. A learning rate of 3e-4 yielded the best precision during test runs, with precision overall evaluated using Root Mean Square Error(RMSE) [54]. With this setup, all models reached root mean square error values of between 13.0 and 16.5. A patience value was set to 50 in order to avoid overfitting or underfitting the model, by forcing a drop out if no improvement where detected within 50 steps.

As previously explained in section 4.2.4 Feature extraction, input was provided to the models in form of Tensorflow records [50]. The settings were a window length of 256, with an overlapping hop size of 64. The input sample rate was set to 48000 and the output sample rate at 16000 in order to fit the data. The number of features is, as previously discussed during feature extraction, set to 129 by discarding the symmetrical half. The number of segments is set to 8 as is the number of noisy short-time Fourier transform vectors concatenate together. Once finished the model is saved as an h5 file for later use during testing of the trained machine learning algorithm.

4.2.6 Test, comparison, and analysis

When all machine learning algorithms have finished training, a series of tests are conducted. First, all of the test noise samples are mixed with the same randomly selected clean speech clip and denoised by the trained machine learning models, as was previously illustrated in Figure 4.2. Each of these denoised samples are tested against the clean speech sample with all quality measure methods (perceptual evaluation of speech quality [41], short-time objective intelligibility [44], and segmental SNR [42]), for all the machine learning models.

The result is then stored in tables by the respective type of audio measuring quality used. Each table hold scores for the 5 machine learning models in all of the 10 noise categories. In addition, a table with an overall score for the measured quality models is also stored, along with the variance for each parameter to display stability (reliability) within the respective category. The purpose of this test is to look for patterns and correlations in data that can help understand if there are indicators that help estimate if and why artificial residual noise is likely to manifest in quality measurement. These results will be presented in chapter 5.

Another test conducted is done on the short-time Fourier transform data. In this test, the noisy audio sample is compared against the denoised version of itself. Then the noisy audio sample is subtracted, element by element, against the noise suppressed version, and the remaining magnitude (in decibel) is stored in the indexes of the array. An analysis is then performed summarizing each segment in the new array, and comparing the corresponding entry in both noisy and noise suppressed versions. The purpose of this test is to look for segments where the total magnitude is higher than it initially was before the noise suppression model processed the audio sample. If such segments are found, it indicates that values in the segment have been amplified by the model while conducting audio noise suppression.

What is achieved by measuring the magnitude of decibel in the segments, is to uncover if there is additional artificial noise added by the machine learning model and not removed as part of denoising. As documented earlier in this thesis this is not wanted behavior and not the purpose of the noise suppression process. The results of this test will be covered in chapter 5, and further discussed in chapter 6.

For the purpose of better understanding, a plot is created for visual analysis where the noisy sample is compared to its respective denoised variant. Then the difference between these two is calculated and plotted for comparison. The aim of this is to visually present differences in how each machine learning model handles noise suppression. Examples of this will also be presented in chapter 5.

Chapter 5

Results

Table 5.1 displays the calculated result of the overall performance of each machine learning model while disregarding the type of noise. This is done for all of the three objective quality measure types (PESQ[41], STOI[44], and segSNR[42]). Two values are presented for each attribute, the mean value, and the variance. The mean value for each attribute operates with its respective scale.

- For PESQ[41] this ranges from -0.5 to 4.5 where a higher value equals a better score.
- The STOI[44] scale ranges from 0 to 1, where a higher value also equals a better score.
- SegSNR[42] indicates the mean value of the segment's distance between noise and clean signal value, with the higher the value equals less noise.

In addition a color gradient has been applied to each column to easily display which machine learning model that performs best and which one that is worse. The more green a color get, that better it is compared to the other values in the column, and in the opposite way the more red in a color, the worse the score. As an example in the case of PESQ mean, crnn_model has the best score, while grn_model has the worst score.

Model	PESQ mean	PESQ variance	STOI mean	STOI variance	segSNR mean	segSNR variance
cnn_model	1.099780459	0.002260387	0.660671655	0.001384592	12.07599318	4.402803336
crnn_model	1.144246805	0.003259915	0.633326674	0.001534108	12.12821492	5.253023318
dcn_model	1.0843152	0.001081946	0.672776618	0.001365926	14.50326785	2.094892975
grn_model	1.057394427	0.0003396	0.640932738	0.00091489	15.17393267	1.74317249
darcn_model	1.073179877	0.00122739	0.647036492	0.00172265	14.40728753	2.109838259

Table 5.1: Table with overall measured audio parameter results.

5.1 PESQ

In Table 5.2 mean PESQ score for every type of noise for each machine learning algorithm is presented. The values adhere to the scale of PESQ, with a range from -0.5 to 4.5 where a higher value equals a better score. In order to clarify the comparison within each noise type category, a color gradient is applied for easier identification of which algorithm performed best within the respective type of noise. The more red colors indicate worse performance, while the darker the green color gets, the better. As an example looking at the air conditioner category, observe that crnn_model displays a dark green color meaning the best score in this case, while grn_model has a dark red color meaning the worst score in this case.

PESQ	cnn_model	crnn_model	dcn_model	grn_model	darcn_model
Air conditioner	1.104935395	1.197383982	1.08810618	1.054722317	1.073293406
Car horn	1.130879706	1.182435343	1.088397629	1.055669792	1.067690521
Children playing	1.095845003	1.146176292	1.089255241	1.057913768	1.075058246
Dog barking	1.122073152	1.151795028	1.122972125	1.080601865	1.106088151
Drilling	1.046038321	1.087320496	1.042019299	1.035611713	1.037560264
Engine idling	1.132795334	1.218950401	1.100585163	1.056868529	1.094520384
Gun shot	1.149410788	1.14348406	1.116701256	1.0935959	1.114204649
Jackhammer	1.032994211	1.072438526	1.035941976	1.032713068	1.030565228
Siren	1.092828436	1.112628328	1.078982347	1.05366947	1.067885735
Street music	1.090004247	1.129855598	1.080190778	1.052577851	1.06493219

Table 5.2: Table presents individual models versus different noise evaluated with the PESQ method.

5.2 STOI

In Table 5.3 mean STOI score for every type of noise for each machine learning algorithm is presented. Just as for the PESQ table, the values adhere to the scale of STOI, with a range from 1 to 1 where a higher value equals a better score. In order to clarify the comparison within each noise type category, a color gradient is applied for easier identification of which algorithm performed best within the respective type of noise the same way as in Table 5.2. The more red colors indicate worse performance, while the darker the green color gets, the better. As an example looking at the air conditioner category, observe that dcn_model displays a dark green color meaning the best score in this case, while crnn_model has a dark red color meaning the worst score in this case.

STOI	cnn_model	crnn_model	dcn_model	grn_model	darcn_model
Air conditioner	0.644003585	0.623302488	0.661278724	0.630346146	0.629813752
Car horn	0.682290349	0.655221305	0.686688409	0.656303558	0.654829015
Children playing	0.652431059	0.622479856	0.673321076	0.637174805	0.646996058
Dog barking	0.674307843	0.653010556	0.695070636	0.658501593	0.682111815
Drilling	0.649870385	0.620667281	0.655494138	0.629997111	0.626752435
Engine idling	0.674859373	0.651109121	0.693842387	0.652869564	0.666994517
Gun shot	0.701875848	0.669436178	0.710991041	0.672370093	0.703803776
Jackhammer	0.630448699	0.615829409	0.62439975	0.611270728	0.587788841
Siren	0.670073794	0.629398462	0.681257269	0.647538513	0.658220513
Street music	0.626555617	0.592812083	0.645422753	0.612955274	0.613054194

Table 5.3: Table presents individual models versus different noise evaluated with the STOI method.

5.3 SegSNR

In Table 5.4 mean segSNR score for every type of noise for each machine learning algorithm is presented. Just as for the PESQ table, the values adhere to the scale of segSNR, where a higher value equals a better score. In order to clarify the comparison within each noise type category, a color gradient is applied for easier identification of which algorithm performed best within the respective type of noise the same way as in Table 5.2. The more red colors indicate worse performance, while the darker the green color gets, the better. As an example looking at the air conditioner category, observe that grn_model and darcn_model display a dark green color meaning the best score in this case, while cnn_model has a dark red color meaning the worst score in this case.

