Utrecht University
Faculty of Science
Master Artificial Intelligence
7th of February, 2022

Speech Detection for noisy audio files

Daniel Hamandouche
6988555

Supervisors:
Dr. Aleksei Nazarov
Dr. Heysem Kaya

Abstract

Voice activity detection is the challenge to detect the presence or absence of speech in audio files. Different obstructions, such as noisy environments, make this challenge tougher. A noisy environment is, in that case, defined as a recording, which has a low signal-to-noise ratio (SNR) [1].

This thesis aims to detect speech in very noisy environments with an SNR of -10dB and lower. As current deep learning methods [2, 3, 4] were not designed for these very noisy environments and, thus, do not perform well, a new model is adapted to the task. Wav2vec2.0 [5] was created for speech recognition and uses raw audio as input. This thesis adopted and finetuned Wav2vec2.0 for speech detection and extended it by using spectrograms as a second input. The resulting model is compared to two existing speech detection models, one using an architecture based on Lenet-5 [2] and one using a U-net-shaped architecture [6]. All three models are tested on the QUT-NOISE-TIMIT dataset. The results show that the Wav2vec2.0 downstream model performs best on all noise levels. Wav2vec2.0 had the lowest half-total error rate for high noise with 7.71% on a signal-to-noise ratio of -10dB and 18.56% for an SNR of -15dB. All models missed at least 40% of speech for higher noise, so no model is stable.

Furthermore, a subquestion investigated whether the results are beneficial for follow-up tasks. For that, the predictions of each model were used as a pre-processing step by removing all segments without speech.

The results show that Wav2vec2.0 not only improves speech detection for high noise environments, but this improvement also affects speech emotion recognition in these environments. Additionally, this thesis shows that self-supervised learning methods as well as using raw audio are beneficial to the task of speech detection.

Acknowledgement

I would like to thank the group of audio experts from the Dutch Police, who helped me with feedback as well as helped me solve issues related to audio processing and feature extraction. They were also a great help pinpointing the problem when the model training resulted in very fluctuating evaluation accuracy.

I especially would like to thank Riwish and Judith, who were my supervisors from the Dutch Police. They helped me every week with valuable tips for programming the models and writing the thesis. Additionally, they gave me great insights on improving the draft of this thesis.

I also would like to thank my supervisors from Utrecht University, Dr. Aleksei Nazarov and Dr. Heysem Kaya, who supported me during each step of the thesis with helpful comments on my drafts as well as insights on how I can improve the models. Thank you especially to Dr. Aleksei Nazarov, who provided many great tips during the weekly meetings and interesting insights into phonology.

Contents

Chapter 1

Introduction

Speech processing is used in many areas, such as virtual voice assistants, telecommunication, and information extraction. Voice activity detection (VAD), also known as speech detection, is a part of speech processing and works by assigning a label, either speech or no speech, to audio segments. Most virtual assistants such as Siri and Alexa [7] rely on voice activity detection and wake-up-word detection to only react when someone speaks to them. For telecommunication, VAD and speech enhancement techniques are used to reduce the background noises to improve the audio quality [8]. Finally, information can, for instance, be extracted from audio or video by automatic transcription of the speech [9].

VAD is usually processed as the first step to reduce the computation of the succeeding methods, such as recognizing the speech or analyzing the emotions, by only keeping the fragments of the audio that contain voice [1].

To improve these succeeding systems, having effective VAD methods is crucial. As audio files are usually filled with varying background noises, it is essential that the detection methods do not pick up on random background noise but only on the speech. Therefore, VAD methods are required to be very precise, making little to no error, while also being lightweight, requiring little resources [1]. This thesis aims to adapt a speech recognition model to VAD to improve speech detection in highly noisy environments.

This thesis is done in cooperation with the Dutch Police. The research, therefore, also aligns with their interest in analyzing noisy audio files recorded in interrogations, interviews, and in-car conversations. The challenge the police faces is to use the parts classified as speech as evidence. That also means that it needs to be ensured that no evidence is missing.

Since the introduction of a deep belief network by Zhang and Wu [10] for VAD in 2012, deep learning models have been the standard for detecting speech (having improved over threshold-based

models) [10, 11]. One improvement of deep learning models over the previous probabilistic models is that they can work with more types of noises, as they allow using multiple or more complex features. Models using simple features might have problems with certain kinds of noises where the features applied to the noise are too similar to speech [11, 12].

However, even deep learning models struggle with a very low signal-to-noise ratio (SNR). This ratio describes the level of the voice signal compared to the noise level. With a higher noise level, many features are not applicable anymore to detect speech because the features show similar values for speech and noise [4, 13]. The lowest SNR for which the models work well has been lowered in the last ten years. In the first machine learning models [14], an SNR of 5dB was the lowest signal ratio with satisfying results. The latest models can handle an SNR of -5 to -10dB [4].

With louder noise, simply extracting the features is not sufficient anymore. Hidayat et al. [15] show in their paper how the accuracy of using MFCC, one of the most common features used in speech processing, is drastically reduced with noise when used in a speech recognition model. One strategy is to denoise the audio before extracting the feature [15, 16, 17]. Alternatively, a broader context is used to represent speech better. Examples are the Adaptive Context Attention Model [4], which uses attention to incorporate surrounding audio segments, and U-net [6], which uses a U-shaped architecture to include a high-level analysis of several segments at once and a detailed analysis of each segment itself.

To improve the accuracy of detecting very noisy audio containing an SNR of below -5dB, Wav2vec 2.0, a state-of-the-art model in speech recognition, will be adapted to VAD. Using Wav2vec[1] has several benefits. Similar to ACAM [4], it uses attention to model the connections between audio segments. However, it uses several transformer blocks for that, so instead of using one encoder-decoder network as ACAM does, it uses several stacks of them with multiple attention heads per stack, which should give Wav2vec a significant advantage [5]. Additionally, contrary to most VAD networks, it uses raw audio instead of extracting some features. As the effectivity of features diminishes with higher noise, having a model that can handle the raw audio input and extract the best-performing features[2] itself will ensure the best results [5]. Lastly, Wav2vec does not need to learn a representation of what speech means for the specific SNR. It is already pre-trained to have an excellent contextual representation of letters and words and can be trained to represent speech in general. Once it finds a similar embedding in noisy audio, it can assume to have found speech.

In the next chapter, the research question of this thesis will be specified. Afterwards, chapters 3 and 4, give the required background for understanding this thesis. Chapter 3 focuses on the background regarding speech processing and Chapter 4 explains the basic information for the deep learning models used. Chapter 5 then combines these two to explain shortly the history of using deep

[1] In this thesis, Wav2vec2.0 will be used, but it will be denoted as Wav2vec

[2] Features here are just data points that seem to have the high influence on the result, not any manually extractable feature such as MFCC

learning for VAD as well as the technical background of the three models used in this thesis. The first two, CNN-VAD (Section 5.2.1) and U-net (Section 5.2.2), were taken from literature to get a representative implementation. The third one, Wav2vec2.0 (Section 5.2.3) is the focus of this thesis. The data and methodology are described in Chapter 6 and 7, respectively. The latter contains the details about the features used and pre and post-processing steps. In Chapter 8, the results for different noise challenges for all three models are reported, and in the discussion, Chapter 9, they are compared to each other. Lastly, the conclusion and an outlook for further research are given in Chapter 10.

Chapter 2

Research Question

As mentioned in the introduction, current speech detection models [2, 3, 4] work well for SNR levels of -5dB and above. However, below that, the results are significantly worse, and very little literature analyzes very noisy audio.

The main goal of this thesis is to assess whether speech detection is reliable in very noisy environments with an SNR of -10dB and lower. Additionally, this thesis tries to improve the results for these SNR levels. As very little research is done on this noisy audio, the goal of this thesis is also to introduce a new model and new model input (raw audio) to the task to bring this research field forward. This could benefit many AI solutions such as digital voice assistants but also AI systems used in noisy environments.

As this research is conducted in collaboration with the Dutch Police, their practical challenges are aligned with the research objective. The police struggles with many recordings, such as from observations that have these high noises. These need to be analyzed manually, as no system can reliably classify the audio files correctly. Next to the interest of the police, being able to classify speech in these noises can also improve speech recognition for recordings taken in bars, festivals, and other crowded events.

The exact research question is, therefore: Can, during the course of this thesis, the accuracy of Voice Activity Detection be improved for high noise environments with an SNR of -10dB and -15dB to above 90% and 75% accuracy, respectively?

The corresponding subquestion is: Does improving the VAD to the accuracies mentioned above increase the average recall of speech emotion recognition in noisy environments compared to previous VAD methods by at least 5%?

The research questions were chosen to be achievable while also ensuring that the model is good enough to work reliably in different environments. For the dataset used in this thesis, around 53%

is classified as no speech. Therefore, having an accuracy of over 75% shows that the model learned a great representation of speech. 5% improvement for speech emotion recognition was chosen, as existing VAD models are already quite good, so removing only a few more non-speech segments might not affect emotion recognition that much.

To answer the main question, three different models will be used and compared in this thesis, namely CNN-VAD [2], U-net [6] and Wav2vec [5]. These models were chosen to compare different approaches and features. At least for speech recognition tasks, features such as MFCCs have shown reduced effectiveness with lower SNR, even though no results could be found for SNR levels of below -5dB [15]. Wav2vec could, therefore, be a good solution, as it uses raw audio instead of manually extracted features. It is expected that Wav2vec 2.0 can be trained on detecting speech in very low SNR environments and perform well even when the other two models, using commonly extracted features, fail.

For recognizing emotions from speech, a separate model will be created. The architecture is similar to the Wav2vec model, as both require the same representation of the noisy speech. However, the downstream model will be fine-tuned to the emotion recognition task to ensure the best performance. The research goal is to determine if the results of VAD benefit the follow-up tasks. If it is possible to detect speech, but the results cannot be used or do not improve the speech or emotion recognition, then it is questionable whether the results are useful for research. The results would still aid the police, as most of their recordings are several hours long, so reducing that to only the important minutes would help the person listening to it.

Chapter 3

Background on Speech

Speech processing is a wide research area focused on studying speech signals and creating better tools to recognize and analyze them. The main research categories [18, 19] include:

- Automatic speech recognition analyses the linguistic content of the speech, such as the words spoken.
- Speaker state and trait recognition focuses on identifying different information such as gender, age, personality traits, language, and emotion of the speaker.
- Speech synthesis aims to generate speech synthetically, for example, from written text [18, 19].

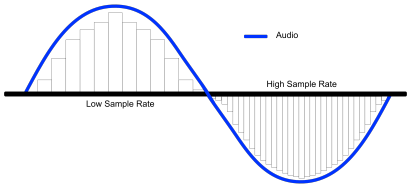
3.1 Voice Activity Detection

Pre-processing can be performed on the speech signal to improve the recognition of speech and speakers. The first improvement is speech enhancement. This research topic focuses on enhancing the speech signal by reducing the effect of noise, and other speech impairments like echo and poor microphone quality [18, 19].

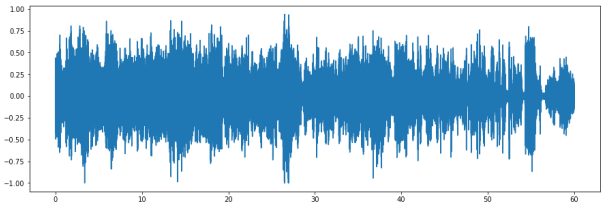
The second improvement is VAD, the main topic of this thesis. The goal of VAD is to detect accurately where speech is present in a given audio file and where not. That is done by labeling each segment of the audio either as speech or no speech and then using these labels to filter out the audio parts containing speech. VAD has the advantage that the following models, such as speech recognition, only encounter those audio signal segments that include speech, rather than being forced to randomly guess a fitting representation or word for a segment with only noise. With these segments gone, the model gets more stable and can focus more on learning the actual words [18, 19].

3.2 Format of the Speech Signal

First, a small introduction to sound signals is given to define what is meant with speech. A sound signal is the variation of air pressure, which happens in waves. When these sound waves reach the human ear, the eardrum begins to vibrate. The brain can then interpret this vibration as the sound we hear [20]. For speech processing, the sound is captured by a microphone and then converted to the digital form instead. An analog to digital converter is used for this conversion. This converter chooses equally distant discrete samples of the continuous analog audio. This selection is known as sampling. The sampling rate used then defines the number of samples per second. For example, for the QUT-NOISE-TIMIT [21] dataset, a sampling rate of 16000Hz is given, which means 16000 data points are available for each second of audio. A visual representation of different sampling rates is shown in Figure 3.1a and an example from the dataset is shown in Figure 3.1b. Additionally, quantization converts the continuous amplitude into discrete values by mapping similar amplitudes to the same digital value. There can also occur some errors in the transformation from analog to digital audio, which this thesis will not go into [22, 23, 24]. The audio files will be taken in digital form as the baseline.



(a) Example of the difference between a low and high sampling rate [25].



(b) Example of a noisy audio file. Each vertical line is one sample taken with 960001 lines in total. The noise of this file is -10dB.

Figure 3.1: These two figures show examples of an audio signal. (a) is an example of different sampling rates. (b) shows an audio file used in this thesis with 16000Hz as sampling rate.

3.3 Speech

Speech is then defined as these parts of the digital audio file where a person is actively speaking. Speaking is hereby meant using the auditory-perceptual definition [26]. That means only when a human listener can perceive the acoustic signature of the human vocal tract (perceive at least one person speaking), it is counted as speaking.

To distinguish the presence or absence of speech, each sample is labeled either 1 for speech or 0 for no speech during the creation of the data sets. These labels will not be perfect, as the person is taking short breaks during the speech, for instance, between phrases. Therefore, it might be that multiple consecutive samples have no actual speech but are still classified as speech.

3.4 Signal versus Noise

Recordings of speech practically always contain other acoustic components next to the acoustic signature of the vocal tract. These other components will be referred to as background noise, and it is important to quantify how strong their presence in the recording is. This quantification is known as the signal-to-noise ratio. As the name says, it measures the strength of the signal itself relative to the background noise. The SNR is usually given in decibels (dB), with positive values indicating that the signal is stronger than the noise and negative values indicating higher noise. If the SNR is zero, the signal level is the same as the noise. Decibels are used as the unit, as they are mathematically defined as $dB = 10 * log_{10}(signal1/signal2)$ [27, 28].

3.5 Windowing

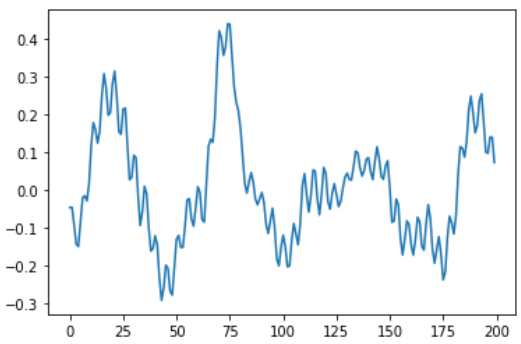
Due to their short lengths, looking at single samples is not that helpful, but together with adjacent samples, they can be quite powerful. To detect when speech occurs, it is therefore important to split the audio into short segments called frames [29]. Ideally, each frame contains exactly one phoneme, the smallest linguistic unit of a language that cannot be substituted for another unit without changing the meaning. It can be, for example, a consonant or short or long vowels. In total, there are 44 distinctive phonemes in English speech [30].

Splitting the audio into small frames is called windowing. As it is impossible to always separate the audio for each phoneme due to temporal overlap between acoustic features of different phonemes, the audio is usually split such that the properties of the speech do not change within one frame. This is also important for applying the Fourier transformation explained in Section 3.6 [31].

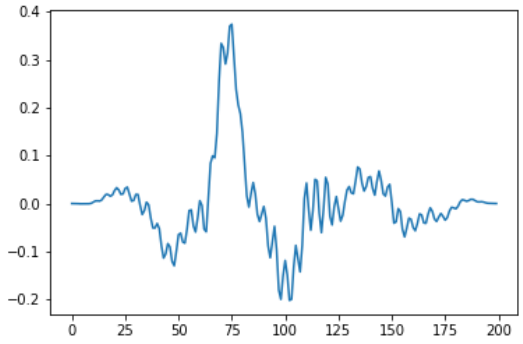
To avoid losing any information at the border of the frames, overlapping is used. With overlapping, each frame contains parts of the previous and parts of the next frame. Commonly, 50% of each frame overlaps with the adjacent ones [31].

Due to the overlap, the borders of a frame can start at any amplitude and are thus not in line with the end of the previous frame. As that could influence the frame properties by, for example, focusing too much on the beginning, which is already covered by the previous frame, windowing functions are applied to ensure that the centre of the frame is in focus. These functions will ensure that the strength of the center stays the same, and the further from the center the sample is, the lower its value. That means values at the edges go towards zero. Windowing functions thus change the original signal, but their change is designed to reduce the effect of windowing on the signal statistics [31]. An example of a normal frame and one to which the Hanning windowing function has been applied can be seen in Figure 3.2 (a) and (b), respectively.

