Voice Activity Detection Based on Deep Neural Networks

Runze Wang

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

December 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Runze Wang**

Entitled: **Voice Activity Detection Based on Deep Neural Networks**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Maria Amer

_____ External to program examiner
Dr. Wen-Fang Xie

_____ Internal examiner
Dr. Maria Amer

_____ Supervisor
Dr. Wei-Ping Zhu

_____ Co-supervisor
Dr. Iman Moazzen

Approved by _____
Yousef R. Shayan, Chair
Department of Electrical and Computer Engineering

_____ 2022

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Voice Activity Detection Based on Deep Neural Networks

Runze Wang

Various ambient noises always corrupt the audio obtained in real-world environments, which partially prevents valuable information in human speech. Many speech processing systems, such as automatic speech recognition, speaker recognition and speech emotion recognition, have been widely used to transcribe and interpret the valuable information of human speech to other formats. However, ambient noise and different non-speech sounds in audio may affect the work of speech processing systems. Voice Activity Detection (VAD) acts as the front-end operation of these systems for filtering out undesired sounds. The general goal of VAD is to determine the presence and absence of human speech in audio signals. An effective VAD method can accurately detect human speech segments under low SNR conditions with any noise. In addition, an efficient VAD method meets the requirements of fewer parameters and computation. Recently, deep learning-based approaches have impressive advancements in detection performance by training neural networks with massive data. However, commonly-used neural networks generally contain millions of parameters and require large amounts of computation, which is not feasible for computationally-constrained devices. Besides, most deep learning-based approaches adopt manual acoustic features to highlight characteristics of human speech. But manual features may not be suitable for VAD in some specific scenarios. For example, some acoustic features are hard to discriminate babble noise from target speech when audio is recorded in a crowd.

In this thesis, we first propose a computation-efficient VAD neural network using multi-channel features. Multi-channel features allow convolutional kernels to capture contextual and dynamic information simultaneously. The positional mask provides the features with positional information using the positional encoding technique, which requires no trainable parameter and costs negligible

computation. The computation-efficient neural network contains convolutional layers, bottleneck layers and a fully-connected layer. In bottleneck layers, channel-attention inverted blocks effectively learn hidden patterns of multi-channel features with acceptable computation cost by adopting depthwise separable convolutions and the channel-attention mechanism. Experiments indicate that the proposed computation-efficient neural network achieves superior performance while requiring a fewer amount of computation compared to baseline methods.

We propose an end-to-end VAD model that can learn acoustic features directly from raw audio data. The end-to-end VAD model consists of three main parts: a feature extractor, dual-attention transformer encoder and classifier. The feature extractor employs a condense block to learn acoustic features from raw data. The dual-attention transformer encoder uses dual-path attention to encode local and global information of learned features while maintaining low complexity by utilizing the linear multi-head attention mechanism. The classifier requires few trainable parameters and few amounts of computation due to the non-MLP design. The proposed end-to-end model impressively outperforms the computation-efficient neural network and other baseline methods by a considerable margin.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Wei-Ping Zhu, for his invaluable guidance, continuous support, and patience during my master study. I genuinely appreciate that he has taught me how to conduct research and justify my findings. His wealth of knowledge and vast experience have always inspired me throughout my academic research and daily life. My gratitude extends to my co-supervisor, Dr. Iman Moazzen from Castofly Company, for his treasured technical support, which influenced my experiments and critiqued my results.

I want to thank Yujia Guo, my girlfriend who works in China, for her understanding and support of my years-long studying in a foreign country. I would also like to thank Xingyu Shen and Jianyu Tang, who shared the joy of life and helped me during my tough time.

Finally and foremost, I have endless gratitude for my beloved parents. They can always cheer me up when I meet difficulties in life or academics. Without their spiritual and material support, I would not be where I am now.

