Compressed, Real-Time Voice Activity Detection with Open Source Implementation for Small Devices

Lasse Ryge Andersen*
lasse2507a@gmail.com
Aalborg University
Denmark

Lukas Juel Jacobsen*
lugialukas@live.dk
Aalborg University
Denmark

David Campos
dgcc@cs.aau.dk
Aalborg University
Denmark

ABSTRACT

This paper proposes a real-time voice activity detection (VAD) system that utilizes a compressed convolutional neural network (CNN) model. On general-purpose computers, the system is capable of accurately classifying the presence of speech in audio with low latency. Whereas, when implemented on small devices, the system is showing higher latency, which is presumably an indication of high-load computations in the preprocessing steps. The results of the evaluation indicate that the proposed VAD system is an improvement over the existing solutions, in terms of reducing the model size and improving the level of accuracy among different evaluation metrics. Furthermore, the proposed VAD system offers an extension of the applicability by training the CNN model on a different and more diverse data set. Moreover, the proposed architecture is capable of being compressed to approximately one-eleventh of the size, facilitating eventual deployment on small devices. In contrast to existing closed VAD solutions, the entire pipeline of the proposed VAD system is developed in Python and made available as open source, ensuring the verifiability and accessibility of the work.

CCS CONCEPTS

• **Computing methodologies → Supervised learning by classification**.

KEYWORDS

voice activity detection, model compression, convolutional neural network, real-time VAD, open source VAD

ACM Reference Format:
Lasse Ryge Andersen, Lukas Juel Jacobsen, and David Campos. 2023. Compressed, Real-Time Voice Activity Detection with Open Source Implementation for Small Devices. In *8th international Workshop on Sensor-Based Activity Recognition and Artificial Intelligence (iWOAR 2023), September 21–22, 2023, Lübeck, Germany*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3615834.3615835

*Equal contributions.

iWOAR 2023, September 21–22, 2023, Lübeck, Germany

ACM ISBN 979-8-4007-0816-9/23/09…$15.00
https://doi.org/10.1145/3615834.3615835

1 INTRODUCTION

Voice activity detection (VAD) is a technique used in speech processing to detect when human speech is present or absent in audio. VAD systems utilize acoustic features such as time, frequency, and amplitude to classify frames of audio, where a frame is defined as a collection of successive samples of audio. Each frame is classified as either speech or noise, where noise is a term for undesired information which, in this case, is the absence of speech. This process is an integral first step in many speech-based applications, allowing for filtering out frames classified as noise, such that the remaining frames, where speech has been detected, can be further processed. This technique is utilized in multiple applications such as speech recognition, speaker diarization, which is a partitioning of audio streams according to the speakers, and hearing enhancement devices like hearing aids and cochlear implants. For example, in hearing aids, VAD is used to detect which frames contain speech, such that speech enhancement algorithms can be applied to said frames, making speech more intelligible to the user. By accurately identifying speech frames, VAD can help improve such systems by reducing the computational load by only passing on the relevant frames for further processing. For example, this could be suppressing frames that are classified as noise or only applying algorithms to frames that are classified as speech.

Designing VAD systems comes with several challenges. One of the most significant ones is handling the varying spectral and temporal characteristics of audio, which are constituted by the acoustic features of speech and noise. This challenge is encountered both during the training and utilization of VAD systems. The spectral and temporal characteristics are influenced by the language, accent, dialect, voice depth, and speech rate of the speaker(s), as well as the environment of the audio recording [1, 7, 21]. For example, there is a difference in the frequency content of speech between men and women, where men usually have deeper voices and women higher. Another example is in languages, where native speakers of some languages are faster at pronouncing words than native speakers of other languages are at pronouncing words of the same amount of syllables [21]. Due to characteristics such as these, it can be difficult to determine specifications such as the amount of samples in a frame, since the most favorable frame size can vary depending on the context.

Another challenge in developing VAD systems is to ensure that they can operate in real-time on small devices, which are devices that have limited resources in terms of computational power and memory, such as hearing aids and smart devices like watches and speakers with digital assistants. This challenge is difficult since the limited resources influence how complex a VAD system can be while still producing accurate results [3, 25]. For example, if it

is desired to have a system with high accuracy, a more complex system is likely needed, which results in producing output at a slower rate since more operations are performed. Conversely, if a fast system is desired, then the accuracy might suffer.

Despite the many challenges associated with developing and utilizing VAD systems in practice, there are various approaches that can be taken to manage these problems. To address the first challenge of the varying spectral and temporal characteristics of speech and noise, a common approach is to utilize digital signal processing (DSP) to create a representation of the data that emphasizes the acoustic features of speech [10, 18, 25]. In Figure 1, a simplified overview of an example of how a VAD system works in practice is illustrated. The first step involves DSP to transform the data, which is represented as a waveform in the figure, into a mel spectrogram, which is a pictorial representation of the frequency content as it varies with time. Then, the next step in the pipeline of the VAD system could be one of various models. For instance, if the data is transformed into mel spectrograms, as seen in the second step in Figure 1, then the mel spectrograms could be provided as input to a convolutional neural network (CNN) since pictorial representations are suitable for CNNs, which then would classify it as either speech or noise. For a CNN model to achieve accurate classification, it must be trained on a data set that must undergo the same DSP as when the model is to be utilized in practice [18, 25]. Furthermore, by selecting a training data set that contains a diverse set of speakers and environmental conditions, the generalization of the VAD system can be improved [1, 32].

Lastly, developing VAD systems to tackle the challenge of real-time operation on small devices involves using model compression techniques on the model after it has been trained. Some examples of model compression techniques are pruning and quantization, which are used to reduce the computational and memory resource requirements [12]. Then, after the model has been compressed, it can be implemented into a VAD system to be utilized in practice.

Currently, there is also a challenge regarding the openness of existing solutions for VAD systems. The challenge lies in the fact that various high-quality VAD systems, such as [28], are available. However, the underlying code, techniques, training data set, and model architecture of these systems are not disclosed. As a result, there is a lack of transparency, which hinders developers and researchers from understanding and making further progress with existing VAD systems.

This paper proposes a VAD model, which addresses the identified challenges in developing VAD systems. The main contributions of this work include:

- An enhanced and compressed CNN model of an existing real-time VAD system, facilitating eventual deployment on small devices.
- An extended and diverse data set for training, in terms of languages, accents, dialects, and amount of speakers, which allows for extending the applicability of VAD systems.
- Releasing an open source and verifiable pipeline on GitHub[1] fully developed in Python for training and compression of VAD systems, enabled for real-time implementation.