The equation for the Hanning window can be found in equation (3.1). s(n) in the equation is the new amplitude of data point n, where n is any point of the M samples of the frame. For any point not in the frame, the value is 0 [32].



(a) Frame that starts and ends at random points.



(b) Frame to which the Hanning window is applied to put the focus on the center of the frame.

Figure 3.2: (a) Frame without windowing function (b) Frame with Hanning windowing. Both figures were manually created to show the same frame of 25ms taken from the QUT-NOISE-TIMIT [21] dataset. The frame contains no speech, only background noise.

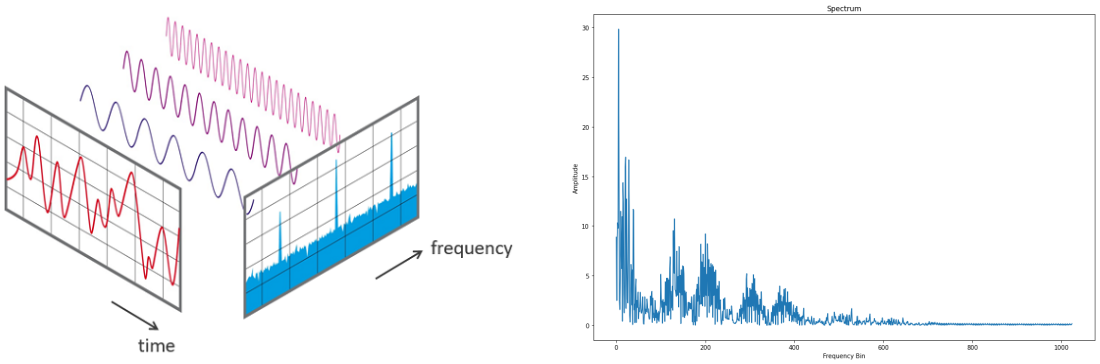
$$s(n) = \begin{cases} 0.5[1 - cos(2\pi \frac{n}{N})], & \text{if } 0 \leq n = N - 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

3.6 Fourier Transform

An important formula, known as the Fourier transform, provides a method to describe any audio merely as a combination of single-frequency sound waves. So far, only the temporal domain has been described. This domain is how the signal changes over time. In the temporal domain, each sample only captures the amplitude of all frequencies that overlap at that point in time. A change over time is then displayed as an alternating total amplitude for the different samples taken. The Fourier transform is a formula that breaks the changing amplitude over time into the different frequencies of which it is made. The result of applying the Fourier transform is called a spectrum. This spectrum is not in the time domain but in the frequency domain [33].

An intuitive figure explaining this conversion is shown in Figure 3.3a. In the time domain, the amplitudes of all frequencies are added together to get the resulting amplitude. In the frequency domain, each frequency itself is shown, and the higher a frequency, the higher the amplitude is shown for that specific frequency. In Figure 3.3b, the Fourier transform is applied to a single sample from the dataset. On the x-axis, each frequency is shown, on the y-axis, the corresponding amplitude. The algorithm used for this transformation is called the Fast Fourier Transform (FFT).

As mentioned previously, the frame size is chosen such that the signal does not vary too much. That is done because the FFT only works on static periodic signals. If the amplitude of one signal changed over time, it would not be possible to display that change in the frequency domain.



(a) Explanation of the Fourier transform [34].

(b) FFT applied to a single frame [33].

Figure 3.3: (a) Visual representation of how the conversion of a time domain signal to the frequency domain using the Fourier transform works. (b) Frequency representation of a frame.

However, signals, when not periodic, change quite frequently, so it is beneficial to have a small frame size. In Figure 3.3b, the frame size is 400 samples, which is 25ms. Therefore, the common approach is to use these short frames and perform the transformation on each one, having an overlap as mentioned before, to not lose any information. This is called the short-time Fourier transform (STFT).

3.7 Spectrogram

The result of applying this STFT to an audio file is a spectrogram. As it is not possible to represent the results in two dimensions anymore, the amplitude is shown as the color. The frequency, which was on the x-axis for an FFT, is now on the y-axis, with the x-axis value being the time step at which the frame belongs. An example of a spectrogram is shown in Figure 7.2 [33].

Additionally, the y-axis is converted to the mel scale, a logarithmic scale, where the difference between two equally distant frequencies always sounds the same. This is beneficial because the difference between 100Hz and 200Hz is perceived by humans to be smaller than the difference between 500Hz and 600Hz. Therefore, a scale called the mel scale was adapted to fix these differences [35].

Chapter 4

Background on Deep Learning

To ensure familiarity with the basic concepts of the deep learning techniques used in this thesis, this section provides a brief overview of the most relevant architectures and choices. This section covers different learning methods such as supervised learning and self-supervised learning because the models used in this thesis are based on these variants of machine learning. Convolutional layers are explained as they are the fundamental building block to all models used in this thesis. Additionally, the idea behind transformers is explained. The transformer architecture is used by Wav2vec, the main model of this thesis.

4.1 Basic Layers and Models

Deep Learning is a field of Machine Learning that uses artificial neural networks to learn from the input data to make predictions for similar unknown inputs [36]. Unlike classical Machine Learning, the neural networks consist of several layers of neural nodes that are connected, similar to how the neurons are connected in the human brain. Each neuron sums up the inputs of the previous nodes, weighted by a value that is learned during training [37].

There are a few different types of machine learning models. Important ones for this thesis are supervised, unsupervised, semi-supervised, and self-supervised learning. They differ by the type of data that is presented to the model at training time [38].

Supervised learning works by giving the model the training data as well as a label for each data point indicating the correct output for the task at hand. For example, for classification tasks, this label would represent the correct class of the data point. The model then tries to learn the weights such that it can predict these labels for the given as well as for data where it does not know the label [38].

In unsupervised learning, the dataset does not have any labels for the data point. The model does, therefore, not know the correct outcome for any data points. Instead, it has to analyze the data to find significant differences or similarities in useful features of the data to structure the data. An example of that is clustering, where the model analyses the features and separates the data into different clusters, where each cluster contains similar elements [38].

Semi-supervised learning is a mixture of these two approaches by using mostly unlabelled data but enhances it with a small set of labeled data. It is useful for challenging tasks, where a classical unsupervised model would not suffice, but it also is not doable to label each data point separately (for example, due to time limitations) [38].

Self-supervised learning is a type of unsupervised learning that uses unlabelled data to create a useful representation of the data. This representation can then be fine-tuned to the task with a limited amount of labeled data [39]. This paradigm has been very successful in Natural Language Processing (NLP) tasks and is commonly used in transformer models like BERT [40], which creates a bi-directional encoder representation of the token (word or word piece) of an unlabelled sentence. This representation can then be adapted to the corresponding NLP task [40]. Next to BERT, self-supervised learning is also used in Wav2vec, the main model of this thesis. As Wav2vec is also a transformer model, transformer architectures will be discussed in more detail in Section 4.3.

4.2 Convolutional Neural Network

Convolutional neural networks (CNNs) are a type of artificial neural network mainly used in computer vision, and they contain several layers of convolution. For images, a convolution is done by applying a filter with weights over each small group of pixels to look at the different local regions of the image. Several filters can be used at the same time so that one filter highlights the regions that have a vertical line and another filter those with a horizontal line. Deeper convolutional layers combine these outcomes to more abstract shapes like edges and squares. These filters are usually initialized randomly and then trained on an image dataset [41]. CNNs are also used for NLP tasks and speech processing, using words and a part of the waveform instead of the pixels, respectively [5, 41, 42]. For the models used in this thesis, CNNs are the central layer. The first two models, CNN-VAD and U-net, rely heavily on CNN layers for analysing the spectrogram images and MFCC features. For Wav2vec, CNNs are also used to process the raw audio input.

4.3 Transformer

The transformer is a model containing a stack of encoders and decoders that uses attention (which parts to focus on for the current input) instead of recurrence and is part of many of the current state-of-the-art models in NLP, such as BERT [40]. The main benefit of the transformer is that

it uses multi-head attention using a self-attention mechanism. This allows the model to get rid of recurrent layers, making the model parallelizable and thus more efficient [43, 44, 45, 46].

Self-attention is a type of attention that connects different parts of a sequence to generate a representation of the whole sequence. That means that for each token that is processed, the relevance of the other tokens is calculated. The transformer uses multiple attention headers at the same time and thus allows the model to focus on multiple positions in the sequence simultaneously [44, 46].

For short sequences, using recurrence can be sufficient. All embeddings are influenced by the previous embeddings. That is done by recursively taking a weighted hidden embedding of the prior token into account when computing the current embedding. However, as the sequence grows, the influence of a token further away decreases, as it went through several iterations of using the weighted hidden embedding.

Self-attention fixes that problem by calculating the importance of the other embeddings in the sequence for the current embedding separately without using several recursions. This allows the model to provide a rich representation of both local and global (long-distance) contexts. For speech and natural language processing, this can be quite important. For example, important cues, such as the noun of the sentence can be far away from the verb. However, the verb and noun should always be seen in the same context. For the task of speech detection, the tokens mentioned above are frames. With self-attention, the frame in the middle of a window can create its embedding taking all other frames, each weighted by its importance to the current frame, in the window into account. So, for example, the first frame might have a very similar embedding to the current one. That means they might be more related than others as they might belong to the same source of noise or might be a similar phoneme. Global patterns, such as repeating noises, can then be treated together and, therefore, processed correctly by the downstream model, even if the rest of the noise changes [47].

Transformers like BERT and Wav2vec also use masking, which means that some of the inputs to the transformer are replaced with a [Mask] token, and the model aims to predict the correct word based on the surrounding words in the sequence [48].

Chapter 5

Deep Learning for Voice Activity Detection

5.1 Existing Machine Learning/Deep Learning Models for VAD

As mentioned in the introduction, voice activity detection is a binary classification task addressing whether speech exists in a frame or not. Several different models have been created for such purposes. An overview of these models, as well as their benefits and drawbacks, is shown in the Appendix in Figure A.1. Many existing models rely on extracting low-level features from the audio, which will then be classified. These low-level features include, for instance: the zero-crossing rate, power, harmonicity, and Mel-frequency cepstral coefficients (MFCC) [49].

The first of these models used a fixed threshold as the discriminator that classifies audio as speech if, for instance, the zero-crossing rate (ZCR) is below a predefined value for that audio segment. The ZCR is the rate of how often the audio signal changes from positive to negative or from negative to positive. It is, therefore, the number of times the signal crosses the zero line in a segment [13, 49]. This method relies on the observation that the ZCR is higher during noise segments than during speech segments [50]. This feature, used together with the energy feature to ensure that no low energy segments are picked up, is able to get a decent accuracy while still being fast and computationally cheap [13].

However, this model only works in high SNR environments of around 30dB, or environments with a stationary noise [49]. To also include non-stationary noise in lower SNR recordings, Kobatake et al. proposed an algorithm in 1989. It contains two subsystems, one for detecting non-stationary segments and the other for detecting speech or non-speech. For that, they used a linear prediction model to model the background noise and the smoothness of the pitch frequency, as well as some

other features for the detection [51]. This model was able to improve the noise level to an SNR of 10dB but was still worse than the model from Tucker which used periodicity, as explained in the next paragraph [12].

Tucker's [12] model uses a periodicity estimator to find the segments that contain periodic elements in the frequency domain, as that is often speech. That works because speech usually has a similar frequency throughout the whole sentence or text. Repetition of this frequency over several frames shows that it likely is a person speaking. Other noises are usually not periodic but rather occur randomly or are constant. A fast Fourier transform (FFT), as explained in Section 3.6, is done on the speech, and adjacent frames are analyzed to see if the same frequencies appear with similar space in between. If that happens, the detector returns speech; otherwise, it is considered noise. As the algorithm needs multiple frames, it is impossible to find the exact boundaries when the speech starts and stops. Additionally, the focus is more on a high recall than on being precise. That means only those segments that contain no speech are rejected, while the rest is passed on. That makes the model work reliably down to an SNR of 0dB and detect most speech at -5dB (80%). However, this also means the model is sensitive to any periodic noise in the background [12].

In 1993, cepstral-based models were introduced to resolve the problem of periodic noise and to allow a wider amplitude range. Cepstra is the discrete transform of the log of the frequency domain, and the most known cepstral-based feature is MFCC. The cepstral-based model relies on the analysis that the variance of the cepstra from noise is lower than from speech, independent of the signal level and amplitude, making the model more robust [52].

The next milestone in voice activity detection was the introduction of statistical model-based approaches by Sohn et al. [53]. The authors applied a statistical model with a likelihood ratio test as decision rule to the task of speech detection, which has the benefit that it is easier to optimize the relevant parameter in contrast to the previously used heuristics. Additionally, Sohn et al. used a hidden Markov Model (HMM) to incorporate the prediction of the previous frame into the decision-making process. Nevertheless, this model is not able to handle low SNR environments as well as the previous models; it works well above an SNR of 5dB [53].

In 2002, Enqing et al. [14] applied support vector machines (SVM), a supervised machine learning method, to the task. The SVM proposed by Enqing et al. uses four different features, such as the zero-crossing difference between the frame and the rolling average, as well as the spectral distortion of the background noise. The model then tries to create a hyperplane separating the noise and speech segments from each other. The SVM model achieves a better error rate than the previous model while requiring fewer training samples. The downside of this model is that it does not work well in noisy environments where the SNR is below 5dB [14].

While simple machine learning models were explored, also various statistical models were tested, such as in Chang et al. [54], who tested which statistical function best represents the noise [54]. Gaussian Mixture Models (GMM) were applied in Ying et al. [55] to the task of voice activity detection to improve the existing statistical models through the help of unsupervised learning. This

model works by creating clusters for speech and non-speech and then classifies each frame to one of the clusters, updating the threshold between them sequentially. This has the advantage that, contrary to the simple models, no threshold has to be set manually, and the threshold will adapt to noise changes over time. Additionally, contrary to previous statistical models, the common assumption that the audio begins with non-speech is not necessary. This assumption usually holds for many models to be able to model the noise in the beginning. However, the key problem of this approach is if an audio file contains no speech. Choosing two clusters for GMM would then result in an arbitrary choice rather than speech/no-speech [55].

In the last ten years, various deep learning approaches have been used. The first one was the deep belief network [10]. The researchers noticed that existing shallow models and statistical models were unable to discover indicators for speech or non-speech across multiple features. Thus, they developed this model, which used ten common features and combined them in a non-linear way to express highly variant functions. This allowed the model to only focus on the important indications for each feature and create a representation that combines them all. The model also uses unsupervised pre-training and supervised fine-tuning afterwards, where the pre-training acts similar to a regularizer by preventing overfitting. This is helpful to be able to fine-tune the model to the environment without needing too much data due to the higher amount of features extracted. They achieved good results, a better detection rate than any SVM or statistical model so far in almost all environments and SNRs tested [10].

Around the same time, Hughes and Mierle [56] had another approach to deep learning by using recurrent neural networks (RNN) as an alternative to the GMM models, which use HMM to keep track of the labels for preceding frames. The model uses 13-dimensional Perceptual linear prediction (PLP) features as input and uses various points of recurrence between two frames to incorporate the results of the previous frames. Using RNNs has the advantage that, compared to HMMs, a greater number of preceding frames can be remembered, and, thereby, better results can be achieved. Additionally, the resulting model has significantly less trainable parameters than GMMs and is, therefore, better for use cases, such as real-time speech detection, where efficient computation is important [56].

Zhang and Wang proposed another model in 2014, this time mainly targeting low SNR environments [11], known as boosted DNN (deep neural network). That algorithm generates several predictions per frame using a DNN and then aggregates them to create a final prediction. In that paper, they also used a new feature, multi-resolution cochleagram (MRCG). Cochleagrams mimic the inner ear and display the simulated excitation pattern of the cochlea, a part of the inner ear [57]. The MRCG feature combines different cochleagram features of different frame windows to use both local as well as global information. Together, the MRCG and the boosted DNN create a model that is robust against different noises and manages to get an area under the curve (AUC) of around 90% at an SNR of -5dB [11].

Recently, an adaptive context attention model (ACAM) has been created by Kim and Hahn [4]. ACAM is built as a recurrent attention model which takes a frame as well as several surrounding frames as input and decides which out of those frames is best suitable for speech detection. For this model, it is not important to know exactly in which frame the speech started, so it can avoid frames where the noise overlaps with the speech in the extracted features and instead focus on more distinct frames, where the label is more obvious. As input features, the model also uses MRCG. The results show a small increase in AUC for some noise environments averaged over all SNR compared to the boosted DNN [4, 11].

5.2 Technical Background on Models Used

To answer the main question, three different models will be used and compared in this thesis, namely CNN-VAD [2], U-net [6] and Wav2vec [5].