Contents

List of Figures

List of Tables

List of Abbreviations

VAD Voice Activity Detection

SNR Signal-to-Noise Ratio

STFT Short-Time Fourier Transform

ZCR Zero-Crossing Rate

MFCC Mel-Frequency Cepstral Coefficient

GFCC Gammatone Frequency Cepstral Coefficient

MRCG Multi-Resolution Cochleagram

T-F Time-Frequency

PCA Principal Component Analysis

NMF Non-negative Matrix Factorization

GMM Gaussian Mixture Model

SVM Support Vector Machine

HMM Hidden Markov Model

DNN Deep Neural Network

MLP Multilayer Perceptron

FFN Feed-Forward Network

RNN Recurrent Neural Network

LSTM Long-Short Term Memory

GRU Gate Recurrent Unit

CNN Convolutional Neural Network

CRNN Convolutional Recurrent Neural Network

PE Positional Encoding

GAP Global Average Pooling

GMP Global Max Pooling

SVD Singular Value Decomposition

CAIB Channel-Attention Inverted Block

MHA Multi-Head Attention

DATE Dual-Attention Transformer Encoder

ROC Receiver Operating Characteristic

AUC the Area Under the ROC Curve

FLOPs FLoating Point Operations

Chapter 1

Introduction

1.1 General background

Spoken language is one of the most important communication tools for human beings to convey high-level concepts and ideas. Speech is the concrete carrier of spoken language and physically produced by human vocal organs. It involves multiple complex information: the message, the speaker's identity, and physical and emotional state. It has been a scientific research hotspot to develop machines that can automatically process complex information in speech. For example, automatic speech recognition (ASR) systems are invented to translate speech into text; speaker recognition (SR) systems are introduced to recognize people from their speech; speech emotion recognition (SER) systems are developed to analyze human emotions, moods and stress through speech features. In recent years, with the rapid development of computers and artificial intelligence, speech research has been converted to real-life applications such as smart homes, dialogue robots, and smart speakers. They gradually change the way of human-machine interaction from touch to voice, which simplifies people's lives and reflects speech's importance.

In speech processing systems, models are supposed to process audio containing human speech. Otherwise, processing non-speech parts will cause unnecessary computation costs. Besides, non-speech parts can result in models' outputs being inaccurate. For example, models predict human emotion based on speech features in an SER system. The models will take ambient noise as a part of human speech, like TV noise in a living room, potentially causing large biases in emotion

predictions. Another example is that ASR systems are engineered to translate human speech into text. However, unexpected background noises may be translated into random English sentences, resulting notable reduction in recognition accuracy.

To reduce the impact of non-speech parts on speech processing systems, the input audio is supposed to contain human speech only. Voice activity detection (VAD), also known as voice endpoint detection, is a technique to detect human speech segments in audio signals. VAD models usually act as preprocessors to downstream speech processing systems, directly filtering out non-speech segments or providing downstream systems with segmentation information instead.

VAD is a crucial upstream filtering technique for speech processing systems on efficiency improvement. An effective speech processing system usually requires large computation resources that computationally-constrained Internet of Things (IoT) devices can not satisfy. For example, Amazon Echo and Google Home are typical IoT devices in a smart home, which deploy their ASR systems on external high-performance servers. If local-collected speech audio is completely sent to servers from IoT devices, there will be a massive load on servers and networks. Local-deployed VAD allows IoT devices to transmit audio that needs processing to servers exclusively. The followings are concrete examples describing the impact of VAD on other systems.

In an ASR system, an accurate VAD can reduce the amount of computation and processing time by skipping non-speech parts and improve recognition accuracy by effectively avoiding noise inference. Experiments show that if the VAD-judged starting point of speech is delayed by five frames, the speech recognition accuracy will drop by 4.3%; if the starting point is delayed by ten frames, the speech recognition accuracy will drop by 10.8%; If the endpoint is advanced by five frames, the speech recognition accuracy will decrease by 1.5%; if the endpoint is advanced by 10 frames, the speech recognition accuracy will decrease by 6.3% [1]. If the VAD misjudges the speech as noise, the speech recognition accuracy will then be greatly reduced. Suppose the starting point of the VAD judgment advances or the endpoint lags too much. In that case, the computation of the recognition system will be increased, resulting in the prolongation of the system response.