[1]https://github.com/lasse2507a/Compressed-Real-Time-Voice-Activity-Detection-with-Open-Source-Implementation-for-Small-Devices

2 RELATED WORK

In the literature, many different techniques have been proposed throughout time for performing real-time VAD. Previously, algorithms based on the energy content of the audio signals were used for VAD. The work of [24] presents four different algorithms for performing real-time VAD that take the approach of utilizing the energy content of the audio. The energy of a frame is a metric for how strong the signal is within that frame. The algorithms are based on a similar naive approach where the energy of each frame is compared with the energy of a typical noise frame. The algorithms try to solve their limitations differently, however, there are still some general problems with VAD systems using this approach. For example, VAD systems using this approach are not well at generalizing across speakers with different voice characteristics and different loudness, which can cause inconsistent performance. Another example of a problem is stationary noise, which can render such VAD systems unable to distinguish between noise and speech segments.

In more recent literature, the most prevalent approaches utilize deep neural networks (DNNs) to perform predictions, which show superior results. For instance, in [30], it is shown that when conducting speech enhancement, a model using a DNN can outperform a traditional approach. Another example of a VAD system, that is not based on a DNN, is the WebRTC VAD [8], which is based on a GMM and considered state-of-the-art when released. In recent literature, there are many examples of DNN-based VAD systems that show far superior results to the WebRTC VAD. For instance, the VAD systems [27] and [23] have been compared to the WebRTC VAD, where the results demonstrate that modern DNN-based VAD systems are superior. The superior performance achieved by DNNs is displayed in [16], where several DNN types are compared on their ability to perform VAD. These types include multilayer perceptrons (MLPs), recurrent neural networks (RNNs), and CNNs, where the best VAD performance among the models was achieved by using a CNN architecture. Similarly, in [25] and [6], the VAD systems utilize a CNN architecture, showing great results in performing VAD. Hence, according to the literature, a CNN is the preferred model in many real-time VAD systems, since it gives the best VAD performance compared to other models.

In general, VAD involves preprocessing steps. For example, in the aforementioned papers [16, 25], and [6] as well as in [27] and [3], the VAD systems include preprocessing. Preprocessing involves representing the data for the system in a way that emphasizes the acoustic features of speech. For example, this could involve constructing spectrograms, which facilitate the identification of patterns in the frequency content, enhancing the ability of the model to classify speech. This approach has been shown to yield far better results. For instance, in [16], the results from only providing time-domain input to the different DNN models did not yield significant enough results to be included in their comparison. Hence, it is beneficial to use preprocessing. In [10], a comparison is conducted of various time-frequency representations of data that is provided as input for CNNs. These representations are obtained using different DSP methods, and the results of the paper show that the MFSC method yielded the best results.

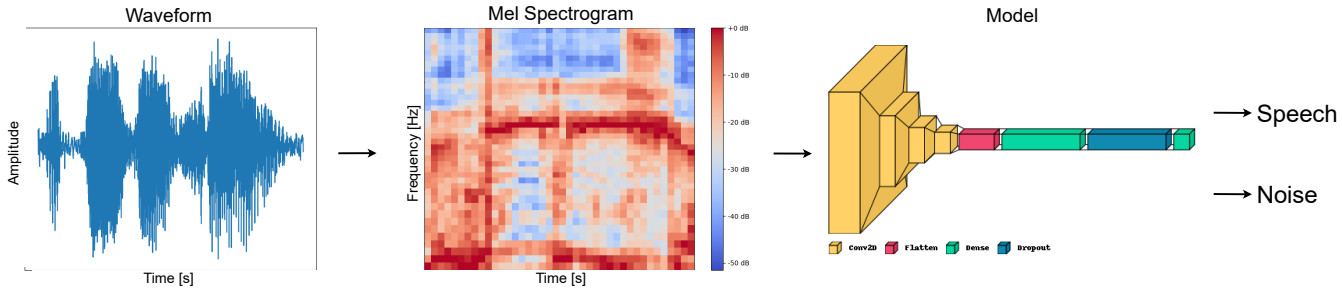

Figure 1: Simplified overview of an example of a VAD system: It takes audio as input, preprocesses the audio to emphasize the acoustic features of speech, and then provides the preprocessed data to a model for classification as either speech or noise.

To achieve VAD systems that can operate in real-time, model compression techniques are often employed. In [17], a comprehensive overview of the latest model compression techniques is provided. The categories of these techniques are network pruning, sparse representation, bits precision, and knowledge distillation. Some prominent methods are pruning and quantization. In [15], an extensive empirical comparison is conducted of some state-of-the-art compression techniques for pretrained CNN models. These include different pruning and quantization techniques. The results of this paper [15] show that it is possible to preserve the accuracy of CNN models while reducing the size of the models to at least 6 times smaller. In [12], they demonstrate the impact that quantization can have on the performance of a VAD system, where they were able to achieve a VAD system that compared to WebRTC had 87x lower delay while still having a better accuracy. Hence, using model compression techniques such as pruning and quantization can reduce the size and inference time significantly without having a significant negative impact on the accuracy of models [12, 15], which makes them beneficial techniques for obtaining models for VAD systems that should operate in real-time on devices with constrained computational and memory requirements.

There are many high-quality, real-time VAD systems available such as [27] and [13], however, the underlying code, techniques, training data set, and model architecture are not disclosed in some of these VAD systems, which is due several of them being commercial. Since the architectures of the models are unknown, it is not possible to make modifications or compress the models. Furthermore, due to VAD systems being non-disclosed, it is difficult to make fair comparisons. For some VAD systems, in which the architecture and underlying code are available, it can still be difficult to extract and use the VAD system. For example, this is the case for MarbleNet [11], which is part of the NeMo toolkit by Nvidia. This toolkit has a lot of functionality that is unnecessary for a VAD system, hence, if it is desired to make modifications or compress the model to implement it on a small device, it would be cumbersome.

3 PRELIMINARY DESCRIPTION

3.1 Sound and Speech

Sound is vibrations that propagate through some medium such as air. These vibrations consist of alternating regions of high and low pressure. Sound and its characteristics can be described as waves. Specifically, a sound wave of a pure tone can be described as

$$x(t) = A\sin(2\pi ft + \phi), \tag{1}$$

where $x(t)$ is the pressure at time t, A is the amplitude, f is the frequency, and ϕ is the phase, which is an offset in time [2].

An illustration of a single period of a sound wave is illustrated in Figure 2. The time it takes for a single wave to pass is called the period T. In the figure, the relation between frequency and period can be seen, which is defined as

$$f = \frac{1}{T}. \tag{2}$$

This relation shows that frequency is the number of periods per second, which has the unit $[Hz] = [s^{-1}]$ [2].