Each of these models uses different features and entirely different architectures. The CNN-VAD is a simple CNN extracting information from spectrogram representations of each frame. The U-net is a U-shaped network, combining both local as well as long-distance features to also include the context. As input, it uses several subsequent MFCCs. Lastly, Wav2vec is a transformer model pre-trained on speech recognition. In this thesis, a downstream model will be created to adapt this model to speech detection and to use additional features next to raw audio. In the following chapters, each of the models will be explained in more detail.

5.2.1 CNN-VAD inspired by LeNet5

The first model is taken from literature and was created by Silva et al. [2]. Its architecture is based on the LeNet5 architecture, a pre-trained convolutional neural network created in 1998 [58]. It was originally created for document recognition for images of size 28 * 28.

The original network is called LeNet5 because it has five layers with learnable parameters. It consists of three convolution layers, each followed by average pooling. The output of that is fed into two fully connected layers and lastly, a Softmax classifier which assigns the corresponding class [59].

Architecture

In the paper by Silva et al. [2], this architecture was adapted to the task of VAD to explore the possibility of using CNNs for speech detection. They modified the LeNet5 model by changing the input size and removing the last convolutional layer. The architecture is shown in Figure 5.1. In the Table A.1 in the Appendix, the details of each layer are shown.

As shown, the input has a feature map size of (128, 32, 1). That means each point of the data has that shape. The total shape of each layer, therefore, also includes the batch size as the first shape.

The first layer has 20 filters with a filter size of 5 * 5 and a stride of 1. Tanh is used as the activation function. This is followed by an average pooling layer with a stride of 2 and a pool size of 2 * 2. The resulting feature map has the size (64, 16, 20).

The second convolutional layer uses the same filter size, stride, and activation function but has 50 filters instead. This results in 25050 learnable parameters. The following average pooling layer reduces the filter map to (32, 8, 50).

To feed the data into a dense layer, a flatten layer is used. This layer has no learnable parameters and just reshapes the feature map to the size (12800).

A dense layer with ReLU activation function and 500 neurons is then applied. This creates 6400500 learnable parameters.

The output of the first dense layer is fed into a second dense layer with 2 output neurons and softmax activation. The softmax activation thus assigns either class 0 or class 1 to the data, where 0 is no speech and 1 is speech.

In total, the model has 6.427.072 learnable parameters and 0 non-trainable parameters.

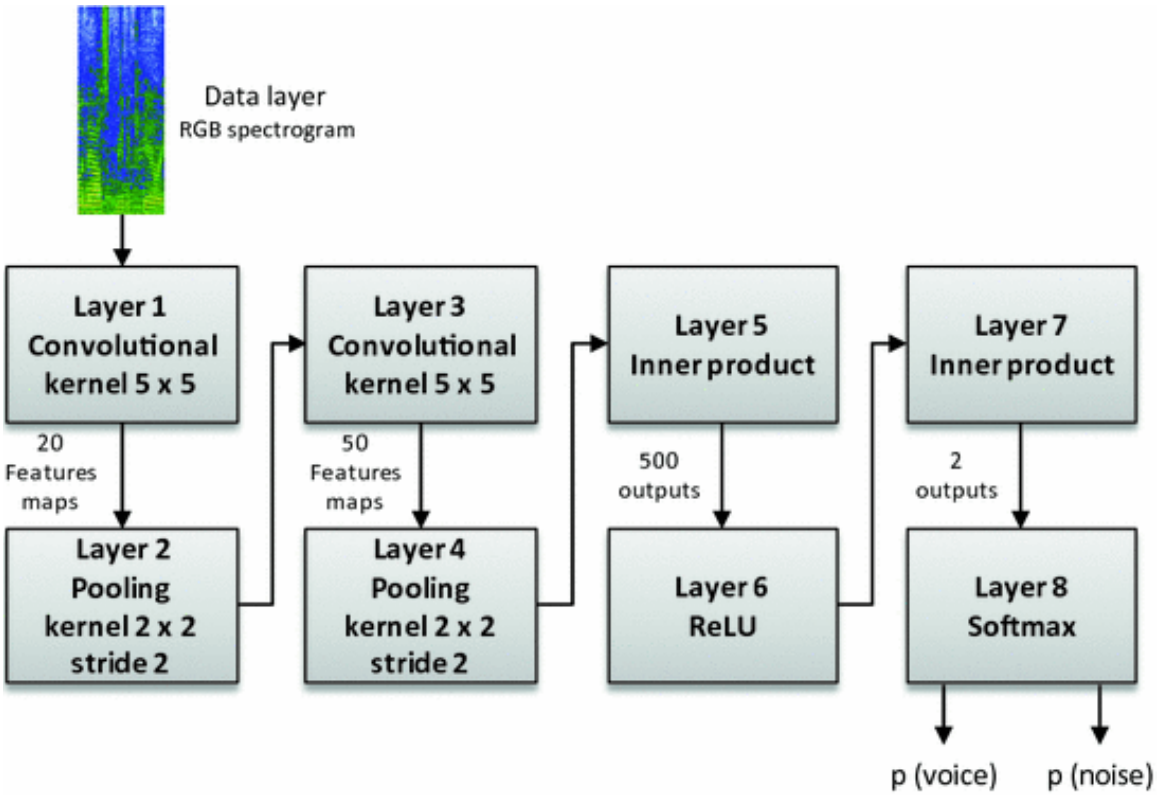


Figure 5.1: This figure shows the architecture of the CNN-VAD model created by Silva et al. [2]. In the original paper, a spectrogram with the shape (128, 32, 1) is used as input to the model.

5.2.2 U-net

The second model is a U-net. This model was created for medical images [60] and was named after its U-shaped architecture. It was adapted for speech source separation tasks by Hennequin et al. at Deezer [61] and Stoller et al. at Spotify [3]. Additionally, in 2020, Gusev et al. used a U-net as VAD step in their approach to recognizing speakers [6]. For their VAD model, they used a smaller version of the U-net proposed by Deezer and Spotify for source separation.

Architecture

The smaller U-net version created by Gusev et al. [6] has the following architecture. An overview is shown in Figure 5.2.

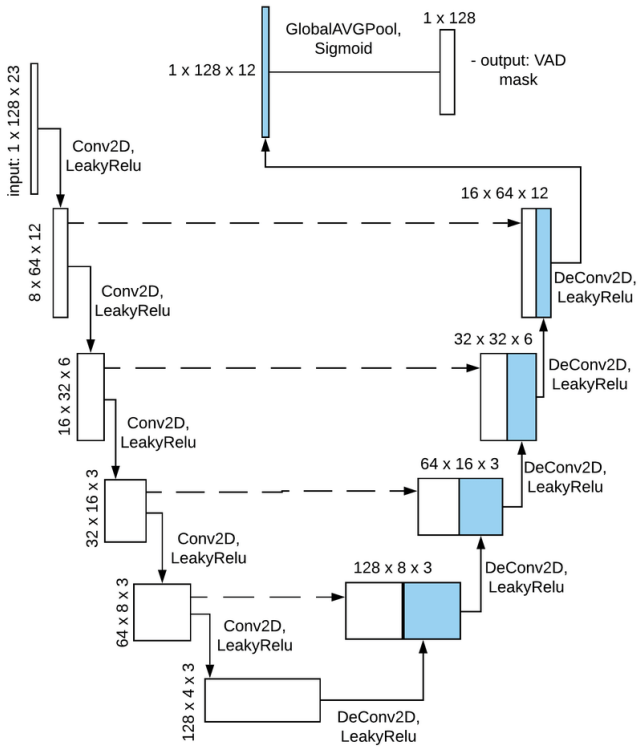


Figure 5.2: U-net architecture created by Gusev et al. [6] for VAD.

The U-net has several kinds of layers, convolutional, deconvolutional, batch normalization, dropout, concatenate, multiply, global average pooling, reshape, and dense layer.

The convolutional layer works as explained in Section 4.2. The deconvolutional layer is used to up-sample the data by reversing the convolution operation.

Batch normalization layers are added to normalize the inputs at various stages of the network.

Lastly, global average pooling is used as the last layer. The difference to a standard average pooling layer is that pooling is not done across parts of the feature map, but instead, one value is returned, which is the average of the complete feature map. Global average pooling is designed to replace the last fully connected layer. This is more native to the structure of a CNN, as the features and output classes are closer correlated and the features can therefore be explained easier [62]. In this case, all features corresponding to the same frame are averaged to get one value per frame.

The U-net also uses the Leaky ReLU activation function, which is an improved version of the ReLU activation function. The difference between them is that the Leaky ReLU has a small slope for negative values, e.g., y=0.01x. This resolves the dying ReLU problem, which means that the neurons return 0 for any input and thereby become inactive [63]. Additionally, Leaky ReLU is more balanced, as the average activation is closer to 0, which allows the model to learn faster [64].

These layers are combined as follows and as depicted in Figure 5.2.

On the left side of the U-net, the convolutional layers are used to extract important feature information while decreasing spatial information. That means with succeeding layers, more details are gathered for a decreasing input size.

On the right side, the original image is inferred from the extracted feature information. This is done by propagating the relevant features to higher resolutions by using deconvolution. The dashed arrows in Figure 5.2 mean that the convoluted image from the left side is combined with the deconvoluted features. This combination helps to create the upsampled image as closely to the original image as possible.

Each input data point has a feature map size of (128, 32, 1). The parameters and shapes of each layer are shown in the Appendix in Table A.2

5.2.3 Wav2vec2.0

Wav2vec 2.0 is a state-of-the-art model in speech recognition. It is a framework for self-supervised learning of speech representations and outperforms the previous models when fine-tuned on 10 minutes of labeled data from the Librispeech dataset [5]. It has also been successfully applied to other speech processing challenges, such as emotion recognition, speaker verification, and language identification [65, 66].

Latent Speech Representations (Z)

As input, the model receives the raw waveform, which is fed into several layers of CNNs. For wav2vec, the receptive field, which is the region in the waveform that the output of one multi-layer CNN is affected by, is 25ms. One multi-layer CNN block is shown in Figure 5.3 as one blue block. Furthermore, shown by the overlap of the blue blocks, the different receptive fields are overlapping by 5ms [5].

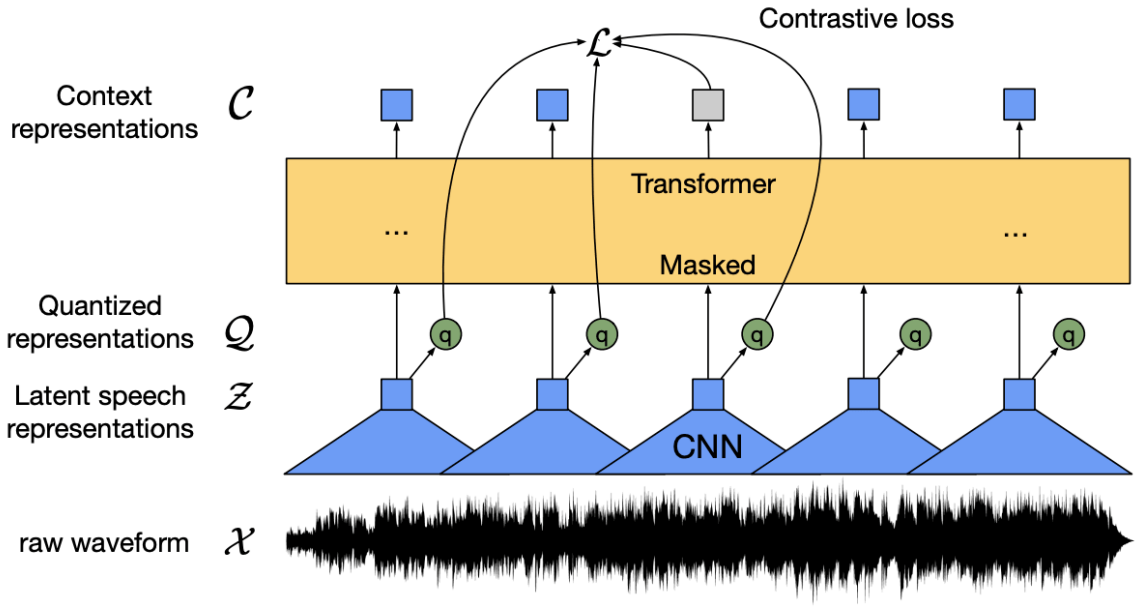


Figure 5.3: Architecture of the wav2vec2.0 framework [5].

This step is the local encoder, as it creates a lexicon of continuous vectors from the output of the CNNs for the different waveforms. Each frame has its vector, which contains a simplified version of all the important speech features of the waveform, also known as the latent speech representation [67].

This latent speech representation is a representation of the frame, representing its different attributes. For instance, the same phoneme spoken by different speakers will be in a similar subspace of the latent space, while different phonemes will be in different subspaces [68]. It is, therefore, possible to detect based on the representation which word is spoken, and there are minor differences based on other statistics of the speech, such as speaker and tone [68]. However, that only works when no neutralization is happening. Neutralization means that different phonemes are pronounced the same way in certain contexts [69].

Quantization (Q)

The next step is to convert vectors from latent speech representations to quantized representations. This is done by using a codebook to transform the continuous values into discrete output, namely the codebook entries. For this, the entry from the codebook, which is the closest match to the speech representation, is chosen. For wav2vec, entries from multiple codebooks are concatenated, using one entry per codebook. A special softmax method, namely Gumbel softmax, is used, as finding the closest match in the codebook is not a differentiable operation. Instead, a softmax over all entries of the codebook is given with the probability of the entry representing the given representation.

This allows the model to learn by backpropagating through this step and updating the probabilities [5, 70]. The benefit of using this discretization is that it is easier for the prediction model to learn if little training data is available, as all very similar data points are discretized to the same discrete values [5].

Transformer

The last step of the model is the Transformer, explained in Section 4.3.

For wav2vec, the Transformer receives the latent speech representations and masks some of them. With that, it attempts to create a representation of the masked token using only the context. This contextualized representation might be something similar to the localized representation but contains additional information related to the surrounding frames. The base model has 12 transformer blocks with eight attention heads each. The model is trained using unlabeled data and fine-tuned on the dataset corresponding to the preferred task. [65].

Loss Function

The general contextual speech representation is trained using three different losses. During the pre-training, a contrastive loss is applied. In this step, the transformer part of the model tries to identify the correct representation for a masked entry from a set of distractors, which are other quantized candidate representations, and are sampled from other masked entries of that audio [5]. The loss is called contrastive because it contrasts the distance to positive and negative examples. The contrastive loss is better (lower) if other positive examples are close to the predicted embedding and negative examples are further away [71].

To balance the usage of the different codebook entries, a codebook diversity loss is used. This is done by ensuring the averaged softmax distribution over the codebook entries is as random as possible, thereby allowing each element to be equally important [5].

For fine-tuning the model on the speech recognition datasets, the authors of wav2vec 2.0 [5] used a connectionist temporal classification (CTC) loss. This means a temporal classifier is trained that converts the context representation into a probability distribution over possible labels. Input sequences can then be classified as those labels by selecting the most probable label. For instance, for speech recognition, an audio file is given as input, and the corresponding transcription or the phonemes are the label sequence. The context representation is then fine-tuned on this classification task [5, 72].

Transfer Learning

The model can also be adapted to other tasks. For that case, the CTC loss will not be used, and instead, a custom head can be added. This downstream model has access to the embeddings created by the model and can use them to fine-tune the outputs to the preferred task.

Chapter 6

Data

6.1 Noise

The background noise for all datasets used is taken from QUT-NOISE [21]. This noise dataset contains 600 hours of noisy sequences in different scenarios as well as different signal-to-noise ratios (SNR), ranging from an SNR of +15 to an SNR of -10. It has five different scenarios: car, cafe, home, reverb, and street, and two conditions per scenario, such as open and closed windows.

An overview of the distribution of noise as well as the speech proportions is shown in Figure 6.1.

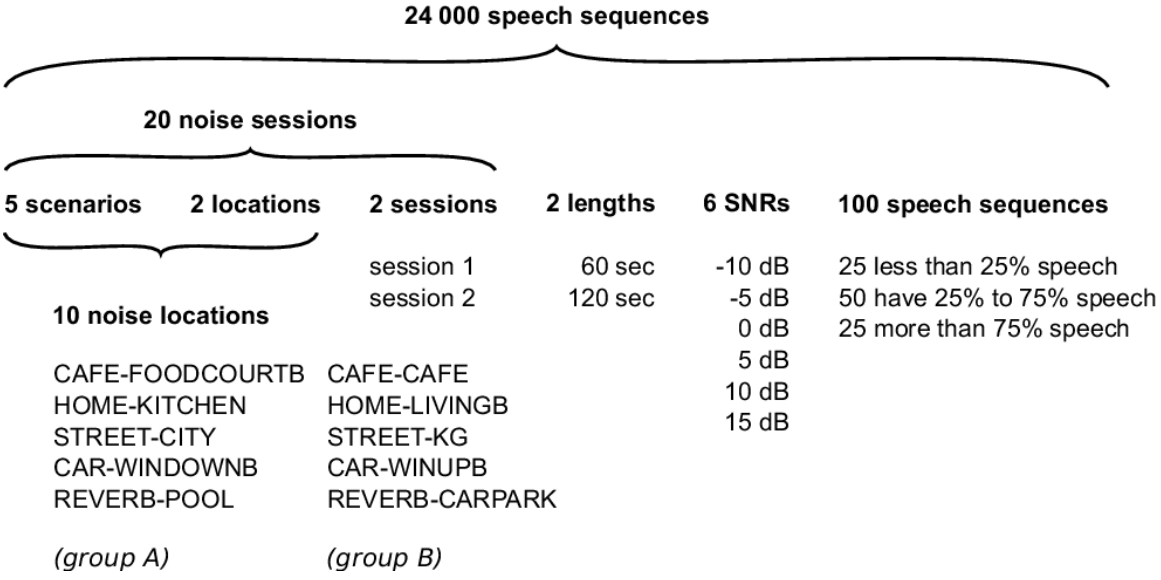


Figure 6.1: Distribution of the QUT-Noise-TIMIT audio sequences by scenario and location [21].