In a voice communication system, the VAD technique is used to improve the utilization rate of transmission channels. Because if we can separate valid speech from background noise and silence, it will be possible to transmit speech data only, which can greatly reduce the amount of

information transmitted. The third-generation mobile communication system standard suggests that voice generally accounts for 40% of the total call duration. Thus, considerable wireless resources can be saved by reducing the number of encoded bits of non-voice segments. Worldwide mobile communication systems adopt the VAD technique to detect non-voice segments and only transmit comfort noise in these segments. This discontinuous transmission mode (DTX) can improve channel capacity. The G.729 speech coding algorithm adopted by the international telecommunication union is to combine a VAD module with a DTX module. This method can theoretically increase the system capacity by 1.5 times and increase the number of cellular network users by 60% [2].

The above examples demonstrate that VAD is a valuable research topic. Audio signals have the short-time steadiness characteristic, and their spectral features remain unchanged in the range of 10 to 30 ms. Thus, audio signals can be divided into overlapped short-time frames, as shown in Fig. 1.1, maintaining the continuity of speech signals and achieving a smooth transition between frames. The duration of each frame is called frame length, and the overlapped length between two frames is called frame shift. Then VAD models classify these audio frames as containing speech or not, which is considered a binary classification problem. The input of VAD models can be raw data of audio frames, but typically some manual feature representations of audio frames. These feature representations highlight discriminatory information of human speech to ambient noise from one or more specific perspectives. The VAD problem can be formulated as

$$\mathcal{X} \to \mathcal{Y} \quad \text{where} \quad \mathcal{X} \in \mathbb{R}^d \quad \text{and} \quad \mathcal{Y} \in \{0, 1\} \tag{1}$$

where d is the dimension of an audio frame or its feature representation. Values in $\mathcal{Y}$ are 0 for non-speech and 1 for speech.

Two major considerations occur when designing VAD models: performance and efficiency. Performance evaluates how well the model predicts speech segments of audio; efficiency reflects how much computation the model requires to make a decision. Since the VAD technique was invented, researchers have proposed many detection approaches based on acoustic features, such as short-term energy, zero-crossing rate, double threshold, cepstral feature, and spectral entropy. Although these VAD methods have achieved considerable high accuracy in a quiet environment,

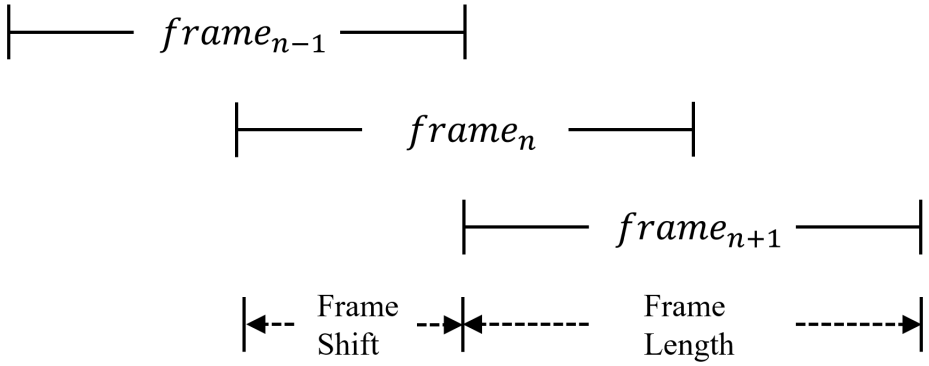

Figure 1.1: Frame length and frame shift.

there is a notable dilution of performance when they are affected by the low signal-to-noise ratio (SNR) real-world noise. This problem hinders the development of noise-robust speech processing systems. More recently, although machine learning and deep learning-based VAD models achieve high detection accuracy in noisy environments, the enormous computation load and model size make them unable to run on computationally-constrained devices. There is still no ideal VAD method which can ensure a high detection accuracy under low SNR conditions while keeping its computation-related requirements as less as possible. Therefore, engineering a VAD method that can achieve a reasonable trade-off between performance and efficiency is a topic worthy of future research.

1.2 Literature review

Since the concept of VAD was first proposed by Bell laboratory in 1959 to improve communication transmission efficiency, it has been developed and applied in many other speech signal processing systems. We can roughly classify VAD methods based on their development process into traditional and machine learning-based methods.