SegSNR	cnn_model	crnn_model	dcn_model	grn_model	darcn_model
Air conditioner	12.59056241	12.93791829	14.53924116	14.93326674	14.93482928
Car horn	10.90736553	10.12490989	14.25367265	14.24967889	14.38444214
Children playing	12.00657207	12.41872246	14.29026557	14.96538645	14.37142694
Dog barking	11.12435252	12.29470361	13.81222411	14.46731325	13.7192939
Drilling	13.38273395	11.96080167	15.39481887	16.09473059	15.0470475
Engine idling	11.43564798	11.40992421	13.88282	14.94908703	13.57149935
Gun shot	9.100205013	13.13440924	13.05952191	13.66087898	13.03540009
Jackhammer	13.66964467	10.84717231	15.0236329	16.47431912	14.65657288
Siren	12.22524543	11.89114628	14.70636202	15.42016495	14.4512208
Street music	14.31760226	14.26244128	16.07011936	16.52450067	15.90114246

Table 5.4: Table presents individual models versus different noise evaluated with the segSNR method.

5.4 Short-Time Fourier Transform analysis

Presented in this section are example outputs from analyzing the short-term Fourier transform. The following 5 figures 5.1,5.2, 5.3, 5.4, and 5.5, are each divided into three spectrograms. The one located at the top in each of them is extracted from the same audio recording containing noise from an air conditioner before it is processed for audio noise suppression by each respective machine learning model. The middle spectrogram in each figure is the denoised version of the same audio recording processed by the respective model. The bottom spectrogram in each figure displays the difference between the top and middle spectrogram in each respective case.

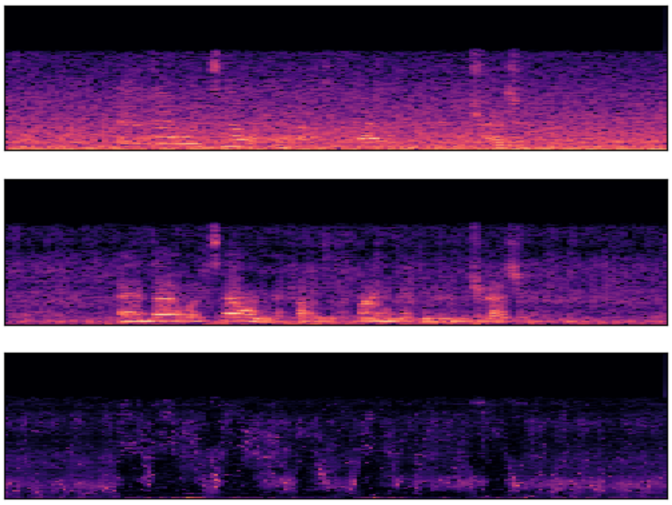

Figure 5.1: Convolutional Neural Network noise to noise suppressed comparison

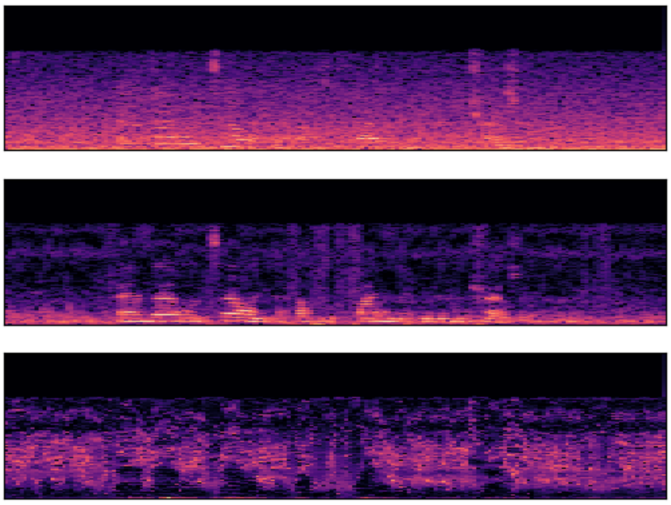

Figure 5.2: Convolutional Recurrent Neural Network noise to noise suppressed comparison

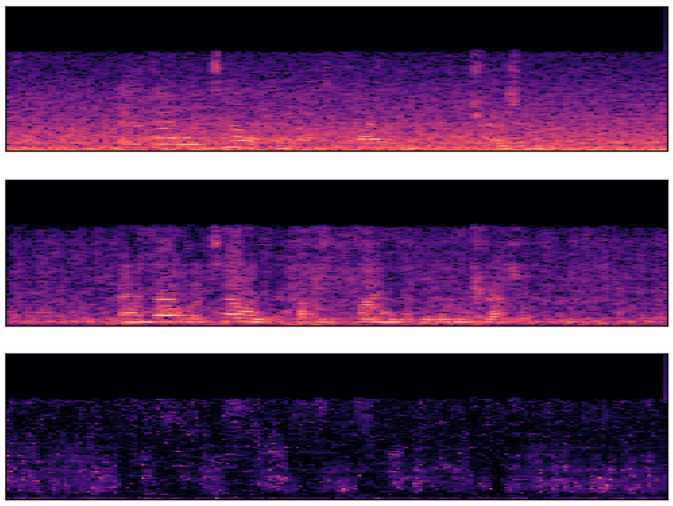

Figure 5.3: Denseley Connected Convolutional Network noise to noise suppressed comparison

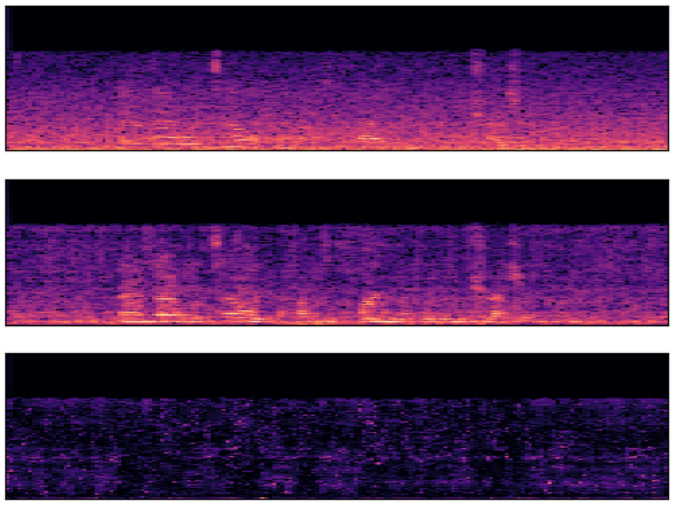

Figure 5.4: Recursive Network with Dynamic Attention noise to noise suppressed comparison

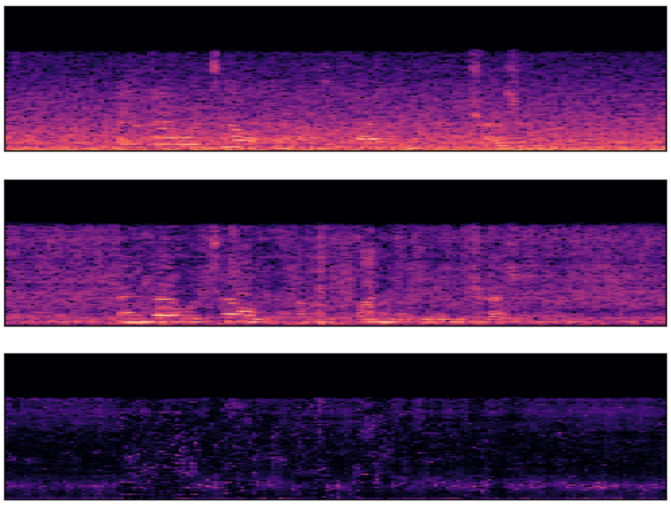

Figure 5.5: Gated Residual Networks With Dilated Convolutions noise to noise suppressed comparison

The next results presented are from comparing noise and denoised short-time Fourier transform segments for each model for sample from each category of noise. Only segments in the denoised audio recording, where a value is higher than the corresponding segment in the noisy audio recording were counted against the total. These numbers are presented in Tables 5.5 to 5.14 both as integers and percent.

Air conditioner		
Model	Segemnts observed	Percentage
cnn_model	21	2%
crnn_model	22	2%
dcn_model	216	21%
darcn_model	263	26%
grn_model	341	33%

Table 5.5: Number of segments with difference - Air conditioner.

Car horn		
Model	Segemnts observed	Percentage
cnn_model	33	3%
crnn_model	95	9%
dcn_model	177	17%
darcn_model	488	48%
grn_model	446	44%

Table 5.6: Number of segments with difference - Car horn.

Children playing		
Model	Segemnts observed	Percentage
cnn_model	177	17%
crnn_model	294	29%
dcn_model	71	7%
darcn_model	317	31%
grn_model	277	27%

Table 5.7: Number of segments with difference - Children playing.

Dog bark		
Model	Segemnts observed	Percentage
cnn_model	509	50%
crnn_model	531	52%
dcn_model	759	74%
darcn_model	668	65%
grn_model	676	66%

Table 5.8: Number of segments with difference - Dog bark.