6.2 QUT-NOISE-TIMIT

To answer the research questions, the primary dataset that is used is the QUT-NOISE-TIMIT dataset [21]. In this dataset, the speech segments are taken from the TIMIT [73] dataset and are placed at random times during one-minute recordings of the background noise. TIMIT is a corpus containing 630 speakers with eight different American English dialects. The dataset consists of the ten spoken sentences per speaker as well as the corresponding hand-verified transcriptions of each sentence [73].

6.3 Common Voice

Next to TIMIT, Common Voice, a very large open source speech dataset from Mozilla [74], consisting of over 1600 hours of validated English speech, will be added to the background noise in the same manner. This dataset contains next to the transcription of the speech, also further identifying details such as age, gender, and accent [74]. This dataset will mainly be used as additional training of the wav2vec model and can also show whether certain accents are especially difficult to pick up as speech, which can be analyzed in future research. Furthermore, other languages of this dataset can be applied to the model to train its cross-lingual capabilities.

6.4 RAVDESS

For the subquestion of recognizing the emotions of the speaker, the RAVDESS dataset is used. This is chosen to be able to compare the results to the ones achieved by Pepino et al. [65]. In that paper, they already use wav2vec 2.0 for their emotion recognition model. However, contrary to their approach, this dataset will, just like the previous ones, be mixed with the noise from the QUT-NOISE dataset.

6.5 Data Generation

Instead of providing the noisy speech dataset directly, the authors [21] provide the Matlab code [1] used to create the dataset, such that the dataset can be generated manually. This allows the user to use different parameters as preferred while also being able to use the standard parameters to get an equal dataset as used in other research. The code works by taking segments of noise as well as segments of speech and randomly distributing the speech across one minute of noise. The noise, as well as the speech segments, can be replaced with other datasets. The only requirement is to use the same file structure as done for QUT-NOISE-TIMIT.

[1] https://research.qut.edu.au/saivt/databases/qut-noise-databases-and-protocols/

The method requires several parameters as input. One is the path to the noise files, including their impulses, which can be downloaded from the official website either including all 10 locations or a subset of them. The noise is a 30-60 minute recording of the various background noises at the set location without any speech. Additionally, labels for each noise segment are included to exclude parts of the audio where the audio was unusable due to poor quality of the recording [75].

Lastly, some parameters for the data generation have to be set. For this thesis, the sampling rate is set to 16000Hz, and the audio file length is 60 seconds. The method was run several times, once for each SNR in [15, 10, 5, 0, -5, -10, -15, -20]. The method supports setting multiple SNR at once, but it will then split the number of audio files between the given SNR, thereby having fewer resulting wav files per SNR level. As each model is only trained on one noise level at a time, it is not a problem that the audio files between several SNR levels overlap.

Using this data generation means that the noisy speech was electronically induced instead of generating it acoustically. This could mean that the models do not learn the speech itself but instead learn the difference of reverb between the background noise and the speech. This method of data generation was chosen despite this drawback, as acoustically creating this noisy speech would require a lot of manual work.

Chapter 7

Methodology

The methodology applied in this thesis to answer the research question is explained in the following sections. The first step is pre-processing, which includes getting the data in the right shape to be able to extract data and get the labels. After that, the features are extracted. Chapter 7.2 explains how each feature is extracted and what shape it has. The features are used as input to the different deep learning models, which are then trained to predict the correct labels. Lastly, post-processing is performed on the predicted labels to remove outliers. The predicted labels are then used for the evaluation of the model. An overview of the pipeline is shown in Figure 7.1.

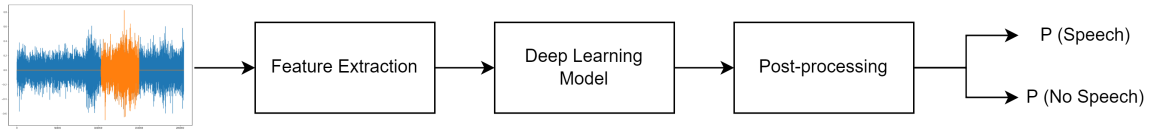


Figure 7.1: Overview of the steps taken as methodology. It starts with the raw audio, represented by the plot of the signal wave, with the orange area being the speech. The first processing step is windowing. For each window, the features are extracted and fed to the deep learning models together with the raw audio in some cases. Each deep learning model is trained on different features to detect speech. The trained model is then evaluated on the test set, using post-processing to smoothen the predictions.

7.1 Pre-Processing

In this section, I will explain how the data is handled before extracting the features. There are two main tasks that are performed on the dataset before any feature extraction. These are the splitting of the data into overlapping windows and creating the labels.

7.1.1 Windowing

As explained in Section 3.5, windowing is done by splitting the audio recording into small segments called frames, where the speech does not change much in one frame to allow STFT. However, common implementations also include a larger-scale of windowing, which is split into these smaller frames. This larger window will be referred to as window, while the smaller windows will be referred to as frames. These larger-scale windows are useful to be able to include knowledge of surrounding frames to make the predictions more stable.

In this thesis, different sizes of the smaller and larger scale windows will be used. For the two models taken from literature, 160ms windows are used for the CNN-VAD [2] model, and 2.56s windows for U-net [6]. Both models use an overlap of 50% and split each window into 128 small frames with a frame size of 40 samples (2.5ms) and 400 samples (25ms), respectively.

As each 60s audio file cannot be split into these windows evenly without having some remainder, zero padding was added at the end of each audio file to get to the next even split. For each window, the Hanning function, as explained in Section 3.5 is applied to focus on the center of the window. Each window is then fed separately into the feature extraction methods to create the desired features.

7.1.2 Labels

Depending on the model used, the labels can be either set per frame or per window. For the CNN-VAD, the authors use one label per 160ms window [2]. That means the 128 frames inside one window will all yield this label together. The labels, therefore, have the shape (#windows, 2). For the U-net model, Gusev et al. [6] use one label per frame instead of per window. This is likely because each frame is ten times the size of the frames of the CNN-VAD implementation by Silva et al. [2] and having one label per 2.56s would not be narrow enough for real-world applications. The U-net model labels thus have a shape of (#windows, 128, 2).

For both window sizes, the labels are created using the same overlap, frame size, and padding as the features extracted. Each label is one-hot encoded, either [0, 1] or [1, 0], where the first describes no speech and the latter speech. As the model should find all segments containing any speech, the window or frame is labeled as speech if it contains any speech samples.

7.2 Feature Extraction

In this thesis, two types of features will be extracted for the different models. In the following sections, each feature will be explained shortly, and the usage in literature and implementation details will be given. Note that for some models, it is possible to use multiple features at the same time.

7.2.1 Spectrograms

The first feature is spectrograms. This is a representation of the acoustic spectrum of the signal as it changes over time, as explained in Section 3.7 [76]. A visual representation of the feature is shown in Figure 7.2. In the image, the color displays the amplitude of the signal in decibel, and the y-axis shows the frequency [76].

There are several possibilities for setting the hyperparameters in the spectrogram creation. Different implementations using varying hyperparameter settings are used in the relevant literature.

In the following subsections, it is explained how each parameter is set. If a parameter is not mentioned, the default value is used. The important parameters that can be set are:

1. length of the FFT window (also referred to as frame): This describes how long each of the frames will be.
2. hop length: This describes how many samples will be skipped before the next FFT will be taken. Given an FFT window length, this describes the overlap between succeeding frames.
3. Number of Mel bands to generate: How many bins are used. The energy is summed up separately in each bin and each bin corresponds to the same size using the mel scale.
4. centre: Whether each frame has the current sample (determined by hop length) in the centre or in the beginning. If the current sample is at an edge, the remaining values will be padded
5. padding: The type of padding to use at the edge, when centre is set
6. window: A window function that is applied to the frame, e.g. Hann
7. fmax: The highest frequency of the data, if not set defaults it to half the sampling rate.

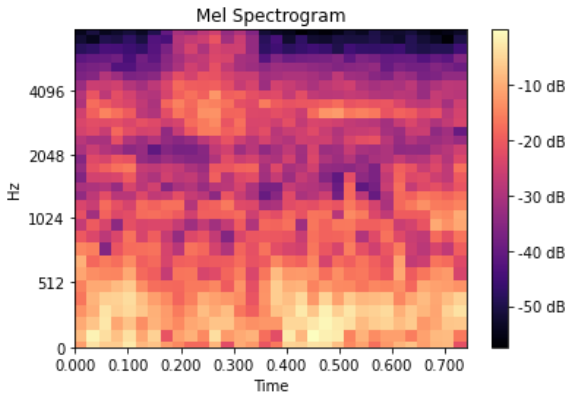


Figure 7.2: Example of a mel-spectrogram of a window as used in this thesis.

CNN-VAD Spectrogram

The first implementation uses the parameters as set in the CNN-VAD paper [2]. Instead of defining the FFT window size directly, the audio file and the shape of the output are given, and the FFT window is calculated implicitly.

In their paper, they use an output shape of (32, 128). This is the transposed version of the spectrogram, with 32 being the number of mel bands and 128 the number of FFT frames per window. The input for their spectrogram is a window of 160ms (2560 samples). This results in a hop length of 20 samples and an FFT window size of 40 samples. This is a 50% overlap with the previous frame and a 50% overlap with the following frame. Each window has a 50% overlap with the surrounding windows. The Hann window is applied to the inputs. Each frame is centered, and zero padding is used for the first and last frame.

The labels, as explained in Section 7.1.2, are set per window. One spectrogram, as shown in Figure 7.2, therefore, returns one label, either speech or no speech.

U-net Spectrogram

The second implementation uses the hyperparameter settings as done in the U-net [6] implementation, only adapted to spectrograms instead of MFCC. So instead of using MFCCs, mel bands are generated. It was chosen to use 32 mel bands instead of 23 MFCCs. That has the advantage that the features used are comparable to the CNN-VAD Spectrogram, only having a larger frame and window size. The window size used by Gusev et al. is 2.56 seconds with 50% overlap, and for the frames, an FFT window of 25ms is used. Between succeeding frames, 20ms are skipped. That means each frame has an overlap of 5ms with the previous and 5ms with the following frame. The resulting shape is (128, 32) per window, which is the same shape as for the CNN-VAD spectrograms and all features extracted in this thesis. This makes it easier for all models to test different features without needing to significantly change the model.

As done by Gusev et al. [6], one label is extracted per frame instead of per window. This means the labels have the shape (128, 2) per window.

Spectrogram 640ms windows

Next to these two main spectrogram variants, there are also two more adaptions, which only differ from the CNN-VAD Spectrogram described in Section 7.2.1 by the size of the frames and window.

This first one has a window size of 640ms (10240 samples) and a frame size of 10ms (160 samples). The hop length is adapted to 5ms (80 samples). This still results in 128 frames per window and one label per window. In summary, each frame and window is four times as big as the one described in Section 7.2.1.

Spectrogram 2.56s windows

The second CNN-VAD Spectrogram adaption increases this window and frame size again to 2.56s (40960 samples) and 25ms (400 samples). This means it uses the same frame size, hop-length, and window size as the U-net spectrogram. The only difference between this spectrogram and the U-net spectrogram from Section 7.2.1 is that this only uses one label per 2.56s window.

7.2.2 MFCC

The second feature is called Mel-frequency cepstral coefficient (MFCC). MFCCs are similar to mel-spectrograms. To compute the MFCC, the log needs to be taken from the spectrogram, and a discrete cosine transform is applied. As the latter is a linear operation, it can also be performed by the first layer of the neural network [77].

For this thesis, the parameters used are similar to the ones used by Gusev et al. [6]. The input is 128 frames, and the frames are extracted from a 2.56s long sliding window with 50% overlap. Each frame covers 25ms, and a new frame is created for every 20ms. The only difference to the implementation from Gusev et al. [6] is that 32-dimensional MFCCs are used instead of 23-dimensional. This change is done to have the same feature shape as the spectrograms. Additionally, using more MFCC dimensions should only improve the accuracy of the model, as the model can simply learn to ignore the higher-order coefficients.

7.2.3 Raw audio

The third type of input, which will be called a feature here for the sake of consistency with the other types of input, is raw audio. Wav2vec2.0 and other models have shown that models can learn a good representation from the audio without the need to manually engineer features [5]. As part of this thesis is about testing Wav2vec2.0 and its possibility to be adapted to VAD, it is helpful to test the feature on Wav2vec2.0 as well as on other models.

Just like the other features, different sizes were tested. As raw audio is mainly used in the wav2vec model using spectrograms as well, the raw audio size is equal to the window size used there. That means the raw audio has a length of either 640ms when using the spectrogram described in Section 7.2.1 or 2.56s when using the spectrogram described in Section 7.2.1.

7.3 Deep Learning Models

Three different deep learning models were implemented in this thesis to compare the outcomes to the literature and provide the best performing model to the police to be used on different use cases, such as detecting speech from in-car conversations and interviews. The first two (CNN-VAD [2] and U-net [6]) are taken from literature and were only adapted to use the features extracted as explained above and to work with this dataset. The third model, which is the most relevant one for this

thesis, is a downstream model for Wav2vec2.0 and is created for this thesis. The general idea and architecture of each model is explained in the corresponding sections (CNN-VAD in Section 5.2.1, U-net in Section 5.2.2 and Wav2vec in Section 5.2.3). In the following sections, the implementation details and adaptions to handle the different features will be given.

7.3.1 CNN-VAD

Implementation

The model described in Section 5.2.1 was implemented in Tensorflow using the method described in the paper. As no details about the usage of padding were given, two models were trained on the same dataset, one with padding and one without. The model with padding performed better (HTER of 12.173% compared to 13.030% without padding on SNR+05), so the one with padding was kept for the analysis.

As input to the model, the authors used spectrograms of 160ms frames of each audio file with an overlap of 50%. These RGB Spectrograms were generated as explained in Section 7.2.1.

An overview of the resulting model is shown in the Appendix in Table A.1.

Adaptions

In addition to the original model from Silva et al. [2], two main adaptions were done to the CNN-VAD model regarding the input features, as extracting RGB spectrograms for an audio file is quite time intensive.

Raw audio The first adaption uses the same division into windows, as mentioned in the original paper. However, instead of getting 128*32 shaped rgb spectrograms for every frame, the raw input is used. This also means that instead of the 2d convolutions used, one-dimensional convolutions will be used. The kernel size and filters were not changed.

MFCC Using the same window as defined above, MFCC features were extracted for each window. Inside each window, MFCCs were extracted with the size of 25ms for every 20ms. 25 MFCCs and 40 mel bands were generated for each frame.

7.3.2 U-net

Adaptions

For the U-net, MFCC and spectrograms were tested. As the labelling varies between label per frame and label per window, the output of the model was adapted, as explained in the next paragraphs.

MFCC The original features for this model are 128 frames using 23-dimensional MFCCs. The 128 frames were extracted from a 2.56s long window with a 50% overlap. As mentioned, each frame accounts for 25ms, and a hop-length of 20ms is used.

To work with the feature used in this thesis, which uses more MFCCs, the input was changed to 32 MFCCs. However, additional tests with 23 and 25 MFCCs were also done. The resulting model architecture is shown in the Appendix in Table A.2.

Spectrograms Next to the 32 MFCCs that were originally used in the model, spectrograms were used as well. For that, the CNN-VAD spectrogram was used, as it has the same input shape (128, 32, 1). However, just like the CNN-VAD, each spectrogram only results in one label, either being speech or no speech. This means the layer merging the MFCC features was replaced with two dense layers with dropout to reshape the output from (128, 32, 2) to (1, 2).

7.3.3 Wav2vec2.0

Implementation

The model uses the standard Wav2vec model as a baseline, as implemented in [78]. Instead of the normal CTC head for the model, a custom head was created, which was adapted from [79]. The adaption is done as explained below.

The model has two tracks, one for embedding raw audio and one for spectrograms, as used in the CNN-VAD implementation from Silva et al. [2]. The raw audio is fed into two dense layers with a dropout layer in between. The spectrogram is fed into two convolutional neural networks with pooling layers following each. It is then flattened, and two dense layers are used to bring it to the same shape (768, 1). The two inputs are combined, and a dense and dropout layer are used to get the output.