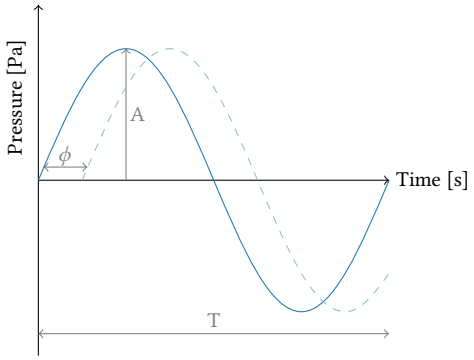

Figure 2: Single period of a sound wave.

In Figure 2, the phase difference of two identical sine waves is illustrated. If the phase is positive, then the wave is considered delayed, and if the phase is negative, then the phase is considered advanced. In Figure 2, the amplitude A expresses how much the pressure varies from normal pressure, which is an advantageous representation since sound consists of pressure variations.

Due to the remarkable sensitivity of human hearing, the variation of pressure that can be detected at the threshold of hearing corresponds to a fraction of atmospheric pressure of less than one billionth. Hence, to account for this phenomenon, the measurement of sound pressure level L_P is expressed in decibels [dB], which is a

logarithmic scale that captures the relationship between different sound pressures. The scale is defined as

$$L_P = 20 \log_{10} \frac{P_{rms}}{P_0}, \quad (3)$$

where P_{rms} is the root mean square of the sound pressure and P_0 is the standard reference sound pressure of 20 µPa that corresponds to the threshold of human hearing [2].

Speech is a complex combination of different sound waves influenced by the movement of the vocal tract. Speech signals are constituted by phonemes, which can be broadly defined as vowel and consonant sounds. Of these, the most prominent sounds come from vowel sounds, which cause peaks in speech known as formants. Hence, the formants constitute the most important information in speech signals, which lies in the range from 300 Hz to 3500 Hz. However, there is still some speech information up to 8000 Hz, meaning that to attain a high-quality speech signal, all the information from 300 Hz to 8000 Hz must be obtained [5].

3.2 Analog to Digital Conversion

Speech signals are analog and need to be converted to a digital representation to be processed. This conversion is known as analog to digital conversion (ADC) [19]. The process is illustrated in Figure 3.

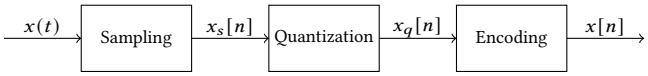

Figure 3: Block diagram showing the ADC process.

As seen in Figure 3, the first step is sampling. This step involves discretizing the signal in time by measuring the amplitude of a signal at discrete intervals. This can be described as

$$x_s[n] = x(nT_s), \quad (4)$$

where $n \in \mathbb{N}_0$ and T_s is the sampling period [19]. From this, the sampling frequency can be determined by

$$f_s = \frac{1}{T_s}, \quad (5)$$

where f_s is samples per second [Hz] [19].

The next step of the ADC is quantization, which discretizes the signal in amplitude. The discretization depends on the bit depth, which is the number of available bits to represent the values of the samples. For example, a bit depth of 4 would result in $2^4 = 16$ possible different values. Hence, quantization discretizes the amplitude of all the samples by rounding to the nearest quantization value. This results in deviations between the continuous signal and the digital signal, which is known as a quantization error. The last step is encoding, which assigns a binary value to all samples based on their quantization values [19].

3.2.1 Aliasing. When sampling signals the choice of sampling frequency is critical to avoid the effects of aliasing, which is an effect that can cause the different frequencies of a signal to be indistinguishable [19].

In Figure 4, an example is given, which illustrates the effect of aliasing. In the figure, the orange curve is sampled with an insufficient sampling frequency, making it indistinguishable from the blue curve.

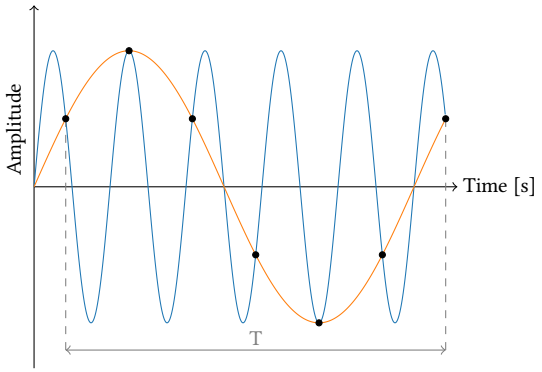

Figure 4: Example showing the effects of aliasing.

Aliasing is essentially high frequencies taking the identity of low frequencies. In a scenario with more complex signals, this could potentially cause a lot of unwanted information to take the identity of the desired information in a recording, which would reduce the quality. In order to avoid aliasing, the Nyquist-Shannon sampling theorem can be taken into consideration. This theorem establishes a sufficient condition for a sampling frequency to allow for capturing all desired information without aliasing occurring in the desired information. Specifically, in order to accurately distinguish between all desired frequencies, they must be below the Nyquist frequency, which is defined as half of the sampling frequency [19].

3.3 MFSC Preprocessing

Preprocessing is the manipulation of data to emphasize features such as speech, which are to be utilized by the following model for classification. A useful method is computing the mel frequency spectrum coefficients (MFSC) to create a mel spectrogram. This method involves several steps, which are shown in Figure 5.

Figure 5: Block diagram showing the MFSC preprocessing steps.

3.3.1 Windowing. The first step is windowing of the signal. Windowing is necessary when creating a mel spectrogram since it splits the signal into temporal segments, which allows for observing the frequency content as it varies with time. The size of the window, which is the amount of samples, affects the trade-off between frequency resolution and temporal resolution. The narrower the window, the better the temporal resolution and the worse the frequency resolution and vice versa [5]. Hence, window size should be chosen based on the concerning application. The windowed segment of a signal is obtained by element-wise multiplication:

$$x_w[n] = x[n]w[n], \quad n = 0, 1, \ldots, N-1, \quad (6)$$

where $x[n]$ is the discrete input signal, $w[n]$ is the discrete samples of the window, and N is the cardinality of the window [5]. Windowing can be done with different types of window functions. In general, the purpose of applying a window function is to mitigate spectral leakage, which can occur in the next step of the MFSC preprocessing. Spectral leakage, which is a distortion in the frequency domain, can occur when using the fast Fourier transform (FFT) since this method assumes that the signal is periodic. The spectral leakage occurs since windowing a signal results in a finite duration window and discontinuities at the endpoints. This assumption of periodicity leads to spectral leakage since, if the signal does not complete an integer number of cycles within the window, the discontinuities at the endpoints of the window introduce additional frequency components that were not present in the original signal [19].