Drilling		
Model	Segemnts observed	Percentage
cnn_model	40	4%
crnn_model	338	33%
dcn_model	303	30%
darcn_model	621	61%
grn_model	302	29%

Table 5.9: Number of segments with difference - Drilling.

Engine idling		
Model	Segemnts observed	Percentage
cnn_model	0	0%
crnn_model	3	0%
dcn_model	0	0%
darcn_model	104	10%
grn_model	121	12%

Table 5.10: Number of segments with difference - Engine idling.

Gun shot		
Model	Segemnts observed	Percentage
cnn_model	365	36%
crnn_model	603	59%
dcn_model	637	62%
darcn_model	720	70%
grn_model	476	46%

Table 5.11: Number of segments with difference - Gun shot.

Jackhammer		
Model	Segemnts observed	Percentage
cnn_model	24	2%
crnn_model	68	7%
dcn_model	16	2%
darcn_model	11	1%
grn_model	26	3%

Table 5.12: Number of segments with difference - Jackhammer.

Siren		
Model	Segemnts observed	Percentage
cnn_model	28	3%
crnn_model	11	1%
dcn_model	9	1%
darcn_model	381	37%
grn_model	141	14%

Table 5.13: Number of segments with difference - Siren.

Street music		
Model	Segemnts observed	Percentage
cnn_model	184	18%
crnn_model	201	20%
dcn_model	136	13%
darcn_model	369	36%
grn_model	290	28%

Table 5.14: Number of segments with difference - Street music.

Chapter 6

Discussion

In this chapter, the experiments conducted in chapter 4 and results presented in chapter 5 are discussed to answer the research questions defined in section 1.5.

- *Research Question 1: How artificial residual noise appear in audio recording enhanced with machine learning?*

In section 2.2 Introduction to artificial residual noise artificial noise and residual noise were explained. Artificial residual noise is a product of the combined components of these two types of noise, as it would fulfill the following criteria:

1. It is residual noise left behind as the filtering process was not able to suppress or cancel all of it.
2. Noise present is an artificial audible product of a machine learning algorithm, which is considered a human device.

Both of these criteria were also proven to be fulfilled after experimentation described in section 4.2.6 on short-Time Fourier transform, and the result presented in 5.4 on how to detect both residual and artificial noise.

The reason for this was that the example spectrograms demonstrate that when looking at the difference between the mixed audio recording containing noise, and the processed audio recording where the machine learning algorithm has made an attempt at suppressing the noise, there is still residual noise present in all 5 algorithms. If no residual noise was present, the bottom spectrogram in Figures 5.1 to 5.5 would have only a background color of black, which is not the case in any of them.

When it comes to the presence of artificial noise, Tables 5.5 to 5.14 are based on observations of elements in the short-time Fourier transform array that have a higher magnitude in terms of decibel than they originally had in the noisy short-time Fourier transform array. If this is the case, the machine learning model would have increased the magnitude instead of suppressing noise as it is intended to do. As analysis of the data show, it is only in a couple of cases where additive noise did not occur. In fact, there were cases reporting as much as 70 percent alteration to the array which is not desired.

The presence of artificial residual noise can also be heard if the denoised audio recording is played. It occurs as distorted artifacts that are not present in the original noisy audio recording. In some cases, it can be misinterpreted as whispering or heavily distorted speech. In other cases where added noise come in for om a short burst of sound, the artifact manifests either right before, or right after, and in some cases both before and after the burst of sound. It sounds like digital distortion and does not resemble anything natural.

In relation to research question 1, it is considered as answered based on how unsuppressed noise created by the machine learning model used was not removed during the denoising process.

- *Research Question 2: Is there a difference in amount of artificial residual noise produced related to difference in types of noise sources? If this is the case how much of a difference is there and which types are more prone to make i manifest?*

When analyzing the results in Tables 5.5 to 5.14, there are variations when comparing the type of noise versus how the machine learning algorithm performs. These differences can be seen when comparing machine learning complexity within a noise category, and there are also variations when comparing overall performance between different noise categories. This can also be further analyzed by taking in the quality measurements in Tables 5.1 to 5.4, by looking at how the machine learning algorithms scored overall and in each category.

The two advanced models, the recursive network with dynamic attention and the gated residual networks with dilated convolutions, seem more prone to generate artificial residual noise compared to the simpler models convolutional neural network and the convolutional recurrent neural network. The densely connected convolutional network is in the middle to lower tier, which is interesting considering that it could be classified as a simple algorithm because of its similarities with the convolutional neural network, except for the added flow of information through interconnected layers.

Type of noise appears to have some impact as noise like children playing, dog barking, drilling, gunshot, and street music seems to have produced a higher count of changed segments than simpler more monotone with little variance, and repeating noise like engine idling and jackhammer. This observation is supported by the theory stating that non-stationary audio sources are more difficult to process correctly [8]. The meaning of this is that the more variation there is to an audio source for instance movement or change in amplitude, the more difficult becomes to properly process it.

In the area of audio forensics, this poses a great challenge. Audible evidence is after all taken from the real world and not a laboratory with a controlled environment. The chance of having to work with the presence of highly non-stationary audio sources is very likely all the time. But as proven in this experiment, one must be aware of the limitations in the capability of machine learning models the more complex the audio sources become.

- *Research Question 3: How much of a difference is there between different types of machine learning algorithms and occurrence of artificial residual noise?*

By looking at the convolutional neural network model, which can be considered the simplest of the 5 used in this thesis, performs well in all categories except for segSNR compared to the other models. In addition, the algorithm does not have the same number of changed segments as the more advanced algorithms in most cases. What should be noted is how variance can impact this algorithm, especially the segSNR variance. With a segSNR mean value of 12.07 and a variance of 4.4 decibels, it means that the segSNR value can drop beneath 10.

What happens then is that the audio recording get heavily distorted. If it were to be presented as evidence in an investigation it would not have any value as the speech can not be distinguished from the background noise. This is also illustrated in Figure 6.1 where the spectrogram shows that there is a good amount of residual noise still present after the noise suppression process.

A model that handles segSNR very well both in terms of mean value and variance is the gated residual networks with dilated convolutions. Observing the difference spectrogram in Figure 6.5 displays less residual noise compared to most of the other algorithms. The mean value makes it so that it is the only algorithm that breaks into the 15 to 25-decibel bracket as explained in section 4.4.2.

The main problem with this model however is that it has a tendency of altering a rather high amount of segments at times, which becomes a problem in a forensics situation as it introduces artificial residual noise as high as 66 percent in one case. In other words over two-thirds of the segments in the audio sample show alterations that are above what the original noisy audio recording had.

When comparing the results of altered segments in the results, the more advanced machine learning models are more often outputting a higher number of alterations than the basic convolutional neural network. One explanation of this can be related to the loss function of the models.

Using mean squared error as a loss function comes with some limitations. This is tied to how it optimizes the average of training examples. Taking this approach limits precision in prediction, especially when handling the more complex samples. It is also important to keep in mind that if the model makes a bad prediction mean squared error magnifies it. Therefore, in order to increase precision in the results, a custom loss function should be applied that is adapted to better handle audio noise samples of a more complex nature.

When presenting evidence as a forensic investigator, having this amount of alteration to an audio sample would not be accepted. The fact that all models show alterations in most cases would not be tolerated, and both reliability and integrity in the data become a big concern.

- *Research Question 4: Why is it important to detect and/or mitigating artificial residual noise, and what are the consequence of not handling it properly in the context of audio forensics?*

Considering the results and discussion presented up to this point creates the foundation for the argument of why it is important to both detect and mitigate the foundation of artificial residual noise. As discussed under research questions 2 and 3, there are clear indicators that data is being altered by introducing artificial additive noise, when comparing the noisy audio recording and the processed denoised version of the respective audio sample.

From a forensic investigations point of view, this makes the investigation more difficult. The fact that artificial audio manifests in the audio recording essentially introduces a new audio source that needs to be considered. This can in the worst case force the investigator to go back several steps depending on how the audio source behaves. The investigator needs to determine if this new source is of significant importance to the ongoing case. If the artificial noise resembles known sources, like speech, or objects of which presence might be important for the investigation, it will be considered worth focusing on until properly identified and classified.

Essentially misleading the forensic investigation this way will come with a cost of both time and resources. In time-critical investigations where there is marginal room for mistakes, it can not be tolerated and should not happen. Especially considering that noise suppression in audio recordings that make use of machine learning is a tool where processes can and should be controlled to such a degree that the tool itself and its output integrity is known and can be fully trusted.

This brings up the next problem of importance, violation of dataset integrity. Even if the main concept of audio denoising is to some degree to make alterations to the data in such a way that it enhances audio sources of interest, what happens in this case, as discussed, is an alteration that negatively influences the data. What needs to be addressed when it comes to artificial residual noise is maintaining control of the number of audible sources in the recording and the level at which they contribute to the overall soundscape of the recording.