As the model seemed to reach infinite gradients after training for several epochs, batch normalization layers were added at different positions in the network to combat that. Adding a batch normalization layer after each dense and pooling layer, the resulting model trained better. An overview of the resulting model is shown in the Appendix in Table A.3.

Adaptions

Two adaptions were performed on the main Wav2vec model, which is described in Section 7.3.3 above. The first tests the effectiveness of raw audio itself without any additional features, and the second changes the output, so the model can be used for emotion recognition.

Raw Audio Only To test the effectiveness of the raw audio itself, the spectrogram features were excluded for one adaption. Instead of the concatenation of the CNN-VAD model with the wav2vec audio embedding, only the latter was fed through the Dense layers.

Emotion Recognition For emotion recognition, the Wav2vec downstream model was adapted to the task. It uses similar features and layers as the Wav2vec VAD. The layers differ only after the concatenation layer. Instead of reducing the number of nodes for each dense layer to 500 and 250, it is kept at 1000. After the three dense layers and normalizations, the output is reshaped to (#of utterances, 7) to account for the seven possible emotions. In most cases, the number of utterances is one as each utterance is longer than 2.56s. However, as the VAD is not perfect and as windows might have the end of one utterance and the beginning of the next one, this number is not set to one, but instead uses CTC to map the labels to the utterances correctly. The details are shown in Table A.4 in the Appendix.

7.3.4 Evaluation Protocol

The QUT-NOISE-TIMIT [21] dataset mentions a protocol for evaluating the models, such that the results can be compared easily. As the training contains two recording sessions per location, the authors of the dataset recommend using session A as training and session B as testing data and vice-versa. That way, it can be guaranteed that the model is trained and tested on all locations but is also able to generalize well, as it faces unknown noises in the testing data [21].

The authors of the dataset [21] also propose to use the half-total-error rate (HTER) as the metric to measure how well the models are performing. The HTER is calculated by taking the average of the false-alarm rate (FAR) and miss rate (MR) as follows [21]:

$$HTER = \frac{FAR + MR}{2} \quad [\%] \tag{7.1}$$

The FAR is the portion of frames containing no speech that the model incorrectly detected as speech. It is calculated as [21]

$$FAR = 100 * \frac{\text{non-speech frames detected as speech}}{\text{all frames without speech}} \quad [\%] \tag{7.2}$$

The MR measures the rate of speech frames that were incorrectly not detected as speech, by using the following formula [21]

$$MR = 100 * \frac{\text{speech frames not detected as speech}}{\text{all frames containing speech}} \quad [\%] \tag{7.3}$$

This thesis will use this evaluation metric to be in line with existing research on this dataset.

7.4 Post-Processing

Post-processing is used to improve the prediction of a model by correcting unreasonable predictions, such as having one speech frame in the middle of non-speech frames.

In literature [2], this smoothing is done using the prediction of surrounding frames as input to the final label. In one paper, the average prediction of the two previous, the current frame, as well as the two following frames, are taken as the resulting label for the current frame [2]. In a paper by Sriskandaraja et al., the 101 surrounding frames are used, consisting of 50 frames in each direction plus the current one. If at least 40% of them are predicted as speech, the final label for the current frame is speech [80].

Additionally, the papers handle mixed label frames differently. Frames have a mixed label when the transition from speech to no-speech or the other way around happens in the middle of a frame instead of between frames. Silva et al. handle this by modifying the frame length at these points such that each frame is either completely speech or no-speech [2]. Sriskandaraja et al., on the other hand, implemented a 'hangover scheme' for these cases, which marks frames directly around predicted speech as speech as well. The reach of the hangover scheme includes the 300ms before the detected speech and 500ms after it [80].

Chapter 8

Results

Using the processes mentioned in the methodology, each model was tested on different features. Below, the results for each model are shown.

8.1 CNN-VAD

8.1.1 Simulation Setup

The model was trained for 100 epochs with a batch size of 32 and with Adam as the optimizer and binary cross-entropy as loss function.

For both training and testing, the QUT-NOISE-TIMIT dataset is used. This dataset has an equal split between no speech and speech segments, such that around 50% (±3%) of all samples from each recording and location are classified as speech. For training, all audio files from recording A were used. Twenty percent of the training set was used as validation using stratified sampling.

The trained model was then tested on all files from recording B. For every model, the best weights were saved and were used for the analysis.

8.1.2 Results

Table 8.1 shows the results of the different features on which the CNN-VAD [2] was tested.

The last row in Table 8.1 shows the results reported in the original experiment by Silva et al. [2]. In their paper, spectrograms are created for every 80ms with a window size of 160ms. Each frame of the spectrogram has a size of 2.5ms. These results are used as verification in this thesis to ensure that the methodology works as expected. Additionally, it is used as a reference to see how each model performed compared to that. Silva et al. [2] do not report results per SNR but instead for low, medium, and high noise. The latter two are the values reported in Table 8.1. Each result is

Features	SNR+05	SNR+00	SNR-05	SNR-10	SNR-15	SNR-20
Spectrogram 2.56s (7.2.1)	5.01%	11.76%	12.36%	**14.89%**	50%	50%
U-net Spectrogram (7.2.1)	**4.66%**	**7.74%**	**11.08%**	17.91%	50%	50%
Original model [2]	8.0%		14.5%		N.A.	

Table 8.1: Comparison of different adaptions of the CNN-VAD model. The values reported are the HTER measured for VAD for the specific signal-to-noise ratio on the QUT-NOISE-TIMIT dataset. In parentheses, the section in which the feature is explained is given.

the average of the two adjacent SNR values, e.g., an HTER of 14.5% means that the average result of SNR-05dB and SNR-10dB is 14.5%.

The first feature is a spectrogram with the same parameters as the original model, only an increased frame and window size. A spectrogram is extracted for every 2.56s window and each FFT window, also denoted as the frame, is 25ms long with a hop length of 20ms. The details are explained in Section 7.2.1. Increasing the frame and window size seems to be helpful, as each HTER is about 2% better than the original model. Only for environments with an SNR of -15dB or lower, the HTER is 50%, which means that only the majority label is predicted, and nothing is learned. This is the worst possible HTER that can be achieved.

The third feature tested has the same parameters as the previous one. The only difference is that labels are given for each frame instead of each 2.56s window. This makes the model less accurate for higher noise levels. Note, however, that HTER is calculated per label given, so having one label per 2.56s means, that the labeled speech is correct if at least one sample inside that 2.56s window is speech. For one label per frame, only one of the 128 labels would be correct if speech is predicted for the window. The latter is, therefore, much more detailed and can detect the start and end of speech more precisely. Using the feature with one label per 25ms frame results in a slightly (2-3%) worse HTER. For SNR-15 and lower, this feature is also unable to provide good results.

Additionally, raw audio and MFCC were tested as features but did not yield any improvements. While MFCC was slightly worse than spectrograms, the raw audio had an HTER of 19.7% for SNR+05dB and worse results for higher noise.

8.2 U-net

8.2.1 Simulation Setup

The model for both features was trained for 100 epochs using Adam as optimizer and binary cross-entropy as loss function. 32 was chosen as batch size. As done with CNN-VAD, the model was trained on all audio files from recording A, using 80% as training and 20% as the validation set. For splitting the recordings, stratified sampling is applied on the windows, such that speech occurs

equally likely in the train and validation set. The test set contains all audio files from recording B.

For every model, the best weights were saved and were used for the analysis.

8.2.2 Results

In Table 8.2, the different features for the U-Net model and the measured HTERs are shown.

Features	SNR+05	SNR+00	SNR-05	SNR-10	SNR-15	SNR-20
U-net Spectrogram (7.2.1)	3.19%	6.38%	9.89%	15.93%	**27.90%**	46.67%
MFCC (7.2.2)	**2.47%**	**3.78%**	**8.22%**	**13.02%**	30.67%	**39.75%**

Table 8.2: Comparison of different adaptions of the U-net model. The values reported are the HTER measured for VAD for the specific signal-to-noise ratio on the QUT-NOISE-TIMIT dataset. In parentheses, the section in which the feature is explained is given.

For the first feature, spectrograms are used with a window size of 2.56s and a label for each frame, as explained in Section 7.2.1. For an SNR of -5dB and higher, the model predicts the speech and no speech segments with an HTER of below 10%. Also, for SNR-10 and SNR-15, it is able to predict with 15.93% and 27.9% HTER, respectively. For SNR-20, the HTER is only slightly better than predicting the majority label.

The second feature, MFCC, uses the same parameters as the previous feature and also the same as in the paper by Gusev et al. [6]. Unfortunately, the authors did not publish any results, so it is unclear if they achieved similar error rates. For SNR-10 and above, the MFCC feature works slightly better than the spectrogram, with 0.5 to 3 percentage points HTER difference between the two. For SNR-15, the spectrogram is a bit better, and for SNR-20, the HTER is 7 percentage points lower than for the spectrogram.

Additionally, a third feature was tested to see the importance of the frame size and window size. A window size of 640ms and a frame size of 160 samples (10ms) were chosen. For SNR-10, this feature reported an HTER of 21%, consisting of an MR 33.67% and a FAR of 8.3%. As that was 6 percentage points higher than the other features, no further tests were done.

8.3 Wav2vec

8.3.1 Simulation Setup

Each combination of features and SNR was trained using AdamW as optimizer and cross-entropy as loss function. Each training ran for 60 epochs. This number was chosen because training takes much longer than for the previous models. 32 was chosen as the batch size.

For training, 300 audio files were randomly chosen from all locations in recording A and were split into the frames as specified for each specific feature. 20 percent of the training set was used as

validation using stratified sampling. 300 randomly chosen files split over all locations from recording B were used for testing. This was done due to the large amount of RAM needed to process the raw audio for all files.

For every model, the best weights were saved and were used for the analysis.

8.3.2 Results

Table 8.3 shows the best HTER measured for each feature.

Features	SNR+05	SNR+00	SNR-05	SNR-10	SNR-15	SNR-20
Spectrogram 640ms (7.2.1)	2.98%	4.10%	7.43%	10.16%	28.31%	50%
Spectrogram 2.56s (7.2.1)	**1.29%**	**1.96%**	**4.43%**	8.25%	21.03%	35.90%
U-net Spectrogram (7.2.1)	2.71%	3.34%	5.04%	**7.71%**	**18.56%**	**34.75%**

Table 8.3: Comparison of different adaptions of the Wav2vec model. The values reported are the HTER measured for VAD for the specific signal-to-noise ratio on the QUT-NOISE-TIMIT dataset. In parentheses, the section in which the feature is explained is given.

In total, the model was trained with three different spectrogram features. The first feature has a window of 640ms, as explained in Section 7.2.1. The other two features have a 2.56s window, in which the spectrograms are extracted. The difference between the latter two is that for the second feature one label per window is given, while for the third feature one label is given per frame. That means, for the latter, 128 labels are given for one 2.56s window.

As shown in Table 8.3, the best results were achieved for the two latter spectrogram features. Out of these two, it depends on the noise level to distinguish which one is better. However, in most cases, there is around one percentage point between the error rates measured for the features.

Additionally, wav2vec was tested using only raw audio using the 2.56s window, as that was performing the best. Testing the effectiveness of raw audio was done to understand how much the raw audio contributes to the results as well as to see if using spectrograms on top adds any value to the prediction.

8.4 Model Comparison

8.4.1 HTER per Location

Table 8.4 shows the HTER, MR, and FAR measured by the different models for a signal-to-noise ratio of -10dB. For each model, only the best feature is used. That means, for U-net, the MFCC feature is used, for CNN-VAD the 2.56s spectrogram, and for Wav2vec the U-net spectrogram.

As the Wav2vec model is only tested on 300 audio files, the other two models are also tested on the same 300 files to ensure that the results are comparable, and it is not just based on easier audio files.

	CNN-VAD			U-net			Wav2vec		
Location	HTER	MR	FAR	HTER	MR	FAR	HTER	MR	FAR
CAFE-CAFE	33.4%	31.3%	35.4%	27.8%	17.8%	37.8%	**14.7%**	12.1%	17.2%
CAFE-FOODCOURTB	33.5%	32.5%	34.5%	24.2%	26.4%	22.0%	**16.6%**	12.5%	20.6%
CAR-WINDOWNB	12.5%	12.4%	12.5%	7.6%	4.3%	11.0%	**6.3%**	3.1%	9.5%
CAR-WINUPB	10.7%	10.9%	10.4%	**4.2%**	2.5%	5.9%	4.9%	3.1%	6.6%
HOME-KITCHEN	12.8%	12.6%	13.0%	8.1%	4.6%	11.6%	**4.8%**	3.2%	6.4%
HOME-LIVINGB	23.3%	26.2%	20.3%	10.2%	9.2%	11.1%	**8.3%**	5.7%	11.0%
REVERB-CARPARK	15.7%	21.5%	9.9%	11.0%	10.6%	11.4%	**8.6%**	7.9%	9.4%
REVERB-POOL	15.9%	12.7%	19.1%	15.4%	11.7%	19.2%	**10.4%**	14.1%	6.8%
STREET-CITY	11.8%	14.6%	8.9%	6.8%	6.4%	7.1%	**6.0%**	5.5%	6.5%
STREET-KG	11.4%	14.4%	8.4%	7.5%	6.7%	8.2%	**5.9%**	6.0%	5.8%

Table 8.4: Comparison of the best feature per model on SNR-10dB for various locations of the test set on the QUT-NOISE-TIMIT corpus. Next to the HTER, the percentage of missed speech frames (MR) and the percentage of false detected speech frames (FAR) is given.

Table 8.4 shows how each model performs on the different locations given in the dataset with regards to the HTER, the MR (miss rate), and the FAR (false alarm rate). In the dataset, each location is visited twice. The values reported here are the mean of the two visits for recording B of each location. For example, for the CAFE-CAFE location, the dataset has a location CAFE-CAFE-1 and a location CAFE-CAFE-2, so these two are combined to get the values. Additionally, as mentioned above, only 300 audio files were taken, and only SNR-10 was tested extensively per location. Other noise levels, e.g., SNR-20, seem to give similar indications which locations are easier to detect noise and which are harder.

In general, looking at the HTER, all models seem to have the most problems with the CAFE-CAFE and CAFE-FOODCOURTB. The easiest locations (the best HTER) are in the car and on the street.

However, the HTER varies significantly for the different models. Looking at the CAFE-CAFE and CAFE-FOODCOURTB location, the CNN-VAD performs worst with an HTER of around 33%. The MR and FAR are in a two percentage point range around the HTER, with a higher FAR and a lower MR. The U-net has an HTER of 28% and 24%, respectively. For the food court, the MR is slightly higher with 26.4%. For the CAFE-CAFE, however, the MR and FAR diverge by 10 percentage points from the HTER. The MR is around 18%, while the FAR is at 38%. The Wav2vec model performs best with an HTER of 14.7% and 16.6%, respectively. The MR and FAR diverge by 3-4 percentage points from the HTER, with MR being smaller.

In general, CNN-VAD has the highest HTER for every location. The MR and FAR are also in most locations the highest of any model. Only for REVERB-POOL, the MR of Wav2vec is higher than the MR of CNN-VAD, and FAR of the U-net model is highest for the CAFE-CAFE, REVERB-CARPARK, and REVERB-POOL. For most locations, the MR and FAR of the CNN-VAD are close together (below 5 percentage points difference). The most significant outliers are REVERB-CARPARK, where the MR is 11.6 percentage points higher than the FAR, and REVERB-POOL, where the FAR is 6.4 points higher than the MR. Additionally, for HOME-LIVINGB and STREET-KG, the MR is also 6 points higher than the FAR. Looking at all locations, it is equally distributed whether the MR is higher or the FAR.

The U-net model is in the middle when comparing the HTER for each location. Only in CAR-WINUPB, the HTER is 0.7 percentage points better than Wav2vec. In all locations except CAFE-FOODCOURTB, the FAR is higher than the MR. The MR and FAR diverge most in CAFE-CAFE, CAR-WINDOWNB, HOME-KITCHEN, and REVERB-POOL. Contrary to CNN-VAD, however, the MR and FAR are close together for REVERB-CARPARK.

The Wav2vec model performed best in almost all locations. The distance between MR and FAR is in no location higher than 8.1 percentage points (CAFE-FOODCOURTB), with FAR being the higher value for all locations except REVERB-POOL (MR 7.3 points higher) and STREET-KG (0.2 points difference).

8.5 Speech Emotion Recognition

To test if the results are useful and improve the results for the follow-up tasks, the challenge of emotion recognition was chosen. Using the RAVDESS dataset, seven emotions can be detected.

The dataset is combined with the QUT-NOISE and, as done for the other datasets, separated into recording A and recording B. Both recordings have the same distribution over the locations, and both contain random speech segments from RAVDESS. Recording A is used for training with 80% used as training set and 20% used as validation set. Recording B is then used to test the trained model. The distribution of labels for each dataset is shown in Table 8.5. This is the utterance-window-wise class distribution, meaning that one label is given per utterance for each window. That could include multiple labels for one utterance if the utterance reaches over multiple windows.