3.3.2 FFT. The next step in the MFSC preprocessing is applying the fast Fourier transform (FFT), which is a more efficient implementation of the discrete Fourier transform (DFT). Given a discrete, finite sequence of samples $x[n]$ of length N, the discrete Fourier transform is given by

$$X[k] = \sum_{n=0}^{N-1} x_w[n] e^{-2\pi jkn/N}, \quad 0 \le k \le N-1. \tag{7}$$

The DFT outputs a complex number that encodes the amplitude and phase of a sine wave with frequency $\frac{k}{N}$ cycles per time unit. As explained earlier, sound waves are represented as sine waves, which is the reason why the DFT is such a convenient method for analyzing the frequency content of signals. The FFT analyzes the frequency content of a signal by decomposing the signal into the constituent frequency components. Through this process, a spectrum is obtained showing the amplitudes of the different frequencies of a signal. Since the FFT is mirrored in frequency, only the first half of the FFT is used. Then, to obtain the power spectrum, the absolute value of $X[k]$ is taken and squared, which results in $|X[k]|^2$.

3.3.3 Mel Filterbank. The last step of the MFSC preprocessing is to apply a mel filterbank to the power spectrum, which results in the MFSC. The mel filterbank is constructed based on the mel scale. The function M for converting from frequency to the mel scale and its inverse $M^{-}1$ is given by

$$M(f) = 2595 \log_{10} \left(1 + \frac{f}{700}\right), \tag{8}$$

$$M^{-1}(m) = 700 \left(10^{\frac{m}{2595}} - 1\right). \tag{9}$$

The mel scale mimics the perception of sound by the human ear, which is better at distinguishing between changes in pitch at low frequencies than at high frequencies [25]. The relationship between the mel scale and frequencies is illustrated in Figure 6, which shows how the higher frequencies are scaled down to lower frequencies to match the perception of sound by the human ear. The perception of the human ear of pitch follows a logarithmic relationship. For example, doubling the frequency of a sound wave does not result in a perceived doubling of pitch.

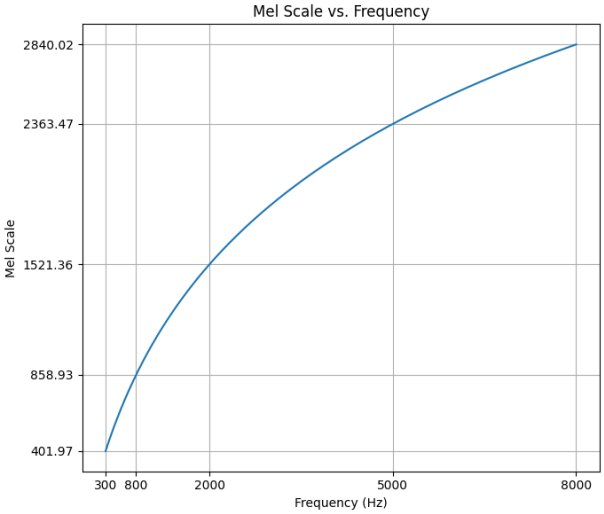

Figure 6: The relationship between the mel scale and frequency for the frequencies constituting the information of human speech spanning from 300 Hz to 8000 Hz.

In a mel filterbank, triangular filters are used. The mel filterbank is constructed with a specific amount of triangular filters that span a desired frequency range. An example of a mel filterbank is shown in Figure 7, which consists of 40 triangular filters with 50% overlap, ranging from 300 Hz to 8000 Hz. These filters are spaced non-linearly in the frequency domain, however, if the frequency domain was scaled according to the mel scale, they would be scaled linearly. Hence, a filter in the lower frequencies is narrower and captures more localized information, and a filter in the higher frequencies is wider and captures information across a broader range. This illustrates the effect that the application of the filterbank has on the frequency content of signals.

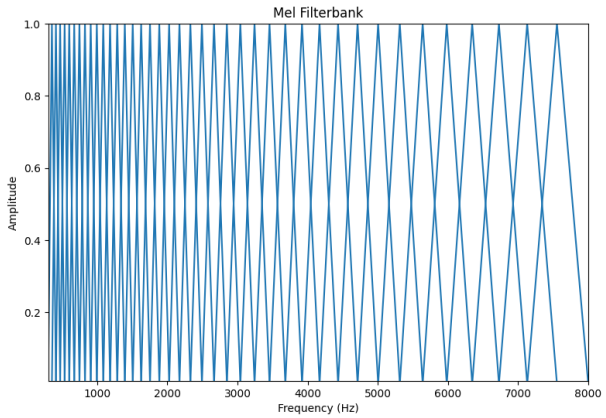

Figure 7: A mel filterbank with 40 triangular filters over frequencies ranging from 300 Hz to 8000 Hz.

When the filterbank has been constructed, it can be applied to the power spectrum obtained from the FFT, which results in the MFSC. The computation of the MFSC for each filter is defined as

$$MFSC(n) = \log\left(\sum_{k=0} H_n[k]|X[k]|^2\right), \quad n = 1, 2, \ldots, N, \qquad (10)$$

where $H[k]$ is the mel filterbank. When N MFSC have been calculated, they can be concatenated to create an $N \times F$ mel spectrogram image, where F is the number of frames. This image can then be used as input for a CNN model.

4 METHODOLOGY

4.1 Design of VAD System

The design of the VAD system is based on the VAD system proposed in [25]. The same preprocessing strategy is used, however, it is implemented differently using other data structures and Python libraries. Additionally, the CNN architecture is used as a starting point whereupon a modification is added.

To design the VAD system, different considerations have been taken into account for the purpose of developing a system that can operate in real-time. These considerations include choosing specifications such as sampling frequency and window size that allow for capturing speech information accurately during recording. Furthermore, to reduce the model size and inference time, a modification to the CNN model and suitable model compression techniques are considered.

In Figure 8, an overview of the VAD system is illustrated. The first step is converting from an analog audio signal $x(t)$ to a digital audio signal, which results in discrete samples $x[n]$. The next step is creating the $N \times F$ mel spectrogram images, which are provided as input to the CNN model. Lastly, the model calculates a prediction and gives a decision on whether the audio contains speech or noise. In order for the VAD system to perform in real-time, multi-threading is utilized by having a thread for each of the blocks in the diagram.

Figure 8: Block diagram showing the VAD system.

4.1.1 ADC. In regards to the first step of the VAD system, which is the ADC, the sampling frequency is determined to be 16000 Hz to achieve a Nyquist frequency of 8000 Hz. This sampling frequency is chosen to allow for sampling with the smallest sampling frequency that contains all information of speech, without aliasing any of the desired information [19]. This results in a high-quality speech signal without unnecessary samples and high-frequency content. Moreover, by encompassing all speech information, the system is expected to exhibit enhanced capability in discerning between noise and speech compared to cases where a lower sampling frequency is employed.