Failing to maintain data integrity and using reliable methods results in, as history has proven, situations where evidence becomes not usable in a court of law. Considering the importance of an investigation method being looked at as reliable and proven in order to be presented as the method for providing evidence, audio forensics using machine learning methods has to demonstrate that precision is narrowed down to the point where it is beyond reasonable doubt that the result can wrong. Anomalies like artificial residual noise will undoubtedly weaken the accountability and position of evidence no matter what degree it appears.

6.1 Limitations

As this thesis has focused on understanding and identifying artificial residual noise, some known audio quality measure methods in the form of perceptual evaluation of speech quality, short-time objective intelligibility, and segmental SNR have been applied. These methods are often made use of in research papers when determining the precision en performance of the machine learning model by measuring the original and processed audio recording.

When trying to identify the presence of artificial residual noise, these quality metrics can help in providing information on the quality of the audio recording, but they do not provide clarification on whether or how the machine learning algorithm introduced artificial residual noise. In an everyday setting where the aim is to present as good a result as possible like clean speech, manipulation of data is of less importance as long as the purpose is met. In these cases, these kinds of metrics serve their intention.

In a forensic setting where the integrity and reliability of the result weigh more than the quality of the output, this does not suffice and has more requirements that must be fulfilled to be considered as an appropriate method that can be considered as a valid go-to method for usage outside of a laboratory.

Chapter 7

Conclusion

The application of machine learning in audio analysis has become an established field of research. Audio noise suppression is one method that is derived from this technology and has become a common method of enhancing audio sources while controlling background noise. In the related field of audio forensics, however, there is still some ground to be covered until technology can be fully accepted.

Machine learning algorithms tend to focus on achieving as good a result as possible, without taking the preservation of data integrity into account. Deciding on this approach sacrifices reliability and data integrity, which are cornerstones when handling any kind of evidence in an investigation. The importance of maintaining a chain of custody can not be set aside in order to increase precision.

When conducting forensic analysis, how the data at hand is treated along the way is almost more important than the result. It is, after all, a case where the focus lies with the idea of data already containing the result, and it is more a matter of refining it in a proper way to highlight important information.

The purpose of this thesis is to prove what artificial residual noise is, how it manifests and why it is important to take action related to it from an audio forensics point of view. We have achieved this by conducting experiments that yields results containing artificial residual noise, illustrating how it can be uncovered, and discussing the importance of properly addressing it by presenting how it can negatively impact the work of a forensic investigation.

As we demonstrate with results presented chapter 5, and through discussion in chapter 6, indicators of artificial residual noise are identifiable when looking directly at the data arrays holding information on audio signal amplitude. We also demonstrate that conventional methods for quality measurement of audio signals is not an efficient way of identifying artificial residual noise.

Machine learning can become a valuable tool or supplement in the forensic area because of how it can reduce the need for manual labor. But in order to get to that point, there is a need for more research in order to improve methods that can address the challenges that negatively impact the technology today from a forensic perspective. Therefore some proposed future work is presented in chapter 8.

Chapter 8

Future work

Based on the discussions and results presented in this thesis, there are multiple challenges remaining that offer path for future work and research. Some of these are presented below:

- **A thorough analysis of STFT array:** When analyzing the audio recordings to detect artificial residual noise, indicators of change in the short-time Fourier transform array were used as signs of occurrence. A better understanding of what exactly is the trigger, and if there are certain frequencies, thresholds in signal amplitude, or similar requirements that need to be handled in order to avoid the problem is an area to look into.
- **More complex loss function:** The machine learning algorithms in the experiments conducted as part of this thesis have all used mean squared error as the loss function. The challenge with this as presented in the discussion in chapter 6 is that it averages out the optimization. Taking this approach sacrifices precision when it comes to handling complex data. The need for a customized loss function would assumably overcome this problem by granting more control to the analyst.
- **More complex machine learning algorithms:** Improved machine learning algorithms are presented rather often as it has become an area that has a lot of attention related to how it can be applied in a wide variety of use. This thesis approached the problem of artificial residual noise using already established machine learning methods to conduct the experiments. These were chosen because how they have already been tested within the area of audio processing. There are other alternatives out there like Generative Adversarial Networks which are more advanced and have the potential to provide better results.
- **The machine learning framework for audio forensics:** A ambitious task would be to try to develop a method specially adapted for audio forensics. As discussed in this thesis it has set requirements that need to be met. Integrity and reliability are some of these. This will require time and effort which probably would require multiple theses to complete, but like everything else in a project it should be broken down into components that combined makes

up the final product, either as a stand-alone solution or part of something bigger.

Bibliography

[1] J. Zjalic, *Digital Audio Forensics Fundamentals: From Capture to Courtroom*. Focal Press, 2020.

[2] A. Guarino, 'Digital forensics as a big data challenge,' in *ISSE 2013 securing electronic business processes*, Springer, 2013, pp. 197–203.

[3] M. Sakpal, *Smartphone subscriptions worldwide 2016-2027*, 2022. [Online]. Available: `https://www.gartner.com/en/newsroom/press-releases/2022-03-01-4q21-smartphone-market-share`.

[4] S. O'Dea, *Smartphone subscriptions worldwide 2016-2027*, 2022. [Online]. Available: `https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/`.

[5] J. I. James and P. Gladyshev, 'Challenges with automation in digital forensic investigations,' *arXiv preprint arXiv:1303.4498*, 2013.

[6] J. S. Sachs, 'Graphing the voice of terror,' *Popular Science*, pp. 38–43, 2003. [Online]. Available: `https://www.popsci.com/scitech/article/2003-02/graphing-voice-terror/`.

[7] R. C. Maher, 'Audio forensic examination,' *IEEE Signal Processing Magazine*, vol. 26, no. 2, pp. 84–94, 2009.

[8] Y. Ke, A. Li, C. Zheng, R. Peng and X. Li, 'Low-complexity artificial noise suppression methods for deep learning-based speech enhancement algorithms,' *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2021, no. 1, pp. 1–15, 2021.

[9] K. Paliwal, K. Wójcicki and B. Shannon, 'The importance of phase in speech enhancement,' *speech communication*, vol. 53, no. 4, pp. 465–494, 2011.

[10] Y. Zhao, Z.-Q. Wang and D. Wang, 'Two-stage deep learning for noisy-reverberant speech enhancement,' *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 1, pp. 53–62, 2018.

[11] D. S. Williamson, Y. Wang and D. Wang, 'Complex ratio masking for monaural speech separation,' *IEEE/ACM transactions on audio, speech, and language processing*, vol. 24, no. 3, pp. 483–492, 2015.

[12] A. Li, C. Zheng, C. Fan, R. Peng and X. Li, 'A recursive network with dynamic attention for monaural speech enhancement,' *arXiv preprint arXiv:2003.12973*, 2020.

[13] D. Greene, A. L. Hoffmann and L. Stark, 'Better, nicer, clearer, fairer: A critical assessment of the movement for ethical artificial intelligence and machine learning,' in *Proceedings of the 52nd Hawaii international conference on system sciences*, 2019.

[14] S. Lo Piano, 'Ethical principles in machine learning and artificial intelligence: Cases from the field and possible ways forward,' *Humanities and Social Sciences Communications*, vol. 7, no. 1, pp. 1–7, 2020.

[15] *Report of the Committee on Ballistic Acoustics*. Washington, DC: The National Academies Press, 1982. DOI: `10.17226/10264`.

[16] United States District Court for the District of Columbia, *Advisory panel on white house tapes, "the executive office building tape of june 20, 1972: Report on a technical investigation,"* 1974. [Online]. Available: `http://www.aes.org/aeshc/docs/forensic.audio/watergate.tapes.report.pdf`.

[17] D. Quick and K.-K. R. Choo, 'Impacts of increasing volume of digital forensic data: A survey and future research challenges,' *Digital Investigation*, vol. 11, no. 4, pp. 273–294, 2014, ISSN: 1742-2876. DOI: `https://doi.org/10.1016/j.diin.2014.09.002`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S1742287614001066`.

[18] Council of European Union, *Norwegian criminal procedure act§294*, 1981. [Online]. Available: `https://lovdata.no/dokument/NL/lov/1981-05-22-25/KAPITTEL_5-5#KAPITTEL_5-5`.

[19] 'Når kan man gjennomføre lydopptak?' Datatilsynet. (28th Mar. 2020), [Online]. Available: `https://www.datatilsynet.no/personvern-pa-ulike-omrader/overvaking-og-sporing/lydopptak/naar_kan_lydopptak_finne_sted` (visited on 11/05/2022).