Wav2vec was adapted to speech emotion recognition. The chosen model uses the 2.56s window size with utterance level features. Therefore, the VAD predictions for each frame are used to exclude segments without any speech detected. The detected speech utterances of each 2.56s window are then fed into the emotion recognition model to predict one of seven emotions, ranging from neutral/calm to surprised, with the emotions being in the order shown in the confusion matrix in Figure 8.1.

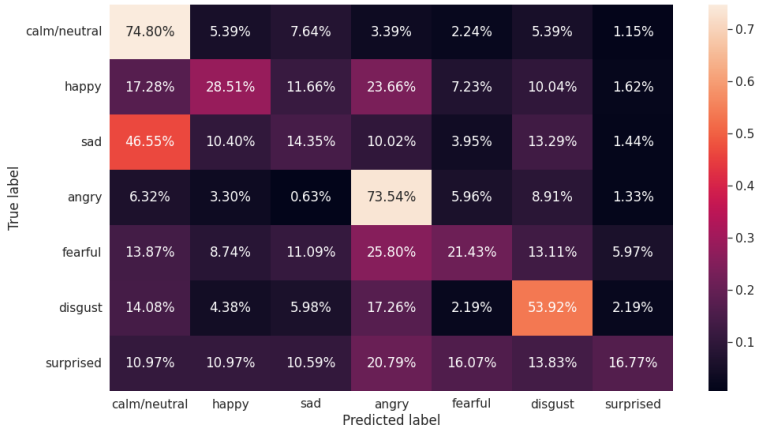
Dataset	calm/neutral	happy	sad	angry	fearful	disgust	surprised
Training	19.80%	12.86%	13.33%	14.51%	14.36%	12.68%	12.46%
Evaluation	19.75%	12.87%	13.35%	14.52%	14.34%	12.70%	12.46%
Testing	19.64%	11.77%	13.76%	14.43%	12.18%	15.40%	12.82%

Table 8.5: Distribution of the labels for speech emotion recognition distributed over the train/eval and test set on the QUT-NOISE-RAVDESS corpus. Train and evaluation dataset are split using stratified sampling based on the emotions using recording A. The test set is retrieved from recording B.

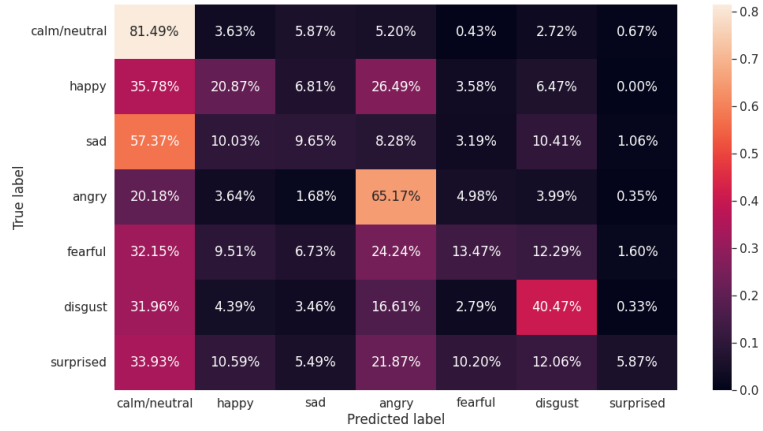
Figure 8.1 shows the results, both for Wav2vec as well as U-net. The results are reported for SNR-05dB.

Using Wav2vec for VAD as pre-processing step, the model achieved a 40% unweighted average recall and 41% unweighted average precision. As shown in Figure 8.1a, the model has the best predictions for calm/neutral, angry, and disgusted emotions. The highest error is predicting calm/neutral for speech segments where the speaker is, in reality, sad.

Using U-net for pre-processing, the model had an average unweighted recall of 34% and unweighted average precision of 39%. In Figure 8.1b, the confusion matrix is shown. For all emotions except anger and disgust, the model is most likely to predict calm/neutral.



(a) Confusion Matrix when using Wav2vec as pre-processing step.



(b) Confusion Matrix when using U-net as pre-processing step.

Figure 8.1: Comparison of the emotion recognition with Wav2vec and U-net VAD as pre-processing step for SNR-05dB on the RAVDESS dataset with added QUT-NOISE.

Chapter 9

Discussion

As shown in Chapter 8, the models performed very differently, and the features and sizes used changed the performance of the model significantly.

For CNN-VAD, using the features as used in the original paper [2] resulted in a similar HTER as reported in the paper. That shows that the general model creation and feature extraction are working as expected.

For the U-net model, which has an architecture close to the one created by Gusev et al. [6], the effectiveness of spectrograms and MFCCs is compared, using the same window size. As MFCCs are simply the log of spectrograms with applying a discrete cosine transform, the results are very similar as was expected. Of the two, the MFCC model seems to perform slightly better on every noise level except for SNR-10, which could be just an outlier. In general, the U-net model performed very well in low noise environments, having an HTER below 4% for any positive SNR. For negative SNR, the half-total-error-rate increased fast, making predictions for SNR of -15 and lower unusable in real-world applications.

For Wav2vec, different window sizes using both the raw audio as well as the spectrogram are tested. To test the advantage of raw audio in the model, the spectrogram part was removed for the last feature. Even though all Wav2vec models performed quite well, the best two used a window size of 2.56s. Further window sizes, including 320ms windows and 1028ms windows, were also tested on some noise levels but did not provide any improvements and are therefore not listed in the table. For 160ms window size, it also needs to be added that it is not possible to use the Wav2vec embedding for that window size, as it is below the minimum length required by the transformer of the wav2vec model.

Comparing the different window sizes, the results show that extending the window size improves the HTER. However, when only one label is used per window, increasing the window size also means

that each label represents a larger window. Instead of having a label for 25ms, one label is now predicted for 2.56s. That means even if the model predicts speech correctly, there could be a lot of non-speech segments inside the window. Additionally, the model will only ever be able to predict a 1.28s window in which speech starts or stops, it cannot be more precise. The 1.28s window is achieved by using 2.56s windows with 50% overlap. If speech is detected in one window but not in the previous or following window, a start/end of speech has been detected in the non-overlapping part of the window.

One way to improve the precision of these labels for long windows is to use one window per frame, as done with the third feature.

9.1 Label per Frame vs Label per Window

As mentioned in the results, the difference between the two 2.56s spectrogram features is not that high, and the better model depends on the noise level used. Both choices, having the label per window and the label per frame, show that the model is able to learn a good representation of speech.

One clear benefit of the latter is that the labels are more detailed. The HTER calculated for the first feature could be inflated due to the large window size. As the labels are created to classify any instance of speech immediately as speech, it would be enough for one of the frames inside the window to contain speech for the whole window to be correct. Having a label per frame also requires getting the correct frame where the speech starts. If speech is detected one frame too early or too late, the HTER goes up, while there is a good chance it would stay the same for the first feature. To check this hypothesis, the HTER is also calculated per sample. Calculating the HTER per sample always increases the HTER, as it is almost impossible to start every frame exactly when the speech starts or stops. As that will not happen, part of the samples in the frame will be declared as speech even though they are no speech and vice-versa.

	HTER per Label			HTER per sample		
Location	HTER	MR	FAR	HTER	MR	FAR
Labels frame	18.56%	17.20%	19.94%	22.50%	**31.15%**	**13.84%**
Labels window	21.03%	10.17%	31.89%	26.63%	**19.73%**	**33.53%**

Table 9.1: Comparison of the HTER, MR and FAR for labels per frame and labels per window on SNR-15dB on the QUT-NOISE-TIMIT corpus.

Comparing the different HTER for the label per frame model and the label per window model on a noise level of -15dB, shown in Table 9.1, shows that the HTER per sample is about 4-5 percentage points higher than the HTER recorded in Table 8.3. For the spectrogram using a label per frame, the HTER increased from 18.56% to 22.50%. For the other spectrogram feature, the HTER increased from 21.03% to 26.63%. Looking at the miss rate and false alarm rate, the results show the expected

pattern. If only one label per window is used, all frames in that window are considered speech or no speech. The FAR is, therefore, significantly higher than the MR. The MR is lower, as the speech usually continues for a few seconds, so it is likely to be picked up as speech at some point in the 2.56s window. For labels per individual frame, however, the results are reversed. As the window is a lot more narrow, only a small portion is labeled speech if speech is found. The FAR is thereby reduced. However, samples surrounding detected speech samples are not necessarily labeled as speech, so the MR is increased.

9.2 Comparisons between Models

9.2.1 CNN-VAD vs Wav2vec

Comparing the different models, CNN-VAD is the worst considering the overall performance. This is the expected result, as it has the simplest network architecture. As CNN-VAD is just a part of the Wav2vec, it is clear that Wav2vec should outperform CNN-VAD.

Looking at the model with the best feature from Table 8.1 and the best-performing feature from Table 8.3, it is shown that Wav2vec outperforms CNN-VAD for any noise level by a significant margin. Additionally, Wav2vec also provides valuable results for SNR of -15dB and lower, while CNN-VAD has to guess the majority label.

Table 8.4 shows the HTER, MR and FAR per location. As explained in the results, the locations CAFE-CAFE, CAFE-FOODCOURTB are the most problematic for all three models. They are so problematic because these locations also have speech in the background. The speech from the background is not labelled as speech, and the models, therefore, need to learn a distinction between speech in the background and a person speaking into the microphone directly. Wav2vec is able to distinguish between these types of speech better than CNN-VAD.

The reason why Wav2vec is performing better than CNN-VAD is that CNN-VAD is a very simple model compared to Wav2vec. CNN-VAD can thus not train on as fine details as Wav2vec. So it might not be able to pick up as well on the fine nuances between speech and background babbling. Additionally, Wav2vec has a larger context that is taken into consideration. For CNN-VAD, only two convolutional layers with average pooling between them allow for incorporating surrounding frames. Wav2vec has a far wider and more flexible context due to the attention heads. Context could be very important to distinguish between speech and background speech, as people talking in the background might move around or might face another way, which reduces the chance that everything is picked up by the microphone, while the person speaking is picked up by the microphone consistently during the whole speech segment. Therefore, having speech segments that are at least a few seconds could already be a good indicator that it is foreground speech. Lastly, if several people are speaking in the background and only the overlapping babbling is heard, having a model such as Wav2vec, which is already trained on human speech, could help distinguish actual spoken words from babbling sounds similar to speech.

Even the Wav2vec model without spectrograms, which uses just raw audio to predict the label works on SNR-15 and has better results than the CNN-VAD.

This means that Wav2vec improves the model that was created by Silva et al. [2]. It also means that using raw audio can be extremely helpful for the challenge of detecting speech. The reason for that is that the Wav2vec embedding of raw audio can maintain some details of speech that are not captured by classical features such as MFCC.

9.2.2 CNN-VAD vs U-net

For the U-net, it was not as clear if the U-net model could detect speech more accurately. The results, however, show that the U-net is better for almost every noise level. Only for SNR+15, CNN-VAD is better. But for that level of noise, almost all models have an HTER below 2%, so there is not a lot of room for improvement.

As this thesis is more focused on the high noise areas, the U-net model performs better. The reason why it performs better, even if the window and frame size are the same, is the U-shaped architecture of the model. Its architecture combines the details of each frame with the broad features of the whole window. That way, the model can focus on the individual frame while taking surrounding frames into consideration. As speech is usually a few seconds long and therefore longer than a few frames, having more context should be very helpful.

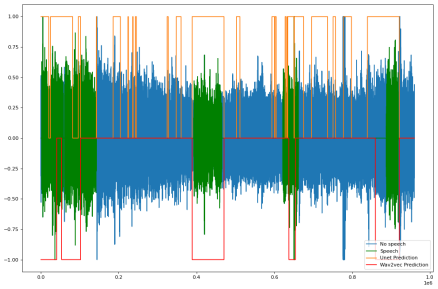
The CNN-VAD, on the other hand, uses only two CNNs compared to the six convolutional and five deconvolutional layers of the U-net. That likely makes the CNN-VAD model miss some details that appear in the later layers of the network. Additionally, neither the CNN-VAD nor the Lenet-5, on which the CNN-VAD is based, is meant to analyse large images. Lenet-5 was made for images with the size of 32 * 32. It is, therefore, likely, that the performance of the CNN-VAD would improve, especially when using noisier environments, when a complex network such as Alexnet [81] is used instead of using Lenet-5. Alexnet is, in this case, just an example with a more complex architecture that uses more CNNs and larger sizes to analyze larger images for image classification.

9.2.3 U-net vs Wav2vec

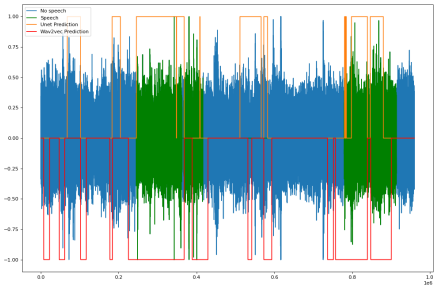
To best compare the U-net model with the Wav2vec model, it is helpful to look at the strength and weaknesses of each model. Both models performed better than the CNN-VAD model on all noise levels, and all locations for SNR-10 and Wav2vec performed better than U-net on most SNRs.

Generally, both models have the same window size and frame size, also using the same labels. That is where the similarity ends. Wav2vec uses a pre-trained network for raw audio to get an embedding of the audio, which can be trained to represent speech. Additionally, it uses the CNN-VAD architecture as an additional feature to improve the predictions. It uses several transformer blocks in the Wav2vec embedding, which allows a good representation of the context of a frame (the other frames in the window).

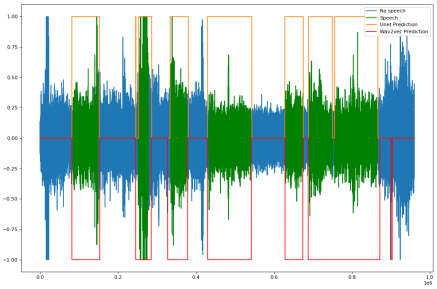
The U-net, on the other hand, only has convolutional and deconvolutional layers, where the first and last layers focus on the context across the whole window, while deeper CNN layers focus on extracting the details, using upsampling to increase the resolution. This, however, means that the context is only extracted using convolution, and therefore only the directly surrounding elements have an effect as the context. For example, for a filter size of (5, 5), the CNN will only consider the surrounding elements. Even if the image, where it extracts the context from, has a low resolution (and the filter, therefore, spans a wider area), it still only considers a smaller area than the transformer architecture, which uses multiple attention heads to get a context for various positions in the window.



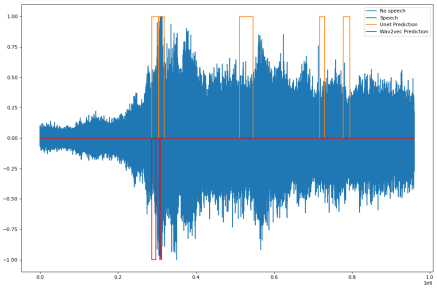
(a) Prediction for CAFE-CAFE.



(b) Prediction for CAFE-FOODCOURTB.



(c) Prediction for CAR-WINDOWNB.



(d) Prediction for STREET-KG without any speech.

Figure 9.1: Comparison of the predictions of U-net and Wav2vec for selected audio files from QUT-NOISE-TIMIT. The red line in the bottom is the prediction of Wav2vec, the orange line in the top the prediction of U-net. The green marked areas are speech, while the blue areas are just background noise at SNR -10dB.

One noise level where the Wav2vec model is significantly better than the U-net is SNR-20. Even though both models have a high error rate (34.75% and 39.75%), looking at the miss rate, it is even more clear that Wav2vec is the better approach. The 34.75% HTER for Wav2vec are calculated

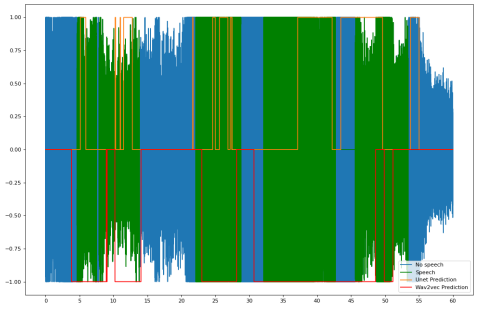


Figure 9.2: Prediction of U-net and Wav2vec for SNR-20dB. The red line in the bottom is the prediction of Wav2vec, the orange line in the top the prediction of U-net. The green marked areas are speech, while the blue areas are just background noise at SNR -20dB.

from a 40.8% miss rate and a 28.6% false alarm rate. For U-net, however, the 39.75% are achieved by having an MR of 64.8% and a FAR of 14.4%. Having a miss rate this high means that the results are unusable for any further analysis. If a model misses 40% of the speech, it is unknown what happens in the audio file. The undetected speech will likely contain important information, so using the detected speech as evidence is not doable for the police. Additionally, using the detected speech for follow-up processing such as speech emotion recognition could skew the results, as part of the speech with a different emotion could be missing.