1.2.1 Traditional VAD methods

Traditional VAD methods separate speech from background noise based on quantifiable audio characteristics. Noisy speech audio consists of pure speech and additive noise. From this

perspective, VAD is to determine audio segments with valid speech. These methods assume that we know noisy speech audio and additive noise so that speech can be obtained without difficulty. For example, if we know the energy of speech and noise, we can use energy subtraction to determine non-zero speech segments. Many audio characteristics have been developed to discriminate speech from noise, and most methods make decisions by comparing audio characteristic values with thresholds. And Thresholds are set empirically. Such a mechanism brings the benefit of training and estimating-model free.

In 1975, the Bell laboratory proposed the first VAD method, which chose short-time energy and zero-crossing rate (ZCR) as audio characteristics [3]. The method assumed that the first 100 ms of audio would be totally background noise so that the short-time energy of pure noise could be calculated. The energy was then referenced to set energy thresholds. The authors also employed ZCR to monitor how an audio signal fluctuates across the zero point because unvoiced speech generally has higher ZCR than voiced speech. This characteristic was used to increase the accuracy of VAD by detecting low-energy unvoiced parts. The authors designed a two-level threshold decision-making mechanism for short-time energy and ZCR. Although the method performs satisfactorily in a nearly noise-free environment (SNR at 30 dB or even higher), many assumptions are difficult to achieve in practical applications. The first is that the additive noise is stationary, and the first 100 ms of audio is pure noise; the second is that audio segments with speech must have higher short-time energy than audio segments with only noise. These ideal assumptions make settings of thresholds improper for whole audio-long decision-making.

Based on the analysis of assumptions mentioned above, dynamically estimating the short-time energy of background noise is a robust choice. If we accurately estimate noise, an SNR can be derived by normalizing the audio energy by estimated noise energy. Then we can make decisions based on thresholds that are set empirically. Specifically, speech exists if the calculated SNR is above the threshold. From this perspective, [4–6] have contributed different noise estimation methods. The method in [4] calculated every frame's short-time energy to estimate background noise, which is not feasible for real-time VAD. The authors of [5] propose to use two VADs. The first VAD makes the final decision based on dynamical thresholds, while the second VAD provides the first

one with interval information so that thresholds can be updated based on non-speech parts. The second VAD detects non-speech intervals based on an assumption: segments containing speech have higher short-time energy than segments with only non-speech. The authors of [6] proposed a new spectral-based thresholds updating method, which explicitly reflects changes within each frequency band.

Tokyo University introduced frequency domain features to VAD in 1980 [7]. Unlike time-domain features, frequency-domain features can better distinguish between speech and non-speech segments. The frequency-domain features are extracted from the frequency-domain signal obtained by applying the fast Fourier transform (FFT) to audio signals. Since then, researchers have explored many features extracted from transformed domains of the speech signal, such as pitch variations [8], spectral entropy [9], sub-band energy [10], multi-band modulation energy [11], harmonicity and modulation frequency [12]. In 1993, the cepstral distance was proposed as a novel feature for VAD [13]. It achieved effective detection under negative SNR conditions with stationary Gaussian noise. Long-term signal variability [14] was another feature designed in 2011. VAD adopted this feature can achieve high detection accuracy in low-SNR environments. Low-frequency ultrasound [15] and single-frequency filtering [16] techniques for VAD were developed later. These designed features only reflect partial speech characteristics, making them unable to distinguish speech from background noise in complex environments.

Another branch of traditional VAD methods is the statistical method. They build models from local observations and accumulated historical information distribution. Statistical methods first make model assumptions on the spectral distribution of speech and background noise and then use statistical algorithms to estimate the model parameters dynamically [17]. Typical model assumptions include the Gaussian distribution [18, 19], Laplace distribution [20], Gamma distribution [21], or their combinations [21]. The estimation methods of model parameters include minimum statistic and minimum mean square error estimation (MMSE) [22]. Since statistical methods refer to the long-term speech context information in the parameter updating process, they perform better than threshold-based methods in most cases. However, statistical approaches also have limitations. First, they make assumptions about the distribution of speech and noise, which may not fit the actual data

distribution accurately; secondly, the model can only estimate parameters in limited local observations, and the parameters of the model are relatively less. In addition, most statistical methods perform parameter initialization estimation in the pure noise stage, which may cause a delay in speech estimation and perform poorly in non-stationary noise scenarios.