[20] '2018 reform of eu data protection rules,' European Commission. (25th May 2018), [Online]. Available: `https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf` (visited on 11/05/2022).

[21] K. Tan, X. Zhang and D. Wang, 'Real-time speech enhancement using an efficient convolutional recurrent network for dual-microphone mobile phones in close-talk scenarios,' in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 5751–5755.

[22] C. D. Motchenbacher, J. A. Connelly *et al.*, *Low-noise electronic system design*. Wiley New York, 1993, vol. 269.

[23] D. P. Ellis and G. E. Poliner, 'Identifyingcover songs' with chroma features and dynamic programming beat tracking,' in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, IEEE, vol. 4, 2007, pp. IV–1429.

[24] B. Logan, 'Mel frequency cepstral coefficients for music modeling,' in *In International Symposium on Music Information Retrieval*, 2000.

[25] R. Bachu, S. Kopparthi, B. Adapa and B. Barkana, 'Separation of voiced and unvoiced using zero crossing rate and energy of the speech signal,' in *American Society for Engineering Education (ASEE) zone conference proceedings*, American Society for Engineering Education, 2008, pp. 1–7.

[26] N. Kehtarnavaz, 'Chapter 7 - frequency domain processing,' in *Digital Signal Processing System Design (Second Edition)*, N. Kehtarnavaz, Ed., Second Edition, Burlington: Academic Press, 2008, pp. 175–196, ISBN: 978-0-12-374490-6. DOI: `https://doi.org/10.1016/B978-0-12-374490-6.00007-6`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/B9780123744906000076`.

[27] E. Schubert and J. Wolfe, 'Does timbral brightness scale with frequency and spectral centroid?' *Acta acustica united with acustica*, vol. 92, no. 5, pp. 820–825, 2006.

[28] J. Park, C. R. Lindberg and F. L. Vernon III, 'Multitaper spectral analysis of high-frequency seismograms,' *Journal of Geophysical Research: Solid Earth*, vol. 92, no. B12, pp. 12 675–12 684, 1987.

[29] I. Kononenko and M. Kukar, *Machine Learning and Data Mining: Introductuion to Principles and Algorithms*. 80 High Street, Sawston, Cambridge CB22 3HJ, UK: Woodhead Publishing, 2007, ISBN: 9781904275213.

[30] S. Lawrence and C. L. Giles, 'Overfitting and neural networks: Conjugate gradient and backpropagation,' in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, IEEE, vol. 1, 2000, pp. 114–119.

[31] C. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, 'Activation functions: Comparison of trends in practice and research for deep learning,' *arXiv preprint arXiv:1811.03378*, 2018.

[32] S. Albawi, T. A. Mohammed and S. Al-Zawi, 'Understanding of a convolutional neural network,' in *2017 international conference on engineering and technology (ICET)*, Ieee, 2017, pp. 1–6.

[33] N. Kalchbrenner, E. Grefenstette and P. Blunsom, 'A convolutional neural network for modelling sentences,' *arXiv preprint arXiv:1404.2188*, 2014.

[34] K. Tan and D. Wang, 'A convolutional recurrent neural network for real-time speech enhancement.,' in *Interspeech*, vol. 2018, 2018, pp. 3229–3233.

[35] R. Fu, Z. Zhang and L. Li, ‘Using lstm and gru neural network methods for traffic flow prediction,’ in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, IEEE, 2016, pp. 324–328.

[36] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, ‘Densely connected convolutional networks,’ in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[37] K. Tan, J. Chen and D. Wang, ‘Gated residual networks with dilated convolutions for monaural speech enhancement,’ *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 1, pp. 189–198, 2018.

[38] K. Tan, J. Chen and D. Wang, ‘Gated residual networks with dilated convolutions for supervised speech separation,’ in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 21–25.

[39] G. Yu, S. Mallat and E. Bacry, ‘Audio denoising by time-frequency block thresholding,’ *IEEE Transactions on Signal processing*, vol. 56, no. 5, pp. 1830–1839, 2008.

[40] M. Souden, J. Benesty and S. Affes, ‘On optimal frequency-domain multichannel linear filtering for noise reduction,’ *IEEE Transactions on audio, speech, and language processing*, vol. 18, no. 2, pp. 260–276, 2009.

[41] A. Rix, J. Beerends, M. Hollier and A. Hekstra, ‘Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs,’ in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 2, 2001, 749–752 vol.2. DOI: `10.1109/ICASSP.2001.941023`.

[42] M. Vondrasek and P. Pollak, ‘Methods for speech snr estimation: Evaluation tool and analysis of vad dependency,’ *Radioengineering*, vol. 14, no. 1, pp. 6–11, 2005.

[43] Y. Hu and P. C. Loizou, ‘Evaluation of objective quality measures for speech enhancement,’ *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 229–238, 2008. DOI: `10.1109/TASL.2007.911054`.

[44] C. H. Taal, R. C. Hendriks, R. Heusdens and J. Jensen, ‘An algorithm for intelligibility prediction of time–frequency weighted noisy speech,’ *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2125–2136, 2011.

[45] C. H. Taal, R. C. Hendriks, R. Heusdens and J. Jensen, ‘A short-time objective intelligibility measure for time-frequency weighted noisy speech,’ in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010, pp. 4214–4217. DOI: `10.1109/ICASSP.2010.5495701`.

[46] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers and G. Weber, 'Common voice: A massively-multilingual speech corpus,' in *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, 2020, pp. 4211–4215.

[47] E. Vincent, N. Bertin, R. Gribonval and F. Bimbot, 'From blind to guided audio source separation: How models and side information can improve the separation of sound,' *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 107–115, 2014. DOI: `10.1109/MSP.2013.2297440`.

[48] J. Salamon, C. Jacoby and J. P. Bello, 'A dataset and taxonomy for urban sound research,' in *22nd ACM International Conference on Multimedia (ACM-MM'14)*, Orlando, FL, USA, 2014, pp. 1041–1044.

[49] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN: 1441412697.

[50] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[51] S. R. Park and J. Lee, *A fully convolutional neural network for speech enhancement*, 2016. DOI: `10.48550/ARXIV.1609.07132`. [Online]. Available: `https://arxiv.org/abs/1609.07132`.

[52] D. M. Allen, 'Mean square error of prediction as a criterion for selecting variables,' *Technometrics*, vol. 13, no. 3, pp. 469–475, 1971.

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

[54] T. Chai and R. R. Draxler, 'Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature,' *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.