The reason why both models are not performing that well is shown in Figure 9.2. As shown, looking at the 20 seconds in the middle of the recording, the raw audio is at the maximum value. This image also shows that Wav2vec actually predicts the general area of the speech accurately. It is just incorrect when predicting the exact start and end.

Figure 9.1 shows four predictions for the U-net and Wav2vec model for different locations with a SNR of -10dB. For all four pictures, the actual audio recording is displayed as a blue soundwave, with the speech segments highlighted in green. The orange line at the top shows the activity detected by the U-Net model, while the red line at the bottom displays the Wav2vec accuracy. For both lines, a value of one means that speech is detected, while zero means no speech is found.

The first two images, Figure 9.1a and Figure 9.1b, were recorded in the environments where all models struggled the most. Even though the location was difficult, both models managed to detect most of the audio sequences, only the start and finish of the audio was incorrect, either detecting speech too early or too late. Also, both models picked up speech in other mostly smaller segments throughout the audio. The reason for that is that the audio was recorded in a cafe or food court where several people were talking in the background. These locations are difficult for the models, as they contain foreground speech that shall be detected as well as speech in the background that should be ignored.

For easier environments, such as in a car, as shown in Figure 9.1c, both models detected the correct start and end of each speech segment. Only in the no speech parts were some frames incorrectly classified as speech.

For the last image in Figure 9.1d, the goal was to see how the models behave when no speech is given. Around 18 seconds into the audio file, both models detect speech. This was caused by a car driving past the microphone, which would not have been classified as speech by a human listener. The model might detect that as noise, as it was a short sound which was very different to the rest of the noises. Additionally, the sound created by the car motor was, just like the human speech, periodic. Besides that, Wav2vec did not detect any more speech, while U-Net detected voice activity in three other segments of the audio.

Both models performed well on most noise levels, but Wav2vec has the upper hand when comparing the miss rate and false alarm rate for several noise levels. However, for SNR-20dB, both models struggled to predict the speech correctly.

9.3 Architecture Choices

9.3.1 Labeling Frames

The labels were assigned to the frames as follows: all frames containing at least some speech were labelled "speech". This is different from the implementation from Silva et al. [2], who chose to cut the frame on label changes and, therefore, have some shorter frame sizes. That would, however, require additional padding after removing some audio, adding a lot of unnecessary noise to the features. Noise, in this case, is not meant as physical noise but as uncommon input which will act as noise to the network. For example, if speech ends at the beginning of a frame and the rest of the frame is either 0 or a repetition of that audio at the beginning, it would make the model learn something that will not occur in the real world.

In their experiment, Silva et al. did not add padding but instead extracted spectrograms relative to the size of the frame. So if one frame is cut off, the spectrogram will be more detailed, using smaller Fourier windows and a smaller hop length. That could also have some negative implications, as occasionally, these more detailed spectrograms will appear during training. If nothing is learned from them, the model cannot handle the edges of a speech signal. If something is learned, the question is whether that is beneficial for the model or if it reduces the overall accuracy by unnecessarily perturbing the weights.

The last reason why a different label creation was chosen is that it makes more sense when considering the testing. For the training set, it might be beneficial to not have mixed labels inside one frame, but for testing, where the correct labels are unknown, that is not possible. Therefore, all frames were labelled as one if at least one sample is speech. This does bring some risk with it, as it could be that exactly one sample is speech, and thereby a non-speech frame is labeled as speech. This way of labelling frames was chosen to ensure that the model also learns when exactly

the speech starts. As a higher false alarm rate is preferred over a higher miss rate, this approach makes sense, as it rather decides to predict speech for these edge frames than no speech.

It is currently unknown if this way of labelling had benefits compared to the other one. For future research, it could be interesting to compare them and find the best way of labeling speech. Most likely, labeling each frame by using the majority of the speech samples will be the best performing method.

9.3.2 Smoothing

As shown in Figure 9.1, the results of the U-net and Wav2vec models are both more or less accurate depending on the location of the background noise. For less difficult locations, both models can detect speech very well, and even the beginning and end of a speech segment are detected quite accurately, as shown in Figure 9.1c. However, for other predictions, the models were uncertain, and the label switched several times, as shown in Figure 9.1b.

This could be improved using post-processing. Different post-processing techniques are explained in Section 7.4. As explained, all predictions in this thesis are smoothened using the surrounding four frames as well as the current frame. However, that does not seem to be not enough, as too many speech segments are missing from the predictions. For the Police, these predictions would not be sufficient to use as evidence when analyzing interrogations or observations due to the high margin of error when having babble in the background.

Therefore, a larger smoothing range could be chosen. Then, it needs to be fine-tuned what the optimal range is without missing small segments of speech already. Usually, no speech is shorter than several hundred milliseconds to a few seconds. It is, therefore, likely that the smoothing range can be increased without any issues. One example where that might be problematic is when a person is shouting something quickly, e.g. 'help' or 'Aaah'. Shouting takes less than a second and could be the deciding factor when considering that the results of this thesis are of interest to the Dutch Police. For the Police, finding even short words like help might help them understand the situation much better. Additionally, even for other use cases, such as digital assistants, finding these short words or exclamations might be useful to be able to call for help.

One other possibility to smoothen the results is using a hangover scheme to label a wide segment around a speech fragment to speech [80]. The details about this approach are explained in Section 7.4. This could be an option to reduce the MR, as currently, for both models, many errors are being made close to a found speech segment. In these cases, either the actual speech stopped several frames before the prediction ended or the speech went on while the model already predicted no speech again. This will significantly increase the FAR for both models, though.

9.4 Emotion Recognition

As shown in Section 8.5 in the results, using Wav2vec as pre-processing step for noisy environments improves the overall performance of the model. Even though the output is not great when using an SNR of -05dB, using Wav2vec helped increase the average unweighted recall from 34% to 40%.

Both models seem to be able to distinguish calm/neutral, angry, and disgust a lot better than any other emotion. That might be because the latter two emotions are very distinct in tone compared to neutral speech. Calm/neutral is predicted most often in general, and therefore it is not surprising that most of the neutral and calm emotions are correctly predicted.

A Wav2vec downstream model was chosen for emotion recognition, as Wav2vec was the best candidate for the VAD, so it might also be good for emotion recognition. Previous literature has shown that Wav2vec downstream models for emotion recognition are successful [65]. However, they have not tested it on noisy data. In this thesis, using emotion recognition was simply a proof-of-concept that the VAD outputs are beneficial for follow-up tasks. As the overall recall and precision increased when using Wav2vec as pre-processing step, this proof-of-concept was successful. However, even using Wav2vec as pre-processing has an F1-score of only 40%. That is because the emotion recognition model used was not fine-tuned, and not many experiments were done to test whether different layers/features work better. This improvement of the emotion recognition results should be tested in future research using different models and features.

Chapter 10

Conclusion

As shown, contrary to CNN-VAD [2], the U-net model by Gusev et al. [6], and the Wav2vec model [5], which was adapted to the task of voice activity detection, both performed very well on the QUT-NOISE-TIMIT dataset, with the latter model being the best out of all three models tested. Additionally, through testing different window sizes, the size of 2.56s per window is found to be the best performing. There are, however, no significant differences between the types of features. For most models, using MFCC or Spectrograms did not influence the error rates significantly. Only the raw audio was not as successful for the CNN-VAD and U-net, but raw audio was the important part that made the Wav2vec model so effective.

Looking back at the research question in Chapter 2, this thesis showed that Wav2vec and raw audio can be used to perform VAD in high noise environments successfully. Compared to the other two models, the HTER of Wav2vec is at least ten percentage points better on noisy speech of -10dB and -15dB. For SNR-10dB, Wav2vec achieved a 92.3% accuracy on the test set, and on SNR-15dB, an accuracy of 81.4% was measured.

For the Dutch Police, this means that Wav2vec can be tested on the actual recordings to see the effectiveness on a real-world dataset. The results should be improved over existing models, and the manual labeling can be reduced. However, the miss rate is still relatively high, so some more work will need to be done before the model can be implemented. In the next section, an improvement to reduce the miss rate is proposed. Additionally, Wav2vec usually manages to detect speech for at least some part of each several-second long speech segment. So looking at the surrounding frames when speech is found might help reduce the miss rate even more. The police plans to use the model for various tasks. One example is long interviews, where the recording goes on for several hours, and only the speech segments should be preserved. Another example is in-car conversations. QUT-NOISE-TIMIT provides both, noise in a car with windows down and windows up, and the resulting Wav2vec model performs great in both locations. Wav2vec should be perfectly suited for both tasks for the police.

Regarding speech emotion recognition, the results show that using Wav2vec improves the overall results by making the predictions somewhat more stable. For higher noise, however, emotion recognition performed badly with and without using Wav2vec as pre-processing step. More work needs to be done before environments containing that much noise can be used for other speech processing tasks. But that is a research question for a future project, where different models can be trained on a specific speech processing task in a noisy environment.

Overall, pre-trained self-supervised models such as Wav2vec seem to be useful to get a good representation of speech, and raw audio as a feature helped improve the predictions significantly. Even though training a downstream model for Wav2vec is more time-intensive than training the other models and using raw audio requires more memory, the model is quite fast predicting labels once it is trained (it takes 21s to predict labels for 100 minutes of audio). Furthermore, the model is more accurate than the other models, and thus it is useful to spend the time training the model.

10.1 Directions for Future Research

Even though the models discussed in this thesis yielded good results for various noise levels, several things can be improved in further research. Some of these things are already mentioned as points for improvement of the current architecture in previous chapters. The first improvement, which should be done to compare the results achieved on this dataset with other state-of-the-art models, is to test the models on different VAD datasets. There are currently not many models focusing on these high levels of noise, and most datasets do not support these noise levels. Therefore, it might be easiest to test the other models on the QUT-NOISE-TIMIT dataset and compare the results. Also, most models focus on less noisy data, so they would need to be adapted to work well on this noise level.

Another improvement, which is more specific to how MR and FAR are balanced now, is creating a threshold as a post-processing step to reduce the MR as much as possible while not increasing the HTER too much. This threshold would assign all frames with a probability above the threshold to a specific class. By lowering the threshold, more frames are detected as speech, and by increasing the threshold, fewer frames are detected. Therefore, lowering would reduce the miss rate and increasing it would reduce the false alarm rate but increase the miss rate, as frames, where the model is not as certain, would be skipped. For the police, it is essential to find all speech segments without missing any parts. These segments can then be used as evidence from interrogations or conversations. Therefore, lowering the miss rate by decreasing the threshold would be a good way for the police to ensure no missing evidence. For other use cases, this threshold could, of course, be adapted to represent the task better.

This thesis tested each model on different features, window sizes, and many combinations. However, due to constraints with time, as it takes a few hours to train each model, not every combination was tested intensively. Additionally, other window sizes or increasing the frame size but keeping the window size the same were not tested. The latter could help distinguish whether the actual size

of the frame is the important part, why the models work so well or if the 2.56s window gave them enough context to work so well. Furthermore, no window size above 2.56s was tested. It is possible that increasing that size even more might improve the model as well, as the context is broader. It could also be that the context is too high, as many speech segments are only a few seconds long, and the accuracy decreases.

Furthermore, features used in ComboSAD seem to work very well in high noise areas. Some of the features used there were tested in this thesis but did not provide any benefits, while others were not tested. Combining the features with the Wav2vec model could improve the detection even further.

The dataset used for this thesis used electronically induced noisy speech, where the clean speech was simply added to a noisy background. To ensure that the models are not focusing on the difference in reverb between the clean speech and the noisy background, the resulting model should also be tested on acoustically generated noisy speech, e.g., recorded in a cafe at a similar noise level.

For Wav2vec, it was only possible to train and test the model on 300 audio files due to the large amount of RAM needed by the raw audio for the model. To get more accurate results, it would be helpful to test this again on the whole dataset.

Bibliography

[1] Tom Bäckström. *Voice activity detection (VAD)*. Aug. 2020. URL: https://wiki.aalto.fi/pages/viewpage.action?pageId=151500905.

[2] Diego Augusto Silva et al. "Exploring convolutional neural networks for voice activity detection". In: *Cognitive Technologies*. Springer, 2017, pp. 37–47.

[3] Daniel Stoller, Sebastian Ewert, and Simon Dixon. "Wave-u-net: A multi-scale neural network for end-to-end audio source separation". In: *arXiv preprint arXiv:1806.03185* (2018).

[4] Juntae Kim and Minsoo Hahn. "Voice activity detection using an adaptive context attention model". In: *IEEE Signal Processing Letters* 25.8 (2018), pp. 1181–1185.

[5] Alexei Baevski et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations". In: *arXiv preprint arXiv:2006.11477* (2020).

[6] Aleksei Gusev et al. "Deep speaker embeddings for far-field speaker recognition on short utterances". In: *arXiv preprint arXiv:2002.06033* (2020).

[7] Matthew B Hoy. "Alexa, Siri, Cortana, and more: an introduction to voice assistants". In: *Medical reference services quarterly* 37.1 (2018), pp. 81–88.

[8] Steven L Gay and Jacob Benesty. *Acoustic signal processing for telecommunication*. Vol. 551. Springer Science & Business Media, 2012.

[9] Tom Bäckström. *Applications and systems structures*. June 2021. URL: https://wiki.aalto.fi/display/ITSP/Applications+and+systems+structures.

[10] Xiao-Lei Zhang and Ji Wu. "Deep belief networks based voice activity detection". In: *IEEE Transactions on Audio, Speech, and Language Processing* 21.4 (2012), pp. 697–710.

[11] Xiao-Lei Zhang and DeLiang Wang. "Boosted deep neural networks and multi-resolution cochleagram features for voice activity detection". In: *Fifteenth annual conference of the international speech communication association*. 2014.

[12] R Tucker. "Voice activity detection using a periodicity measure". In: *IEE Proceedings I (Communications, Speech and Vision)* 139.4 (1992), pp. 377–380.

[13] LR Rabiner and MR Sambur. "Algorithm for determining the endpoints of isolated utterances". In: *The Journal of the Acoustical Society of America* 56.S1 (1974), S31–S31.

[14] Dong Enqing et al. "Applying support vector machines to voice activity detection". In: *6th International Conference on Signal Processing, 2002.* Vol. 2. IEEE. 2002, pp. 1124–1127.

[15] Risanuri Hidayat et al. "Denoising speech for mfcc feature extraction using wavelet transformation in speech recognition system". In: *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE).* IEEE. 2018, pp. 280–284.

[16] Xiao-Lei Zhang and Ji Wu. "Denoising deep neural networks based voice activity detection". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing.* IEEE. 2013, pp. 853–857.

[17] Javier Ramirez et al. "Voice activity detection with noise reduction and long-term spectral divergence estimation". In: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing.* Vol. 2. IEEE. 2004, pp. ii–1093.

[18] S. Akarsh and K. Avinash. *Speech Processing, speech synthesis amp; speech recognition – S.Akarsh amp; K.Avinash.* May 2010. URL: http://www.yuvaengineers.com/speech-processing-speech-synthesis-speech-recognition-s-akarsh-k-avinash/.

[19] Lawrence Rabiner. *Introduction to Digital Speech Processing.* 2010. URL: https://web.ece.ucsb.edu/Faculty/Rabiner/ece259/digital%5C%20speech%5C%20processing%5C%20course/lectures_new/Lecture%5C%201_fall_2010_6tp.pdf.

[20] Keith Johnson. *Acoustic and auditory phonetics.* 3rd ed. Wiley-Blackwell, 2012.

[21] David Dean et al. "The QUT-NOISE-TIMIT corpus for evaluation of voice activity detection algorithms". In: *Proceedings of the 11th Annual Conference of the International Speech Communication Association.* International Speech Communication Association. 2010, pp. 3110–3113.

[22] MediaCollege. *How sound waves work.* URL: https://www.mediacollege.com/audio/01/sound-waves.html.

[23] Tom Bäckström. *Waveform.* Sept. 2020. URL: https://wiki.aalto.fi/display/ITSP/Waveform.

[24] Analog.com. *Chapter 20: Analog to Digital Conversion.* Jan. 2021. URL: https://wiki.analog.com/university/courses/electronics/text/chapter-20.

[25] MasteringTheMix. *What is the sample rate?* Mar. 2016. URL: https://www.masteringthemix.com/blogs/learn/113159685-sample-rates-and-bit-depth-in-a-nutshell.

[26] Casey O'Callaghan. "Auditory Perception". In: *The Stanford Encyclopedia of Philosophy.* Ed. by Edward N. Zalta. Fall 2021. Metaphysics Research Lab, Stanford University, 2021.

[27] *Decibel.* URL: https://www.britannica.com/science/decibel.

[28] Robert Sheldon and John Burke. *What is signal-to-noise ratio and how is it measured?* Aug. 2021. URL: https://www.techtarget.com/searchnetworking/definition/signal-to-noise-ratio.