The data type, which determines the bit depth used by the ADC, is determined to be integers of 16-bit. This is a good general-purpose setting for audio analysis [26], providing sufficient quality for the MFSC preprocessing. Furthermore, this data type matches the data type of the training data since it is in the WAV audio format.

4.1.2 MFSC. The next step in the VAD system is the MFSC preprocessing. The design of the preprocessing follows the suggestions of [25]. Hence, the different specifications, such as the window function and size, are chosen in accordance with their findings. The first step of this method is to perform a windowing on the input signal. This windowing is conducted with the Hann window function [19], which is a symmetric window where the values of the samples of the window are lower at the beginning and the end of the frame. This window function has a reduced spectral leakage compared to using no window. The signal is windowed into frames of 400 samples, which takes $400/16000 = 0.025s$ per window. Increasing the window size would lead to an increased amount of data to be processed, which could affect the ability of the system to perform in real-time. Furthermore, it would increase the latency of the system, which would affect how the real-time capabilities of the system are perceived. Conversely, decreasing the window size would lead to a smaller amount of data to be processed, which would decrease the computation time. However, decreasing the window size excessively introduces a challenge in the preprocessing of the training data since the precision of the timestamps of the labels may not be sufficient for very small windows, leading to potential incorrect labels in the training process.

Since the values of the samples at the beginning and end of the Hann window are lower, the frames are overlapped by 50%, which allows for the samples in the beginning and end to be taken into account, allowing for a smooth transition between frames. Each frame is then zero-padded to $2^9 = 512$ samples, which is the nearest power of 2 since a signal of this length speeds up the computation of the next step, which is applying the FFT [29]. Each frame is processed using the FFT as described in Equation 7, and the power spectrum is calculated as $|X[k]|^2$.

For speech processing applications, the amount of triangular filter in a mel filterbank is usually between 24 and 40 [31]. Considering the results in [25], the mel filterbank is constructed with 40 overlapping, triangular filters ranging from 300 Hz to 8000 Hz using the method described in [25]. The frequency range is chosen to match the lower and upper bound of speech information, as well as to match the Nyquist frequency. Then, Equation 10 is applied to calculate the MFSC of each frame until 40 frames are obtained, which are then concatenated to create an $40x40$ mel spectrogram image. This image is then given to the CNN model to make a prediction. For the VAD system to perform in real-time, the mel spectrogram image is updated every five frames, which takes $5 \cdot 200/16000 = 0.0625s$ since the frames overlap by 50%.

4.1.3 CNN Model. For the VAD system to make predictions, a CNN model is used. The architecture of the model is based on the architecture of the model in [25]. The architecture of the CNN model of the VAD system is shown in Figure 1. The MFSC preprocessing produces an image as output, which contains information about both the time and frequency content. Hence, a 2D CNN model is used since this allows for utilizing the information in the image to its full extent.

The difference between the architecture of the model presented in [25] and this model is the addition of a fourth convolutional

layer consisting of five kernels. This modification is chosen to reduce the number of weights and enhance the model capacity by allowing it to learn more complex features. This modification decreases the number of weights since the number of connections from the third layer to the new layer and from the new layer to the fully connected layer is from 250 to 45 to 100 nodes, whereas, the number of connections there would have been from the third layer to the fully connected layer would have been from 250 to 100 nodes. Therefore, it is expected that the computation time and the model size is reduced, which is beneficial for operating in real-time and on a small device.

Table 1: The layers in the architecture of the CNN model.

Layer	Activation Function	Number of Kernels/ Nodes	Kernel Width
Convolutional	ReLU	40	5x5
Convolutional	ReLU	20	5x5
Convolutional	ReLU	10	5x5
Convolutional	ReLU	5	5x5
Fully-connected	ReLU	100	-
Fully-connected	Sigmoid	1	-

CNNs process images as input, which are represented as matrices. In this case, the input is single channel mel spectrogram images, which have the dimensions $40 \times 40 \times 1$. CNNs consist of different types of layers. In the case of the CNN model of the VAD system, these are convolutional, flatten, dropout, and fully connected layers.

Convolutional layers are the building blocks of CNNs. They are capable of extracting local information in the images through weighted kernels that are applied across the entire input images, generating feature maps. This is convenient since this allows for preprocessing the audio signals and representing the results with enhanced speech features as images. The convolution layers are trained to activate these feature maps when detecting patterns of interest [25]. For example, noise can be static and therefore have a repetitive pattern, while the pattern of speech is constituted by formants, which are peaks in the frequency spectrum [5]. As seen in Table 1, the architecture consists of four convolutional layers. The images decrease in size after each layer since strides of 2 are used. This is used instead of pooling layers to reduce the amount of computations.

The next layer after the convolution layers is a flatten layer. The flatten layer converts the two-dimensional output of the convolutional layers into a vector, which is then fed to a fully connected layer. This is used to ensure that the entire output from the last convolutional layer is taken into consideration. Then, a final fully connected layer is used as the output layer, which is used to make predictions to classify whether audio contains speech or noise. This layer uses a sigmoid activation function. This is a non-linear function that is defined as [22]

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}. \tag{11}$$

This function is used in order to get a prediction between 0 and 1 to classify whether the input is speech or noise. In order to determine when to classify the input, a threshold of 0.5 divides the predictions between speech and noise.

In the remaining layers, the activation function that is used is the ReLU activation function. This function is defined as [22]

$$ReLU(x) = \max(0, x), \tag{12}$$

where x denotes the input to the layer. This activation function returns an output of 0 if x Ÿ 0 and x if $x \geq 0$. This function is simple and therefore computationally efficient, which is beneficial for small devices. Furthermore, ReLU facilitates mitigation of the vanishing gradient problem [4].

When training the CNN model, the dropout regularization technique is used between the two fully connected layers. This technique is used to reduce overfitting. During forward passes in the training, the dropout technique randomly ignores some of the output in accordance with a dropout rate, which is between 0 and 1. A dropout rate of 0.75 is used, which means that 75% of the input is accepted and 25% is dropped out. Then during backpropagation, the gradients are only propagated through the non-dropped nodes, meaning that the dropped nodes have no effect on the updates of the gradients. By doing this, the nodes reduce their reliance on specific nodes, since it is not certain that those specific nodes are available every step during an epoch.