1.2.2 Machine learning-based methods

As the computing performance of computers has been drastically improved, machine learning-based VAD methods have been vigorously developed. Machine learning methods can be divided into two categories: supervised and unsupervised, where unsupervised machine learning methods explore numerous features and train models from large amounts of data. Dimension reduction and clustering are two common unsupervised learning techniques. Dimension reduction methods extract robust low-dimensional features from high-dimensional observations and then apply them to classifiers, including principal component analysis (PCA) [23], non-negative matrix factorization (NMF) [24] and the spectral decomposition of graph Laplacian [25]. Clustering algorithms are machine learning methods that group data according to observed characteristics, including K-means and Gaussian mixture models (GMM) [26, 27]. Since unsupervised machine learning-based methods do not require massive labelled data, it is easy to obtain training data. However, these methods do not perform well when the SNR is low.

Another research branch of machine learning-based VAD is supervised-learning methods. Supervised learning methods treat VAD as a binary classification problem, and they need to classify audio frames as speech or non-speech. Although supervised methods require a large amount of labelled data and computation resources for model training and testing, existing data resources and the computing power of modern computers are sufficient to support these purposes. Common supervised machine learning-based methods used for VAD include support vector machines (SVM) [28], sparse coding [24], hidden markov model (HMM) [29].

Since the last decades, deep neural network (DNN) has been an indispensable part of machine learning. Their unprecedented effectiveness has also been recognized in many areas, such as computer vision and natural language processing. Many researchers have proposed DNN-based VAD methods that have extraordinary performance on detection accuracy in severe SNR scenarios. These

approaches can learn hidden patterns by implicitly modelling extensive labelled data, which do not require prior knowledge and assumption of the explicit model. This characteristic is attributed to the non-linear transformation ability of DNN methods, which enables them to capture speech variability.

There have been several classic and groundbreaking DNNs that are the foundation of other famous models. The multilayer perceptron (MLP) is the simplest neural network that maps inputs to output non-linearly. As shown in Fig. 1.2, an MLP has input, output and one or more hidden layers with many neurons stacked together. In the perceptron, the neuron must have an activation function that imposes a threshold, such as ReLU and sigmoid. Inputs are combined with initial weights in a weighted sum and subjected to the activation function. The linear combinations feed to the next layer until the output layer, and this process is called forward propagation. Backpropagation is an indispensable learning mechanism that allows the MLP to adjust weights iteratively, with the goal of minimizing the cost function.

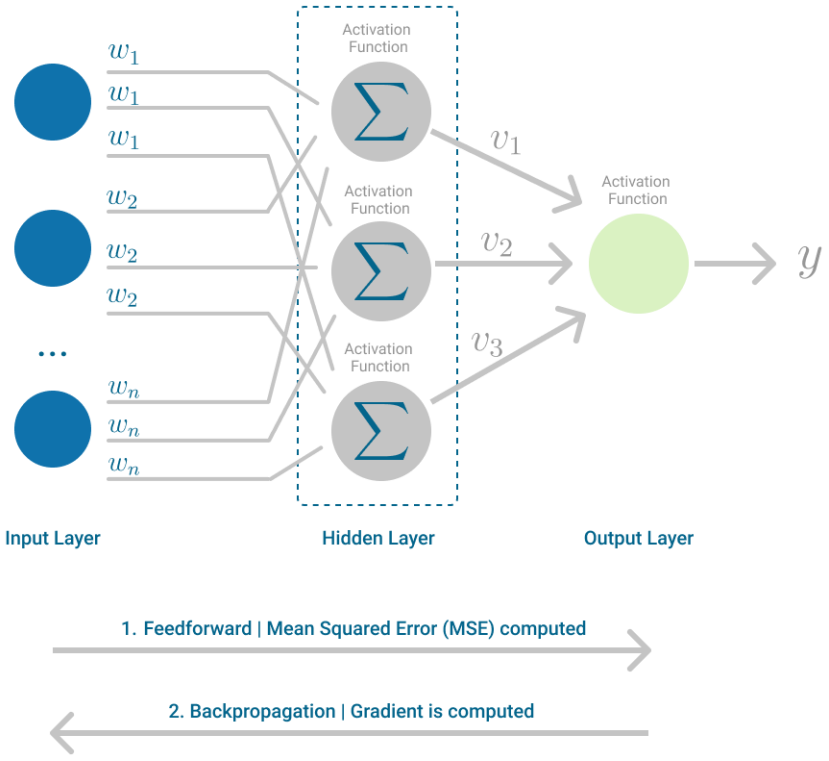

Figure 1.2: Multilayer perceptron [30].