[29] Robert Mannell. 2008. URL: https://www.mq.edu.au/about/about-the-university/our-faculties/medicine-and-health-sciences/departments-and-centres/department-of-linguistics/our-research/phonetics-and-phonology/speech/phonetics-and-phonology/phoneme-and-allophone.

[30] Jake Pool. *44 phonemes in English and other sound blends.* May 2021. URL: https://magoosh.com/english-speaking/44-phonemes-in-english-and-other-sound-blends/.

[31] Tom Bäckström. *Windowing.* Aug. 2016. URL: https://wiki.aalto.fi/display/ITSP/Windowing.

[32] Simon G. Braun. "Windows". In: *Encyclopedia of vibration.* Elsevier, 2001, pp. 1587–1595.

[33] Leland Roberts. *Understanding the mel spectrogram.* Mar. 2020. URL: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53.

[34] nti-audio. *View of a signal in the time and frequency domain.* URL: https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft.

[35] Pratheeksha Nair. *The dummy's guide to MFCC.* July 2018. URL: https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd.

[36] Jason Brownlee. *What is deep learning?* Aug. 2020. URL: https://machinelearningmastery.com/what-is-deep-learning/.

[37] Arthur Arnx. *First Neural Network for beginners explained (with code).* Jan. 2019. URL: https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf.

[38] Isha Salian. *SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?* Aug. 2018. URL: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/.

[39] Nilesh Vijayrania. *Self-Supervised learning methods for computer vision.* Jan. 2021. URL: https://towardsdatascience.com/self-supervised-learning-methods-for-computer-vision-c25ec10a91bd.

[40] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[41] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET).* Ieee. 2017, pp. 1–6.

[42] Denny Britz. "Understanding convolutional neural networks for NLP". In: *URL: http://www. wildml. com/2015/11/understanding-convolutional-neuralnetworks-for-nlp/(visited on 11/07/2015)* (2015).

[43] Alexander Rush. "The Annotated Transformer". In: *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 52–60. DOI: 10.18653/v1/W18-2509. URL: https://www.aclweb.org/anthology/W18-2509.

[44] Ashish Vaswani et al. "Attention is all you need". In: *arXiv preprint arXiv:1706.03762* (2017).

[45] Jay Alammar. "The illustrated transformer". In: *GitHub Blog, Online: http://jalammar. github. io/illustrated-transformer* (2018).

[46] Eduardo Muñoz. *Attention is all you need: Discovering the Transformer paper*. Feb. 2021. URL: https://towardsdatascience.com/attention-is-all-you-need-discovering-the-transformer-paper-73e5ff5e0634.

[47] *A simple overview of RNN, LSTM and attention mechanism*. Jan. 2021. URL: https://medium.com/swlh/a-simple-overview-of-rnn-lstm-and-attention-mechanism-9e844763d07b.

[48] Rani Horev. *BERT Explained: State of the art language model for NLP*. Nov. 2018. URL: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[49] Simon Graf et al. "Features for voice activity detection: a comparative analysis". In: *EURASIP Journal on Advances in Signal Processing* 2015.1 (2015), pp. 1–15.

[50] Theodoros Giannakopoulos and Aggelos Pikrakis. "Chapter 4-audio features". In: *Introduction to audio analysis* (2014), pp. 59–103.

[51] Hidefumi Kobatake, Katsuhisa Tawa, and Akira Ishida. "Speech/nonspeech discrimination for speech recognition system under real life noise environments". In: *International Conference on Acoustics, Speech, and Signal Processing,* IEEE. 1989, pp. 365–368.

[52] JA Haigh and JS Mason. "Robust voice activity detection using cepstral features". In: *Proceedings of TENCon'93. IEEE Region 10 International Conference on Computers, Communications and Automation*. Vol. 3. IEEE. 1993, pp. 321–324.

[53] Jongseo Sohn, Nam Soo Kim, and Wonyong Sung. "A statistical model-based voice activity detection". In: *IEEE signal processing letters* 6.1 (1999), pp. 1–3.

[54] Joon-Hyuk Chang, Nam Soo Kim, and Sanjit K Mitra. "Voice activity detection based on multiple statistical models". In: *IEEE Transactions on Signal Processing* 54.6 (2006), pp. 1965–1976.

[55] Dongwen Ying et al. "Voice activity detection based on an unsupervised learning framework". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.8 (2011), pp. 2624–2633.

[56] Thad Hughes and Keir Mierle. "Recurrent neural networks for voice activity detection". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 7378–7382.

[57] DeLiang Wang and Guy J. Brown. *Computational auditory scene analysis principles, algorithms, and applications*. Wiley interscience, 2006.

[58] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[59] Shipra Saxena. *Lenet-5: Lenet-5 architecture: Introduction to lenet-5*. Mar. 2021. URL: https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/.

[60] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[61] Romain Hennequin et al. "Spleeter: a fast and efficient music source separation tool with pre-trained models". In: *Journal of Open Source Software* 5.50 (2020), p. 2154.

[62] Min Lin, Qiang Chen, and Shuicheng Yan. "Network in network". In: *arXiv preprint arXiv:1312.4400* (2013).

[63] Lu Lu et al. "Dying relu and initialization: Theory and numerical examples". In: *arXiv preprint arXiv:1903.06733* (2019).

[64] Danqing Liu. *A Practical Guide to ReLU*. Nov. 2017. URL: https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7.

[65] Leonardo Pepino, Pablo Riera, and Luciana Ferrer. "Emotion Recognition from Speech Using Wav2vec 2.0 Embeddings". In: *arXiv preprint arXiv:2104.03502* (2021).

[66] Zhiyun Fan et al. "Exploring wav2vec 2.0 on speaker verification and language identification". In: *arXiv preprint arXiv:2012.06185* (2020).

[67] Ekin Tiu. *Understanding Latent Space in Machine Learning*. Feb. 2020. URL: https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d.

[68] Wei-Ning Hsu. *Learning Latent Representations for Speech Generation and Transformation*. URL: http://people.csail.mit.edu/wnhsu/vae_speech/.

[69] Paul Kiparsky. *Linguistic universals and linguistic change*. De Gruyter, 2012.

[70] Alexei Baevski, Steffen Schneider, and Michael Auli. "vq-wav2vec: Self-supervised learning of discrete speech representations". In: *arXiv preprint arXiv:1910.05453* (2019).

[71] Brian Williams. *Contrastive Loss Explained*. Mar. 2020. URL: https://towardsdatascience.com/contrastive-loss-explaned-159f2d4a87ec.

[72] Alex Graves et al. “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks”. In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 369–376.

[73] John S Garofolo et al. “DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1”. In: *NASA STI/Recon technical report n* 93 (1993), p. 27403.

[74] Rosana Ardila et al. “Common voice: A massively-multilingual speech corpus”. In: *arXiv preprint arXiv:1912.06670* (2019).

[75] Artificial Intelligence Signal Processing and Vision Technologies. Oct. 2017. URL: https://research.qut.edu.au/saivt/databases/qut-noise-databases-and-protocols/.

[76] *Understanding the Mel Spectrogram*. https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53. Accessed: 2021-11-22.

[77] Haytham Fayek. *Speech Processing for Machine Learning: Filter Banks, Mel-frequency cepstral coefficients (mfccs) and what's in-between*. Apr. 2016. URL: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html.

[78] Patrick von Platen. *Fine-tune WAV2VEC2 for English ASR in hugging face with transformers*. Mar. 2021. URL: https://huggingface.co/blog/fine-tune-wav2vec2-english.

[79] Mehrdad Farahani. *M3HRDADFI/WAV2VEC2-xlsr-greek-speech-emotion-recognition · hugging face*. URL: https://huggingface.co/m3hrdadfi/wav2vec2-xlsr-greek-speech-emotion-recognition.

[80] Kaavya Sriskandaraja et al. “A model based voice activity detector for noisy environments”. In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[81] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

Appendices

Appendix A

Model Architectures

Table A.1: Details of CNN-VAD architecture.

Layer	Filter size	Stride	Feature Map size	#Parameters	Activation Function
Input			(128, 32, 1)		
Conv 1	5 * 5	1	(128, 32, 20)	520	Tanh
Average Pooling	2 * 2	2	(64, 16, 20)	0	
Conv 2	5 * 5	1	(64, 16, 50)	25050	Tanh
Average Pooling	2 * 2	2	(32, 8, 50)	0	
Flatten			12800	0	
Dense 1			500	6400500	ReLU
Dense 2			2	1002	Softmax

Table A.2: Details of U-Net architecture.

Layer	Filter size	Stride	Feature Map size	#Parameters	Activation Function
Input			(128, 32, 1)		
Conv 1	5 * 5	2	(64, 16, 8)	208	Leaky ReLU
Batch norm			(64, 16, 8)	32	
Conv 2	5 * 5	2	(32, 8, 16)	3216	Leaky ReLU
Batch norm			(32, 8, 16)	64	
Conv 3	5 * 5	2	(16, 4, 32)	12832	Leaky ReLU
Batch norm			(16, 4, 32)	128	
Conv 4	5 * 5	2	(8, 2, 64)	51264	Leaky ReLU
Batch norm			(8, 2, 64)	256	
Conv 5	5 * 5	2	(4, 1, 128)	204928	Leaky ReLU
Deconv 1	5 * 5	2	(8, 2, 64)	204864	Leaky ReLU
Batch norm			(8, 2, 64)	256	
Dropout			(8, 2, 64)	0	
Concat			(8, 2, 128)	0	
Deconv 2	5 * 5	2	(16, 4, 32)	102432	Leaky ReLU
Batch norm			(16, 4, 32)	128	
Dropout			(16, 4, 32)	0	
Concat			(16, 4, 64)	0	
Deconv 3	5 * 5	2	(32, 8, 16)	25616	Leaky ReLU
Batch norm			(32, 8, 16)	64	
Dropout			(32, 8, 16)	0	
Concat			(32, 8, 32)	0	
Deconv 4	5 * 5	2	(64, 16, 8)	6408	Leaky ReLU
Batch norm			(64, 16, 8)	32	
Dropout			(64, 16, 8)	0	
Concat			(64, 16, 16)	0	
Deconv 5	5 * 5	2	(128, 32, 1)	401	Leaky ReLU
Batch norm			(128, 32, 1)	4	
Conv 6	4 * 4	1	(128, 32, 2)	34	
Multiply			(128, 32, 2)	0	
Glob avg pool			(128, 1, 2)	0	
Reshape			(128, 2)	0	
Dense			(128, 2)	6	Softmax

Table A.3: Details of Wav2vec downstream model architecture.

Layer	Filter size	Stride	Feature Map size	#Parameters	Activation Function
Wav2vec Embedding			(768)		
Dense			(768)	589824	
Batch norm			(768)	384	
Dropout			(768)		
Dense			(500)	384000	
Batch norm			(500)	250	
Spectrogram			(128, 32, 1)		
Conv 1	5 * 5	1	(128, 32, 20)	520	Tanh
Average Pooling	2 * 2	2	(64, 16, 20)	0	
Batch norm			(64, 16, 20)	80	
Conv 2	5 * 5	1	(64, 16, 50)	25050	Tanh
Average Pooling	2 * 2	2	(32, 8, 50)	0	
Batch norm			(32, 8, 50)	200	
Reshape			(32, 8 * 50)		
BLSTM			(32, 400)		
Flatten			(76800)	0	
Dense 1			(500)	38400000	ReLU
Dropout			(500)		
Concat			(1000)		
Dense			(500)	500000	
LayerNorm			(500)		
Dense			(500)	250000	
LayerNorm			(500)		
Dense			(250)	125000	
LayerNorm			(250)		
Dense			(256)	64000	
Reshape			(128, 2)		Softmax

Table A.4: Details of Wav2vec downstream model for Emotion Recognition architecture.

Layer	Filter size	Stride	Feature Map size	#Parameters	Activation Function
Wav2vec Embedding			(768)		
Dense			(768)	589824	
Batch norm			(768)	384	
Dropout			(768)		
Dense			(500)	384000	
Batch norm			(500)	250	
Spectrogram			(128, 32, 1)		
Conv 1	5 * 5	1	(128, 32, 20)	520	Tanh
Average Pooling	2 * 2	2	(64, 16, 20)	0	
Batch norm			(64, 16, 20)	80	
Conv 2	5 * 5	1	(64, 16, 50)	25050	Tanh
Average Pooling	2 * 2	2	(32, 8, 50)	0	
Batch norm			(32, 8, 50)	200	
Reshape			(32, 8 * 50)		
BLSTM			(32, 400)		
Flatten			(76800)	0	
Dense 1			(500)	38400000	ReLU
Dropout			(500)		
Concat			(1000)		
Dense			(1000)	1000000	
LayerNorm			(1000)		
Dense			(1000)	1000000	
LayerNorm			(1000)		
Dense			(1000)	1000000	
LayerNorm			(1000)		
Dense			(1024)	1024000	
Reshape			(#utterances, 7)		Softmax

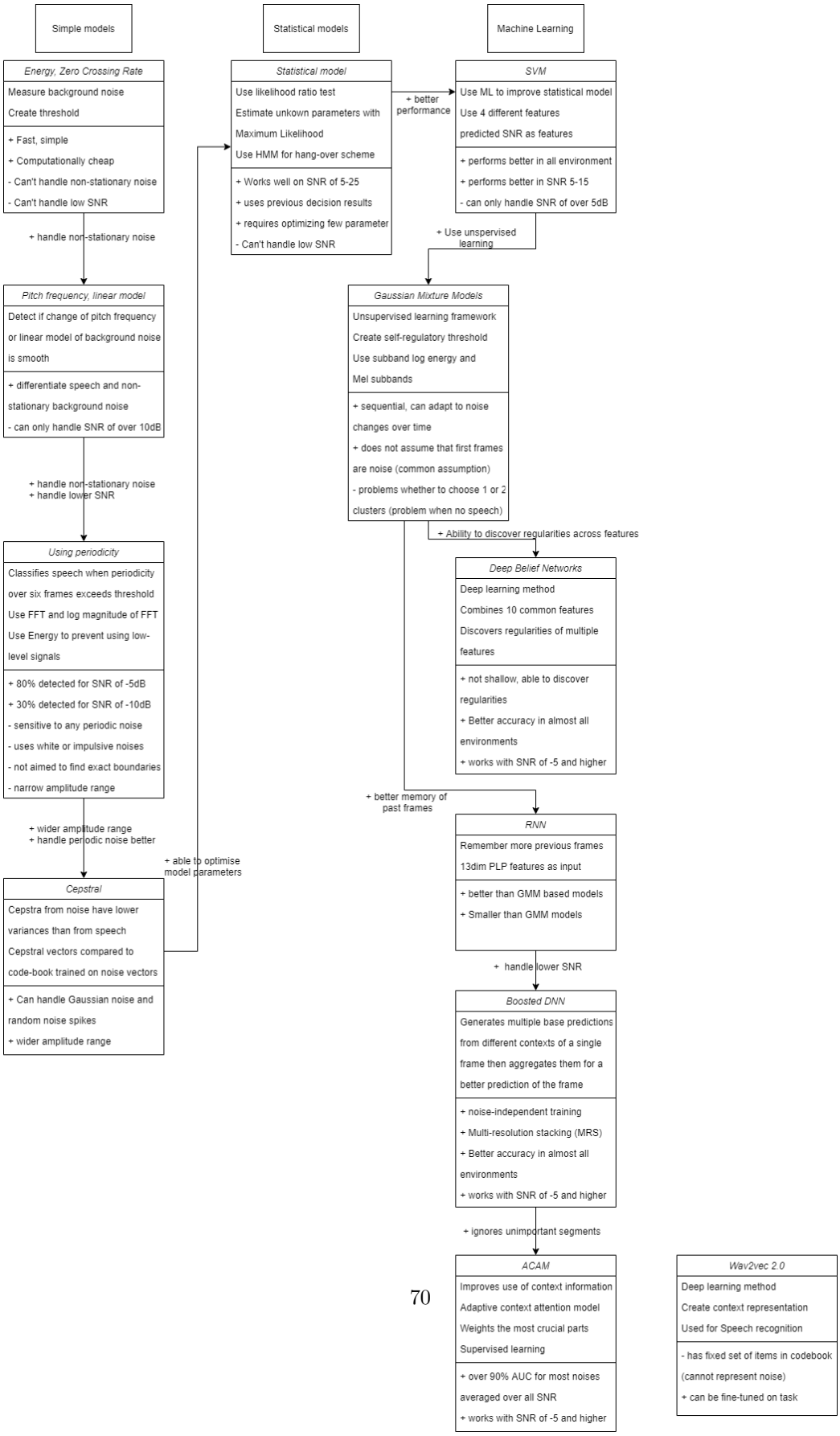


Figure A.1: Comparison of previous models for voice activity detection. The models can be divided into simple, statistical and machine learning models. Each model shows the key features of the model and the pros and cons of that model compared to the preceding and succeeding models.