4.1.4 Model Compression. To prepare the proposed model for real-time operation, it is compressed using Tensorflow Lite with the default optimization setting. The default optimization setting of Tensorflow Lite is to apply post-training quantization, which involves quantizing the activations and weights of the model by converting them from 32-bit floating-point numbers to 8-bit integers. This reduces the model size and computational requirements, enabling it to run inference more efficiently [14]. The model size of the CNN model was reduced from 430 kB to 40 kB.

5 EXPERIMENTS AND RESULTS

5.1 Setup

The project is fully developed in Python using different libraries, and it is available on GitHub.

To evaluate the real-time capabilities of the VAD system, the VAD system has been implemented on a laptop and a small device. The laptop is a Lenovo 320S, that has Windows 10 installed, an Intel i5-8250U CPU, and 8 GB of DDR4 RAM.

The small device is a Raspberry Pi 3 Model B+ that has the 64-bit version of the Raspberry Pi OS installed. It has a BCM2837B0 CPU and 1GB LPDDR2 SDRAM.

The real-time implementation uses a sampling frequency of 16000 Hz and a window size of 400 samples. Since the model receives input every five frames a prediction should ideally be provided every $5 \cdot 200/16000 = 0.0625s$.

5.2 Offline Evaluation

5.2.1 Data Set and Training. The CNN model of the VAD system has been trained on the AVASpeech data set [9], from which 78 labeled movie clips with a length of 15 minutes and a sampling frequency of 16000 Hz were used. Another clip of 15 minutes was taken aside and not included in the training data, such that it could be used for evaluation. The labeled segments of the movie clips were

divided into shorter clips of 17200 samples. A clip size of 17200 samples results in 10 mel spectrogram images when the MFSC preprocessing is applied since a window size of 400 samples with 50% overlap is used, resulting in a new image for every five new frames after the initial image of 40 frames. From this data, 25000 randomly selected images were set aside and used for validation, and the remaining images were used for training.

The model was trained for 25 epochs with 2044 iterations per epoch with a batch size of 256. The Adam optimization algorithm was used to train the model with a learning rate that was scheduled to gradually decrease. The learning rate started at 10^{-3} for the first 12 epochs, then it decreased to 10^{-4} for 8 epochs, after which it was decreased to 10^{-5} for the remaining 5 epochs. After each epoch, validation is performed. This step encompasses evaluating the performance of the model on unseen data, which is the 25000 images in the validation set. Unseen data is used to provide insights into the overall performance and potential overfitting of the model.

The loss function that was used was the cross-entropy loss function for a binary classification problem, which is defined as

$$loss = -\left(y \log(p)\right) + (1-y)\left(\log(1-p)\right), \tag{13}$$

where y denotes the true prediction, which is 0 for noise and 1 for speech, and p is the output of the CNN model, which is a probability for the occurrence of speech.

5.2.2 Comparison. To evaluate the VAD system, a comparison has been conducted between the proposed VAD system, the proposed VAD system with a compressed model, the VAD system using the original architecture of [25], and the high-quality VAD system called Silero VAD [27].

The CNN model with the original architecture from [25] was created and trained with the same training and validation data sets, as well as the same optimization algorithm, loss function, number of epochs, iterations per epoch, batch size, and learning rate schedule.

To evaluate the VAD systems, the movie clip, which was set aside for evaluation was used. This clip is from a British English movie and was set aside for evaluation since the version of the Silero VAD, that is used, is trained on data with English speakers. The labeled segments of the clip were divided into clips of 17200 samples. To perform predictions with the VAD system and the VAD system using the original model, the MFSC preprocessing was applied to the clips resulting in 7000 images. The Silero VAD uses clips sampled at 16000 Hz as input without any preprocessing applied, therefore, the clips of 17200 samples were provided as input.

To compare the results of the predictions some different metrics are used. These metrics are the precision-recall (PR) curves and the average precision (AP) of these curves, as well as the receiver operating characteristic (ROC) curves and the area under the curve (AUC) of these.

A PR curve shows the relationship between precision and recall across all thresholds. A threshold is a boundary between 0 and 1, which determines the division of the two classes, which are speech and noise. For example, if the threshold is 0.5, then any prediction ≥ 0.5 is classified as speech, and any prediction $\ddot{Y}\ 0.5$ is classified as noise. The precision is defined as

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{14}$$

where TP denotes the number of true positives and FP denotes the number of false positives. Precision can be interpreted as a metric that assesses the quality of results, meaning that a high precision indicates that the proportion of relevant outcomes is greater than irrelevant ones. Then, recall is defined as

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{15}$$

where TP denotes the number of true positives and FN denotes the number of false negatives. Recall can be interpreted as a metric that assesses the quantity, meaning that a high recall implies that a significant portion of relevant results is retrieved, regardless of the inclusion of irrelevant ones.

The AP metric can be used to make comparisons between the PR curves of different models since it summarizes a PR curve as the weighted mean of precisions at each threshold into a single value. The AP is defined as [20]

$$\text{AP} = \sum_{n=0}^{N\tilde{\Theta}1} (R_n - R_{n-1}) P_n, \tag{16}$$

where R_n and P_n are the precision and recall at the n-th threshold, and N is the number of thresholds.

A ROC curve shows the relationship across all thresholds between the true positive rate, which has the same definition as recall, and the false positive rate, which is defined as

$$\text{FPR} = \frac{FP}{FP + TN}, \tag{17}$$

where TN denotes the number of true negatives. The AUC can be calculated to make comparisons between the ROC curves.

In Figure 9, the PR curves and the AP for the different VAD systems are compared, including a compressed version of the proposed VAD system. The PR curves show that the proposed VAD system has an AP of 0.9653, which is slightly closer to the ideal VAD system, which would have an AP of 1, than the VAD system with the original model architecture, which had an AP of 0.9610. This shows that the modifications made in the proposed model improve the model. When compared to the Silero VAD with an AP of 0.6908, the proposed model shows superior results. Furthermore, the compressed version of the proposed VAD system has an AP of 0.9654, which shows that the compressed version yield as good results as the non-compressed proposed VAD system, despite it only being 9.3% of the size.

In Figure 10, the ROC curves and the AUC for the different VAD systems are compared, including the compressed version of the proposed VAD system. This figure shows similar results to the precision-recall figure, where the proposed VAD system has slightly better results than the VAD system with the original CNN model architecture. The AUC of the ROC curve of the proposed VAD system is 0.9745 whilst the AUC of the ROC curve of the VAD system with the original model architecture is 0.9721. The ROC curve of the Silero VAD has an AUC of 0.8608, which again is substantially worse than the proposed VAD system. Again, the compressed version of the proposed VAD system yields just as good results as the non-compressed proposed VAD system.