Appendix A

CNN layer layout

```
 Layer type                    Output Shape        Param #     Connected to
==================================================================================================
 input_1 InputLayer            [None, 129, 8, 1]  0           []

 zero_padding2d ZeroPadding2D   None, 137, 8, 1   0           ['input_1[0][0]']

 conv2d Conv2D                 None, 129, 1, 18   1296        ['zero_padding2d[0][0]']

 activation Activation         None, 129, 1, 18   0           ['conv2d[0][0]']

 batch_normalization BatchNorm  None, 129, 1, 18  72           ['activation[0][0]'] alization

 conv2d_1 Conv2D               None, 129, 1, 30   2700        ['batch_normalization[0][0]']

 activation_1 Activation       None, 129, 1, 30   0           ['conv2d_1[0][0]']

 batch_normalization_1 BatchNo  None, 129, 1, 30  120          ['activation_1[0][0]'] rmalization

 conv2d_2 Conv2D               None, 129, 1, 8    2160        ['batch_normalization_1[0][0]']

 activation_2 Activation       None, 129, 1, 8    0           ['conv2d_2[0][0]']

 batch_normalization_2 BatchNo  None, 129, 1, 8   32           ['activation_2[0][0]'] rmalization

 max_pooling2d MaxPooling2D    None, 129, 1, 8    0           ['batch_normalization_2[0][0]']

 conv2d_3 Conv2D               None, 129, 1, 18   1296        ['max_pooling2d[0][0]']

 activation_3 Activation       None, 129, 1, 18   0           ['conv2d_3[0][0]']

 batch_normalization_3 BatchNo  None, 129, 1, 18  72           ['activation_3[0][0]'] rmalization

 conv2d_4 Conv2D               None, 129, 1, 30   2700        ['batch_normalization_3[0][0]']

 activation_4 Activation       None, 129, 1, 30   0           ['conv2d_4[0][0]']
```

```
batch_normalization_4 BatchNo  None, 129, 1, 30  120         ['activation_4[0][0]'] rmalization

conv2d_5 Conv2D              None, 129, 1, 8    2160       ['batch_normalization_4[0][0]']

activation_5 Activation      None, 129, 1, 8    0          ['conv2d_5[0][0]']

batch_normalization_5 BatchNo  None, 129, 1, 8   32          ['activation_5[0][0]'] rmalization

conv2d_6 Conv2D              None, 129, 1, 18   1296       ['batch_normalization_5[0][0]']

activation_6 Activation      None, 129, 1, 18   0          ['conv2d_6[0][0]']

batch_normalization_6 BatchNo  None, 129, 1, 18  72          ['activation_6[0][0]'] rmalization

conv2d_7 Conv2D              None, 129, 1, 30   2700       ['batch_normalization_6[0][0]']

activation_7 Activation      None, 129, 1, 30   0          ['conv2d_7[0][0]']

batch_normalization_7 BatchNo  None, 129, 1, 30  120         ['activation_7[0][0]'] rmalization

max_pooling2d_1 MaxPooling2D  None, 129, 1, 30  0          ['batch_normalization_7[0][0]']

conv2d_8 Conv2D              None, 129, 1, 8    2160       ['max_pooling2d_1[0][0]']

activation_8 Activation      None, 129, 1, 8    0          ['conv2d_8[0][0]']

batch_normalization_8 BatchNo  None, 129, 1, 8   32          ['activation_8[0][0]'] rmalization

conv2d_9 Conv2D              None, 129, 1, 18   1296       ['batch_normalization_8[0][0]']

activation_9 Activation      None, 129, 1, 18   0          ['conv2d_9[0][0]']

batch_normalization_9 BatchNo  None, 129, 1, 18  72          ['activation_9[0][0]'] rmalization

conv2d_10 Conv2D             None, 129, 1, 30   2700       ['batch_normalization_9[0][0]']

tf.__operators__.add TFOpLamb  None, 129, 1, 30  0           ['conv2d_10[0][0]', da

activation_10 Activation     None, 129, 1, 30   0          ['tf.__operators__.add[0][0]']

batch_normalization_10 BatchN  None, 129, 1, 30  120         ['activation_10[0][0]'] ormalization

conv2d_11 Conv2D             None, 129, 1, 8    2160       ['batch_normalization_10[0][0]']

activation_11 Activation     None, 129, 1, 8    0          ['conv2d_11[0][0]']

batch_normalization_11 BatchN  None, 129, 1, 8   32          ['activation_11[0][0]'] ormalization

conv2d_12 Conv2D             None, 129, 1, 18   1296       ['batch_normalization_11[0][0]']

activation_12 Activation     None, 129, 1, 18   0          ['conv2d_12[0][0]']

batch_normalization_12 BatchN  None, 129, 1, 18  72          ['activation_12[0][0]'] ormalization
```

```
 conv2d_13 Conv2D             None, 129, 1, 30   2700       ['batch_normalization_12[0][0]']

 tf.__operators__.add_1 TFOpLa  None, 129, 1, 30  0          ['conv2d_13[0][0]', mbda

 activation_13 Activation     None, 129, 1, 30   0          ['tf.__operators__.add_1[0][0]']

 batch_normalization_13 BatchN  None, 129, 1, 30  120        ['activation_13[0][0]'] ormalization

 conv2d_14 Conv2D             None, 129, 1, 8    2160       ['batch_normalization_13[0][0]']

 activation_14 Activation     None, 129, 1, 8    0          ['conv2d_14[0][0]']

 batch_normalization_14 BatchN  None, 129, 1, 8   32         ['activation_14[0][0]'] ormalization

 spatial_dropout2d SpatialDrop  None, 129, 1, 8   0          ['batch_normalization_14[0][0]'] out2D

 conv2d_15 Conv2D             None, 129, 1, 1    1033       ['spatial_dropout2d[0][0]']

 activation_15 Activation     None, 129, 1, 1    0          ['conv2d_15[0][0]']

 batch_normalization_15 BatchN  None, 129, 1, 1   4          ['activation_15[0][0]'] ormalization

 max_pooling2d_2 MaxPooling2D  None, 129, 1, 1   0          ['batch_normalization_15[0][0]']

 spatial_dropout2d_1 SpatialDr  None, 129, 1, 1   0          ['max_pooling2d_2[0][0]'] opout2D

 dense Dense                  None, 129, 1, 1    2          ['spatial_dropout2d_1[0][0]']

 reshape Reshape              None, 129, 1, 1    0          ['dense[0][0]']

==================================================================================================
Total params: 32,939
Trainable params: 32,377
Non-trainable params: 562
__________________________________________________________________________________________________
```

Appendix B

CRNN layer layout

```
 Layer type                    Output Shape        Param #     Connected to
===================================================================================================
 input_1 InputLayer            [None, 129, 8, 1]  0           []

 zero_padding2d ZeroPadding2D  None, 137, 8, 1   0           ['input_1[0][0]']

 conv2d Conv2D                 None, 129, 1, 18   1296       ['zero_padding2d[0][0]']

 activation Activation         None, 129, 1, 18   0          ['conv2d[0][0]']

 batch_normalization BatchNorm  None, 129, 1, 18  72          ['activation[0][0]'] alization

 conv2d_1 Conv2D               None, 129, 1, 30   2700       ['batch_normalization[0][0]']

 activation_1 Activation       None, 129, 1, 30   0          ['conv2d_1[0][0]']

 batch_normalization_1 BatchNo  None, 129, 1, 30  120         ['activation_1[0][0]'] rmalization

 conv2d_2 Conv2D               None, 129, 1, 8    2160       ['batch_normalization_1[0][0]']

 activation_2 Activation       None, 129, 1, 8    0          ['conv2d_2[0][0]']

 batch_normalization_2 BatchNo  None, 129, 1, 8   32          ['activation_2[0][0]'] rmalization

 max_pooling2d MaxPooling2D    None, 129, 1, 8    0          ['batch_normalization_2[0][0]']

 conv2d_3 Conv2D               None, 129, 1, 18   1296       ['max_pooling2d[0][0]']

 activation_3 Activation       None, 129, 1, 18   0          ['conv2d_3[0][0]']

 batch_normalization_3 BatchNo  None, 129, 1, 18  72          ['activation_3[0][0]'] rmalization

 conv2d_4 Conv2D               None, 129, 1, 30   2700       ['batch_normalization_3[0][0]']

 activation_4 Activation       None, 129, 1, 30   0          ['conv2d_4[0][0]']
```

```
batch_normalization_4 BatchNo  None, 129, 1, 30  120         ['activation_4[0][0]'] rmalization

conv2d_5 Conv2D              None, 129, 1, 8    2160       ['batch_normalization_4[0][0]']

activation_5 Activation      None, 129, 1, 8    0          ['conv2d_5[0][0]']

batch_normalization_5 BatchNo  None, 129, 1, 8   32          ['activation_5[0][0]'] rmalization

conv2d_6 Conv2D              None, 129, 1, 18   1296       ['batch_normalization_5[0][0]']

activation_6 Activation      None, 129, 1, 18   0          ['conv2d_6[0][0]']

batch_normalization_6 BatchNo  None, 129, 1, 18  72          ['activation_6[0][0]'] rmalization

conv2d_7 Conv2D              None, 129, 1, 30   2700       ['batch_normalization_6[0][0]']

activation_7 Activation      None, 129, 1, 30   0          ['conv2d_7[0][0]']

batch_normalization_7 BatchNo  None, 129, 1, 30  120         ['activation_7[0][0]'] rmalization

max_pooling2d_1 MaxPooling2D  None, 129, 1, 30  0          ['batch_normalization_7[0][0]']

conv2d_8 Conv2D              None, 129, 1, 8    2160       ['max_pooling2d_1[0][0]']

activation_8 Activation      None, 129, 1, 8    0          ['conv2d_8[0][0]']

batch_normalization_8 BatchNo  None, 129, 1, 8   32          ['activation_8[0][0]'] rmalization

conv2d_9 Conv2D              None, 129, 1, 18   1296       ['batch_normalization_8[0][0]']

activation_9 Activation      None, 129, 1, 18   0          ['conv2d_9[0][0]']

batch_normalization_9 BatchNo  None, 129, 1, 18  72          ['activation_9[0][0]'] rmalization

conv2d_10 Conv2D             None, 129, 1, 30   2700       ['batch_normalization_9[0][0]']

tf.__operators__.add TFOpLamb  None, 129, 1, 30  0           ['conv2d_10[0][0]', da

activation_10 Activation     None, 129, 1, 30   0          ['tf.__operators__.add[0][0]']

batch_normalization_10 BatchN  None, 129, 1, 30  120         ['activation_10[0][0]'] ormalization

conv2d_11 Conv2D             None, 129, 1, 8    2160       ['batch_normalization_10[0][0]']

activation_11 Activation     None, 129, 1, 8    0          ['conv2d_11[0][0]']

batch_normalization_11 BatchN  None, 129, 1, 8   32          ['activation_11[0][0]'] ormalization

conv2d_12 Conv2D             None, 129, 1, 18   1296       ['batch_normalization_11[0][0]']

activation_12 Activation     None, 129, 1, 18   0          ['conv2d_12[0][0]']

batch_normalization_12 BatchN  None, 129, 1, 18  72          ['activation_12[0][0]'] ormalization
```

```
 conv2d_13 Conv2D             None, 129, 1, 30   2700        ['batch_normalization_12[0][0]']

 tf.__operators__.add_1 TFOpLa  None, 129, 1, 30  0           ['conv2d_13[0][0]', mbda

 activation_13 Activation     None, 129, 1, 30   0           ['tf.__operators__.add_1[0][0]']

 batch_normalization_13 BatchN  None, 129, 1, 30  120         ['activation_13[0][0]'] ormalization

 conv2d_14 Conv2D             None, 129, 1, 8    2160        ['batch_normalization_13[0][0]']

 activation_14 Activation     None, 129, 1, 8    0           ['conv2d_14[0][0]']

 batch_normalization_14 BatchN  None, 129, 1, 8   32          ['activation_14[0][0]'] ormalization

 spatial_dropout2d SpatialDrop  None, 129, 1, 8   0           ['batch_normalization_14[0][0]'] out2D

 conv2d_15 Conv2D             None, 129, 1, 1    1033        ['spatial_dropout2d[0][0]']

 activation_15 Activation     None, 129, 1, 1    0           ['conv2d_15[0][0]']

 batch_normalization_15 BatchN  None, 129, 1, 1   4           ['activation_15[0][0]'] ormalization

 max_pooling2d_2 MaxPooling2D  None, 129, 1, 1   0           ['batch_normalization_15[0][0]']

 spatial_dropout2d_1 SpatialDr  None, 129, 1, 1   0           ['max_pooling2d_2[0][0]'] opout2D

 flatten Flatten              None, 129          0           ['spatial_dropout2d_1[0][0]']

 reshape Reshape              None, 129, 1       0           ['flatten[0][0]']

 bidirectional Bidirectional  None, 129, 258     102168      ['reshape[0][0]']

 batch_normalization_16 BatchN  None, 129, 258    1032        ['bidirectional[0][0]'] ormalization

 spatial_dropout1d SpatialDrop  None, 129, 258    0           ['batch_normalization_16[0][0]'] out1D

 bidirectional_1 Bidirectional  None, 129, 258    301086      ['spatial_dropout1d[0][0]']

 batch_normalization_17 BatchN  None, 129, 258    1032        ['bidirectional_1[0][0]'] ormalization

 spatial_dropout1d_1 SpatialDr  None, 129, 258    0           ['batch_normalization_17[0][0]'] opout1D

 dense Dense                  None, 129, 129     33411       ['spatial_dropout1d_1[0][0]']

 activation_16 Activation     None, 129, 129     0           ['dense[0][0]']

 batch_normalization_18 BatchN  None, 129, 129    516         ['activation_16[0][0]'] ormalization

 dense_1 Dense                None, 129, 1       130         ['batch_normalization_18[0][0]']

 reshape_1 Reshape            None, 129, 1, 1    0           ['dense_1[0][0]']

==================================================================================================
Total params: 472,312
```

```
Trainable params: 470,460
Non-trainable params: 1,852
```