Recurrent neural network (RNN) is a classic model for sequence data learning, its structure is shown in Fig. 1.3. It consists of input neurons x_t, hidden states s_t and output neurons o_t, where t represents timesteps. From the observation, we can find that calculating the output at time step t requires completing calculations of all previous time steps, which is a sequence computing process. Every hidden state is calculated using the input of the current timestep and the hidden state of the previous timestep, as the formula shown

$$s_t = f(u \cdot x_t + w \cdot st - 1 + b) \tag{2}$$

where u, w and b are weights and biases constantly updated during training. The f represents an activation function, such as ReLU or sigmoid. This unique sequential structure allows the hidden state at any time step to always remember previous information, which means the final output integrates all previous information.

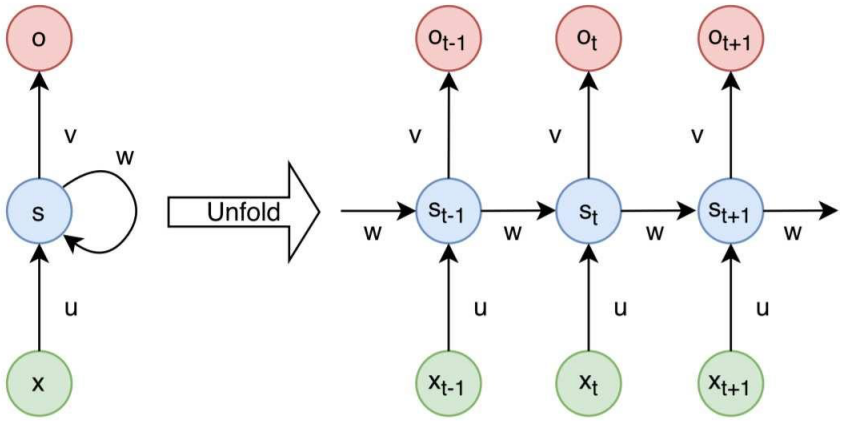

Figure 1.3: Recurrent neural networks.

Deep belief networks were first employed for the VAD task in [31] and were proven to perform better than SVM. The idea of boosting contextual information was applied to an MLP to form boosted deep neural networks in [32]. A new acoustic feature multi-resolution cochleagram (MRCG) proposed in [33] was also first introduced to the VAD task. Compared with simple MLP, this method was reported to have better detection accuracy in complex noise environments. RNNs were adopted for VAD to memorize historical contextual information of speech signals in [34].

They suffered from state saturation and long inference problems when the input utterance was too long [35]. Methods in [36, 37] had a common fault of weak generalization ability when the training and test noises types are not matching. Due to the great success of the attention mechanism in sequence learning [38, 39], it has been attempted to solve the VAD problem in [40–42]. The authors of [40] applied attention mechanism to temporal feature while authors of [41] applied it to spectral-temporal feature. The authors of [42] applied the idea of boosted contextual information in [32] to a transformer encoder, which showed outstanding performance in extensive real-world scenarios testing.

1.3 Contribution of the thesis

Even though existing DNN-based methods have achieved sufficiently high detection accuracy, the model size and computation cost still impede their practical application. DNN models requiring millions of parameters and calculations are difficult to run on computers with limited computation resources.

The generalization capability is indispensable for all supervised VAD methods, so research on improving the detection accuracy in unseen noisy scenarios with different SNRs is critically vital. One of the key points that affect model performance is acoustic features. Existing deep learning-based methods generally extract manual features from audio at the beginning, such as mel-frequency cepstral coefficient, gammatone frequency cepstral coefficient, MRCG or their combination. However, these features are not best suited for the VAD task and all kinds of neural network models.