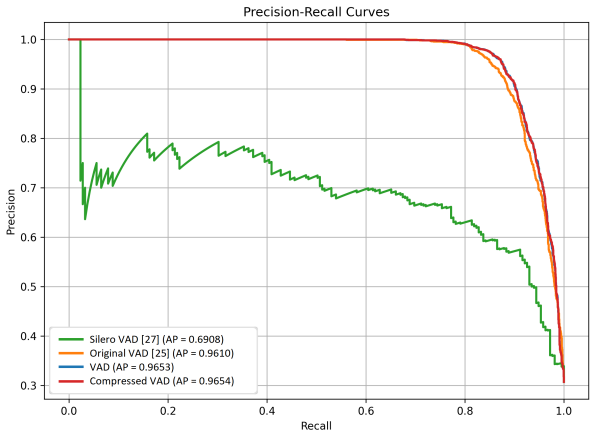

Figure 9: Comparison of the PR curves for the Silero VAD system, the VAD system with the original model architecture, the proposed VAD system, and the proposed VAD system with a compressed model.

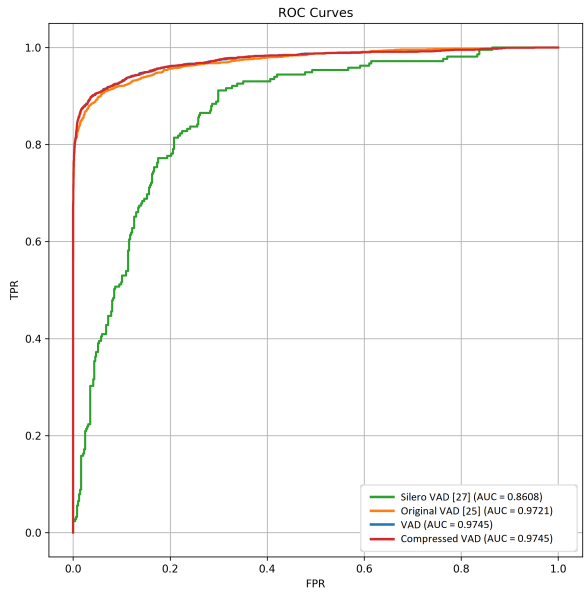

Figure 10: Comparison of the ROC curves for the Silero VAD system, the VAD system with the original model architecture, the proposed VAD system, and the proposed VAD system with a compressed model.

5.3 Real-Time Evaluation

5.3.1 CPU and RAM Usage. When running the real-time implementation on the laptop, the usage was around 0.5% CPU and 365 MB of RAM. On this device, the VAD system was able to keep up with the rate at which the input was received, meaning that output was given every 0.0625*s*.

When running the real-time implementation on the small device, the usage was around 5% CPU and 202 MB of RAM. However, when running the VAD system on the small device, it was not able to keep up with the rate at which the input was received, hence, it had a delay, which kept increasing while the program was active. The program uses multithreading, which should allow for better utilization of the CPU, however, the CPU usage was still very low. It was observed that the RAM usage was increasing while the program was running. The reason for this could be the preprocessing, which involves a lot of transformations of the data such as overlapping windows, windowing, applying the FFT, creating power spectrums, and creating mel spectrogram images. Therefore, the delay was presumably caused by the efficiency of the RAM of the small device.

5.3.2 Classification. To display the results of the classification of the real-time implementation, a window displaying a continuously updating label and plot was created, which can be seen in Figure 11. A new mel spectrogram image is created for every five new audio frames.

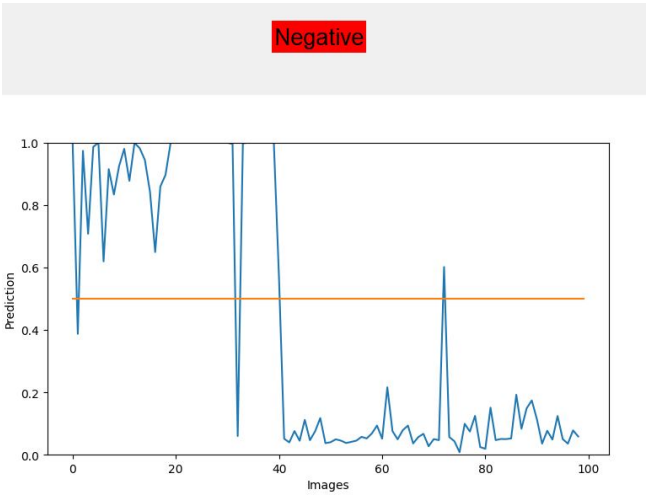

Figure 11: Window with continuously updating label and plot. The label displays positive or negative depending on whether speech is detected, and the plot shows the threshold and the predictions in real-time. The plot shows a prediction for the most recent audio frames represented by mel spectrogram images.

To test the ability of the implementation to classify speech and noise, different simple experiments were conducted. These experiments were not meant to test and measure exactly how well the VAD system is at classifying in real-time, but rather to verify that the VAD system works as intended and to investigate how it works in different scenarios.

The first experiment involved testing the VAD in an environment with minimal noise. The VAD system excelled in this scenario by correctly classifying speech every time there was speech and not making false positives. The next scenario was in noisy environments such as rainfall, white noise, instrumental music, and electronic music. In this scenario, the VAD system again excelled at classifying speech every time there was speech. However, there were some rare false positives, which were caused by some of the sudden loud noises.

To further investigate the generation of false positives, more sudden loud noises were also tested. These noises included clapping and other percussions. These noises were frequently classified as speech, however, observing the predictions when the percussions occurred, the predictions were typically not as high as for the predictions when there was speech, thus, one way to partly fix this problem could be to increase the threshold. In Figure 11, a clap can be seen as the spike around image 70, while there is speech present among the 40 first frames. Looking at the plot, the clap reaches a prediction of around 0.6 while the speech is much closer to 1. Hence, by choosing a higher threshold such as 0.8, then the clap would not be falsely classified as speech in this case.

6 CONCLUSION AND FUTURE WORK

This paper has contributed with a VAD system with a compressed CNN model, which is able to accurately classify in real-time with low latency when speech is present in audio. The experiments show that on general-purpose computers, the system is capable of accurately classifying the presence of speech in audio with low latency. Whereas, when implemented on small devices, the system is showing higher latency, which is presumably an indication of high-load computations in the preprocessing steps. The results of the evaluation indicate that the proposed VAD system is an improvement over the existing solutions, in terms of reducing the model size and improving the level of accuracy among different evaluation metrics. Furthermore, in comparison to existing work, the applicability of the proposed VAD system is extended by training the CNN model on a different and more diverse data set. The entire pipeline, encompassing training and compression of VAD models as well as real-time implementation, is developed in Python and offered as open source. Moreover, all necessary details for replication of the work are disclosed, meaning the work is verifiable and available for future development.