Appendix C

DCN layer layout

```
 Layer type                    Output Shape         Param #     Connected to
==================================================================================================
 input_1 InputLayer            [None, 129, 8, 1]  0           []

 zero_padding2d ZeroPadding2D  None, 137, 8, 1   0           ['input_1[0][0]']

 conv2d Conv2D                 None, 129, 1, 32   2304        ['zero_padding2d[0][0]']

 batch_normalization BatchNorm  None, 129, 1, 32  128         ['conv2d[0][0]'] alization

 activation Activation         None, 129, 1, 32   0           ['batch_normalization[0][0]']

 conv2d_1 Conv2D               None, 129, 1, 32   1024        ['activation[0][0]']

 spatial_dropout2d SpatialDrop  None, 129, 1, 32  0           ['conv2d_1[0][0]'] out2D

 concatenate Concatenate       None, 129, 1, 64   0           ['conv2d[0][0]',
                                                                  'spatial_dropout2d[0][0]']

 batch_normalization_1 BatchNo  None, 129, 1, 64  256         ['concatenate[0][0]'] rmalization

 activation_1 Activation       None, 129, 1, 64   0           ['batch_normalization_1[0][0]']

 conv2d_2 Conv2D               None, 129, 1, 32   2048        ['activation_1[0][0]']

 spatial_dropout2d_1 SpatialDr  None, 129, 1, 32  0           ['conv2d_2[0][0]'] opout2D

 concatenate_1 Concatenate     None, 129, 1, 96   0           ['conv2d[0][0]',
                                                                  'spatial_dropout2d[0][0]',
                                                                  'spatial_dropout2d_1[0][0]']

 batch_normalization_2 BatchNo  None, 129, 1, 96  384         ['concatenate_1[0][0]'] rmalization

 activation_2 Activation       None, 129, 1, 96   0           ['batch_normalization_2[0][0]']

 conv2d_3 Conv2D               None, 129, 1, 32   3072        ['activation_2[0][0]']
 spatial_dropout2d_2 SpatialDr  None, 129, 1, 32  0           ['conv2d_3[0][0]'] opout2D

 batch_normalization_3 BatchNo  None, 129, 1, 32  128         ['spatial_dropout2d_2[0][0]'] rmalization

 activation_3 Activation      None, 129, 1, 32   0           ['batch_normalization_3[0][0]']

 conv2d_4 Conv2D              None, 129, 1, 32   1024        ['activation_3[0][0]']

 spatial_dropout2d_3 SpatialDr  None, 129, 1, 32  0           ['conv2d_4[0][0]'] opout2D

 concatenate_2 Concatenate    None, 129, 1, 64   0           ['spatial_dropout2d_2[0][0]',
                                                                  'spatial_dropout2d_3[0][0]']

 batch_normalization_4 BatchNo  None, 129, 1, 64  256         ['concatenate_2[0][0]'] rmalization

 activation_4 Activation      None, 129, 1, 64   0           ['batch_normalization_4[0][0]']

 conv2d_5 Conv2D              None, 129, 1, 32   2048        ['activation_4[0][0]']

 spatial_dropout2d_4 SpatialDr  None, 129, 1, 32  0           ['conv2d_5[0][0]'] opout2D

 concatenate_3 Concatenate    None, 129, 1, 96   0           ['spatial_dropout2d_2[0][0]',
                                                                  'spatial_dropout2d_3[0][0]',
                                                                  'spatial_dropout2d_4[0][0]']

 batch_normalization_5 BatchNo  None, 129, 1, 96  384         ['concatenate_3[0][0]'] rmalization

 activation_5 Activation      None, 129, 1, 96   0           ['batch_normalization_5[0][0]']

 flatten Flatten              None, 12384        0           ['activation_5[0][0]']

 dense Dense                  None, 129          1597665     ['flatten[0][0]']

 reshape Reshape              None, 129, 1, 1    0           ['dense[0][0]']

==================================================================================================
Total params: 1,610,721
Trainable params: 1,609,953
Non-trainable params: 768
__________________________________________________________________________________________________
```