For future work, it is of interest to address the identified challenge with the small devices and more thoroughly test the VAD system. For instance, by evaluating the VAD system with a lower sampling frequency to reduce the input size. Also, evaluating other CNN modules and including spike detection algorithms to improve the performance. Additional devices could be used for evaluation to verify the general applicability of the model.

REFERENCES

[1] Pietro Barbiero, Giovanni Squillero, and Alberto Tonda. 2020. Modeling Generalization in Machine Learning: A Methodological and Computational Study. arXiv:2006.15680 [cs.LG]

[2] Richard E. Berg. 2023. Sound | Properties, Types, & Facts. *Encyclopedia Britannica* (May 2023). https://www.britannica.com/science/sound-physics

[3] George Boateng, Prabhakaran Santhanam, Janina Lüscher, Urte Scholz, and Tobias Kowatsch. 2019. VADLite: an open-source lightweight system for real-time voice activity detection on smartwatches. In *UbiComp/ISWC*, Robert Harle, Katayoun Farrahi, and Nicholas D. Lane (Eds.). ACM, London, United Kingdom, 902–906.

[4] Jason Brownlee. 2020. How to Fix the Vanishing Gradients Problem Using the ReLU - MachineLearningMastery.com. *MachineLearningMastery* (Aug 2020). https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function

[5] Tom Bäckström, Okko Räsänen, Abraham Zewoudie, Pablo Pérez Zarazaga, Liisa Koivusalo, Sneha Das, Esteban Gómez Mellado, Marieum Bouafif Mansali, Daniel Ramos, Sudarsana Kadiri, and Paavo Alku. 2022. *Introduction to Speech Processing* (2 ed.). https://speechprocessingbook.aalto.fi

[6] Muhammad Hilmi Faridh and Ulil Surtia Zulpratita. 2021. HiVAD : A Voice Activity Detection Application Based on Deep Learning. *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika* 9, 4 (Oct. 2021), 856.

[7] Stefaan Van Gerven and Fei Xie. 1997. A comparative study of speech detection methods. In *Fifth European Conference on Speech Communication and Technology*. Citeseer.

[8] Google Git. 2015. webRTC VAD. https://chromium.googlesource.com/external/webrtc/+/branch-heads/43/webrtc/common_audio/vad/.

[9] Yun-Ning Hung, Karn N. Watcharasupat, Chih-Wei Wu, Iroro Orife, Kelian Li, Pavan Seshadri, and Junyoung Lee. 2021. AVASpeech-SMAD: A Strongly Labelled Speech and Music Activity Detection Dataset with Label Co-Occurrence. arXiv:2111.01320

[10] M. Huzaifah. 2017. Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks. arXiv:1706.07156 [cs.CV]

[11] Fei Jia, Somshubra Majumdar, and Boris Ginsburg. 2021. MarbleNet: Deep 1D Time-Channel Separable Convolutional Neural Network for Voice Activity Detection. arXiv:2010.13886 [eess.AS]

[12] Jong Hwan Ko, Josh Fromm, Matthai Philipose, Ivan Tashev, and Shuayb Zarar. 2018. Limiting Numerical Precision of Neural Networks to Achieve Real-Time Voice Activity Detection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2236–2240.

[13] Ian Lavery, Alireza Kenarsari, Reza Rostam, and Dilek Karasoy. 2023. Picovoice. https://picovoice.ai

[14] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. 2021. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* 461 (2021), 370–403.

[15] Giosué Cataldo Marinó, Alessandro Petrini, Dario Malchiodi, and Marco Frasca. 2023. Deep neural networks compression: A comparative survey and choice recommendations. *Neurocomputing* 520 (2023), 152–170.

[16] Serban Mihalache, Ioan-Alexandru Ivanov, and Dragos Burileanu. 2021. Deep Neural Networks for Voice Activity Detection. In *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*. 191–194.

[17] Rahul Mishra, Hari Prabhat Gupta, and Tanima Dutta. 2020. A Survey on Deep Neural Network Compression: Challenges, Overview, and Solutions. arXiv:2010.03954 [cs.LG]

[18] Yasunari Obuchi. 2016. Framewise speech-nonspeech classification by neural networks for voice activity detection with statistical noise suppression. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5715–5719.

[19] Alan V. Oppenheim and Ronald W. Schafer. 2013. *Discrete-Time Signal Processing*. Pearson Education.

[20] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, et al. 2011. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* 12 (2011), 2825–2830.

[21] Erwan Pépiot. 2012. Voice, speech and gender:: Male-female acoustic differences and cross-language variation in English and French speakers. *Corela* (06 2012).

[22] Andrinandrasana David Rasamoelina, Fouzia Adjailia, and Peter Sinčák. 2020. A Review of Activation Function for Artificial Neural Network. In *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. 281–286.

[23] Abhipray Sahoo. 2020. Voice activity detection for low-resource settings. (2020).

[24] A. Sangwan, M.C. Chiranth, H.S. Jamadagni, R. Sah, R. Venkatesha Prasad, and V. Gaurav. 2002. VAD techniques for real-time speech transmission on the Internet. In *5th IEEE International Conference on High Speed Networks and Multimedia Communication (Cat. No.02EX612)*. 46–50.

[25] Abhishek Sehgal and Nasser Kehtarnavaz. 2018. A Convolutional Neural Network Smartphone App for Real-Time Voice Activity Detection. *IEEE Access* 6 (2018), 9017–9026.

[26] Audacity Team. 2023. Audacity Development Manual. https://alphamanual.audacityteam.org/man/Sample_Format_-_Bit_Depth

[27] Silero Team. 2021. Silero VAD: pre-trained enterprise-grade Voice Activity Detector (VAD), Number Detector and Language Classifier. https://github.com/snakers4/silero-vad.

[28] Alexander Veysov and Dimitrii Voronin. 2022. One Voice Detector to Rule Them All. https://thegradient.pub/one-voice-detector-to-rule-them-all/. *The Gradient* (2022).

[29] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, et al. 2019. SciPy 1.0-Fundamental Algorithms for Scientific Computing in Python. *CoRR* abs/1907.10121 (2019). arXiv:1907.10121

[30] Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee. 2015. A Regression Approach to Speech Enhancement Based on Deep Neural Networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23, 1 (2015), 7–19.

[31] Hsiao-Wuen Hon Xuedong Huang, Alex Acero. 2001. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development.* Prentice Hall PTR.

[32] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How Transferable Are Features in Deep Neural Networks?. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (Montreal, Canada) *(NIPS'14)*. MIT Press, Cambridge, MA, USA, 3320–3328.