Appendix D

DARCN layer layout

```
 Layer type                    Output Shape         Param #     Connected to
==================================================================================================
 input_1 InputLayer            [None, 129, 8, 1]  0           []

 conv2d_32 Conv2D              None, 129, 8, 64   640         ['input_1[0][0]']

 activation_28 Activation      None, 129, 8, 64   0           ['conv2d_32[0][0]']

 conv2d_33 Conv2D              None, 129, 8, 64   36928       ['activation_28[0][0]']

 activation_29 Activation      None, 129, 8, 64   0           ['conv2d_33[0][0]']

 conv2d_34 Conv2D              None, 129, 8, 1    65          ['activation_29[0][0]']

 dense_1 Dense                 None, 129, 8, 129  258         ['conv2d_34[0][0]']

 dense Dense                   None, 129, 8, 129  258         ['conv2d_34[0][0]']

 activation_30 Activation      None, 129, 8, 129  0           ['dense_1[0][0]']

 batch_normalization_24 BatchN  None, 129, 8, 129  516         ['dense[0][0]'] ormalization

 batch_normalization_25 BatchN  None, 129, 8, 129  516         ['activation_30[0][0]'] ormalization

 tf.math.multiply TFOpLambda  None, 129, 8, 129  0           ['batch_normalization_24[0][0]',
                                                                  'batch_normalization_25[0][0]']

 dense_3 Dense                 None, 129, 8, 129  16770       ['tf.math.multiply[0][0]']

 dense_2 Dense                 None, 129, 8, 129  16770       ['tf.math.multiply[0][0]']

 activation_31 Activation      None, 129, 8, 129  0           ['dense_3[0][0]']

 batch_normalization_26 BatchN  None, 129, 8, 129  516         ['dense_2[0][0]'] ormalization

 batch_normalization_27 BatchN  None, 129, 8, 129  516         ['activation_31[0][0]'] ormalization
 tf.math.multiply_1 TFOpLambda  None, 129, 8, 129  0          ['batch_normalization_26[0][0]',

 dense_5 Dense                 None, 129, 8, 129  16770      ['tf.math.multiply_1[0][0]']

 dense_4 Dense                 None, 129, 8, 129  16770      ['tf.math.multiply_1[0][0]']

 activation_32 Activation      None, 129, 8, 129  0          ['dense_5[0][0]']

 batch_normalization_28 BatchN  None, 129, 8, 129  516        ['dense_4[0][0]'] ormalization

 batch_normalization_29 BatchN  None, 129, 8, 129  516        ['activation_32[0][0]'] ormalization

 tf.__operators__.add TFOpLamb  None, 129, 8, 129  0          ['tf.math.multiply[0][0]', da

 tf.math.multiply_2 TFOpLambda  None, 129, 8, 129  0          ['batch_normalization_28[0][0]',

 tf.__operators__.add_1 TFOpLa  None, 129, 8, 129  0          ['tf.__operators__.add[0][0]', mbda

 conv2d_35 Conv2D              None, 129, 8, 1    130        ['tf.__operators__.add_1[0][0]']

 conv2d_68 Conv2D              None, 129, 8, 64   640        ['conv2d_35[0][0]']

 activation_61 Activation      None, 129, 8, 64   0          ['conv2d_68[0][0]']

 conv2d_69 Conv2D              None, 129, 8, 64   36928      ['activation_61[0][0]']

 activation_62 Activation      None, 129, 8, 64   0          ['conv2d_69[0][0]']

 max_pooling2d_7 MaxPooling2D  None, 64, 4, 64    0          ['activation_62[0][0]']

 flatten Flatten               None, 16384        0          ['max_pooling2d_7[0][0]']

 dense_6 Dense                 None, 129          2113665    ['flatten[0][0]']

 reshape Reshape               None, 129, 1, 1    0          ['dense_6[0][0]']

==================================================================================================
Total params: 2,259,688
Trainable params: 2,258,140
Non-trainable params: 1,548
__________________________________________________________________________________________________
```

Appendix E

GRN layer layout

```
 Layer type                    Output Shape         Param #     Connected to
==================================================================================================
 input_1 InputLayer            [None, 129, 8, 1]  0           []

 zero_padding2d ZeroPadding2D  None, 137, 8, 1   0           ['input_1[0][0]']

 spatial_dropout2d SpatialDrop  None, 137, 8, 1   0           ['zero_padding2d[0][0]'] out2D

 conv2d Conv2D                 None, 137, 8, 8    208         ['spatial_dropout2d[0][0]']

 activation Activation         None, 137, 8, 8    0           ['conv2d[0][0]']

 batch_normalization BatchNorm  None, 137, 8, 8   32          ['activation[0][0]'] alization

 conv2d_1 Conv2D               None, 137, 8, 8    1608        ['batch_normalization[0][0]']

 activation_1 Activation       None, 137, 8, 8    0           ['conv2d_1[0][0]']

 batch_normalization_1 BatchNo  None, 137, 8, 8   32          ['activation_1[0][0]'] rmalization

 conv2d_2 Conv2D               None, 137, 8, 16   3216        ['batch_normalization_1[0][0]']

 activation_2 Activation       None, 137, 8, 16   0           ['conv2d_2[0][0]']

 batch_normalization_2 BatchNo  None, 137, 8, 16  64          ['activation_2[0][0]'] rmalization

 conv2d_3 Conv2D               None, 137, 8, 16   6416        ['batch_normalization_2[0][0]']

 activation_3 Activation       None, 137, 8, 16   0           ['conv2d_3[0][0]']

 batch_normalization_3 BatchNo  None, 137, 8, 16  64          ['activation_3[0][0]'] rmalization

 conv1d Conv1D                 None, 137, 8, 64   1088        ['batch_normalization_3[0][0]']

 activation_4 Activation       None, 137, 8, 64   0           ['conv1d[0][0]']
```

```
batch_normalization_4 BatchNo  None, 137, 8, 64  256         ['activation_4[0][0]'] rmalization

conv1d_4 Conv1D              None, 137, 8, 64   4160       ['batch_normalization_4[0][0]']

activation_8 Activation      None, 137, 8, 64   0          ['conv1d_4[0][0]']

batch_normalization_8 BatchNo  None, 137, 8, 64  256         ['activation_8[0][0]'] rmalization

conv1d_8 Conv1D              None, 137, 8, 64   4160       ['batch_normalization_8[0][0]']

activation_12 Activation     None, 137, 8, 64   0          ['conv1d_8[0][0]']

batch_normalization_5 BatchNo  None, 137, 8, 64  256         ['batch_normalization_4[0][0]'] rmalization

batch_normalization_6 BatchNo  None, 137, 8, 64  256         ['batch_normalization_4[0][0]'] rmalization

batch_normalization_9 BatchNo  None, 137, 8, 64  256         ['batch_normalization_8[0][0]'] rmalization

batch_normalization_10 BatchN  None, 137, 8, 64  256         ['batch_normalization_8[0][0]'] ormalization

batch_normalization_12 BatchN  None, 137, 8, 64  256         ['activation_12[0][0]'] ormalization

tf.math.multiply TFOpLambda  None, 137, 8, 64   0          ['batch_normalization_5[0][0]',
                                                                'batch_normalization_6[0][0]']

tf.math.multiply_1 TFOpLambda  None, 137, 8, 64  0           ['batch_normalization_9[0][0]',

batch_normalization_13 BatchN  None, 137, 8, 64  256         ['batch_normalization_12[0][0]'] ormalization

batch_normalization_14 BatchN  None, 137, 8, 64  256         ['batch_normalization_12[0][0]'] ormalization

conv1d_3 Conv1D              None, 137, 8, 64   4160       ['tf.math.multiply[0][0]']

conv1d_7 Conv1D              None, 137, 8, 64   4160       ['tf.math.multiply_1[0][0]']

tf.math.multiply_2 TFOpLambda  None, 137, 8, 64  0           ['batch_normalization_13[0][0]',

activation_7 Activation      None, 137, 8, 64   0          ['conv1d_3[0][0]']

activation_11 Activation     None, 137, 8, 64   0          ['conv1d_7[0][0]']

conv1d_11 Conv1D             None, 137, 8, 64   4160       ['tf.math.multiply_2[0][0]']

batch_normalization_7 BatchNo  None, 137, 8, 64  256         ['activation_7[0][0]'] rmalization

batch_normalization_11 BatchN  None, 137, 8, 64  256         ['activation_11[0][0]'] ormalization

activation_15 Activation     None, 137, 8, 64   0          ['conv1d_11[0][0]']

tf.__operators__.add TFOpLamb  None, 137, 8, 64  0           ['batch_normalization_7[0][0]', da

batch_normalization_15 BatchN  None, 137, 8, 64  256         ['activation_15[0][0]'] ormalization

tf.__operators__.add_1 TFOpLa  None, 137, 8, 64  0           ['tf.__operators__.add[0][0]', mbda
```

```
 conv1d_12 Conv1D             None, 137, 8, 64   4160       ['tf.__operators__.add_1[0][0]']

 activation_16 Activation     None, 137, 8, 64   0          ['conv1d_12[0][0]']

 batch_normalization_16 BatchN  None, 137, 8, 64  256        ['activation_16[0][0]'] ormalization

 conv1d_13 Conv1D             None, 137, 8, 64   4160       ['batch_normalization_16[0][0]']

 activation_17 Activation     None, 137, 8, 64   0          ['conv1d_13[0][0]']

 batch_normalization_17 BatchN  None, 137, 8, 64  256        ['activation_17[0][0]'] ormalization

 conv1d_14 Conv1D             None, 137, 8, 32   2080       ['batch_normalization_17[0][0]']

 activation_18 Activation     None, 137, 8, 32   0          ['conv1d_14[0][0]']

 batch_normalization_18 BatchN  None, 137, 8, 32  128        ['activation_18[0][0]'] ormalization

 flatten Flatten              None, 35072        0          ['batch_normalization_18[0][0]']

 dense Dense                  None, 129          4524417    ['flatten[0][0]']

 reshape Reshape              None, 129, 1, 1    0          ['dense[0][0]']

==================================================================================================
Total params: 4,572,057
Trainable params: 4,570,105
Non-trainable params: 1,952
```