To solve the first problem, we designe a computation-efficient VAD neural network that consumes less computation and keeps a small model size while achieving superior performance. The model's good performance is partially attributed to the multi-channel acoustic features. The features contain MRCG with its dynamic information Delta and Delta-Delta, and these components are channel-wise arranged with positional information.

We suggest that neural networks can learn how to extract acoustic features for the VAD task rather than using manual features. Therefore, we propose an end-to-end VAD model, in which a

feature extractor network learns to extract features from raw audio data during training and a dual-attention transformer encoder encodes and understands patterns in extracted features. To reduce the computation and model size, we employ the group convolution technique and linear multi-head attention mechanism in the feature extractor and encoder, respectively. Finally, a classifier without fully-connected layers is presented to produce prediction results.

Comparison and analysis of experiments demonstrate that our proposed methods achieve superior performance than baseline methods, including state-of-the-art statistical and deep learning-based methods. In addition, our computation-efficient model's size and amount of computation are significantly less than baseline methods. The performance of our end-to-end VAD model advances all comparison methods by a considerable margin.

1.4 Organization of the thesis

The rest of the thesis is organized as follows:

Chapter 2: This chapter proposes a computation-efficient VAD neural network using multi-channel features. We first introduce background knowledge of our computation-efficient VAD model, including commonly-used acoustic features, standard and depthwise separable convolutions, and channel attention mechanism. Then, we present the flowchart of the proposed model with a brief illustration of modules. We elaborate on the calculation process of proposed multi-channel features step by step. We then describe the structure of the channel-attention inverted block, which enables the CNN framework to be computation-efficient while keeping extraordinary performance.

Chapter 3: This chapter proposes an end-to-end VAD model. Related knowledge of previous works is introduced first, including CondenseNet, Transformer and Linformer. We then provide an overview flowchart of the proposed model. The model comprises a feature extractor, dual-attention transformer encoder and classifier. Rather than calculating manual acoustic features, we design the feature extractor for learning acoustic features from massive data during training. Following, we elaborate on the dual-attention transformer encoder, which employs dual-path linear multi-head attention to process local and global information of extracted features. The classifier with only a convolutional and pooling layer is shortly explained.

Chapter 4: This chapter first introduces the experimental setup, including datasets, baseline methods, parameter settings and evaluation metrics. Experimental results and related discussions are then presented.

Chapter 5: This chapter summarizes the thesis work and suggests directions for future research.

Chapter 2

Computation-efficient voice activity detection based on convolutional neural network

In this chapter, we propose multi-channel acoustic features that incorporate contextual and dynamic information of audio frames. Besides, we design a novel computation-efficient VAD model using convolutional neural networks (CNN) with the channel-attention mechanism. This chapter is organized as follows. In Section 2.1, we first introduce background knowledge related to our computation-efficient VAD model, including acoustic features, convolutional operations and channel attention mechanism. Section 2.2 first provides a flowchart of our proposed model. We then elaborate on the proposed acoustic features and the core of the proposed model, channel-attention inverted blocks.

2.1 Introduction

CNN has been one of the most classic DNN models other than MLP and RNN. CNN was first applied in computer vision tasks and then in speech processing tasks, including speech enhancement, VAD, etc. In speech processing tasks, the input of CNN-based models is generally acoustic

features extracted from raw audio data. Thus, we will introduce several commonly-used acoustic features and their arrangements. To reduce the computational complexity of CNN, we present depthwise separable convolutions and compare them with the standard convolution. Furthermore, we explain the channel attention mechanism that has been proven to enhance the performance of CNN-based models.

2.1.1 Commonly-used acoustic features and arrangements

Most deep learning-based VAD methods need to select acoustic features to highlight speech characteristics. Because acoustic features extracted from a noisy mixture are crucial for correct classification in very low SNR conditions. There are several commonly used and promising acoustic features for speech-related classification problems, such as mel-frequency cepstral coefficient (MFCC) [43, 44], gammatone frequency cepstral coefficient (GFCC) [45] and multi-resolution cochleagram (MRCG) [33]. We will explain and compare these features in the following.

MFCC has been widely used in speech-related tasks, such as automatic speech and speaker recognition, speech separation, and VAD. They are coefficients that collectively make up a mel-frequency cepstrum (MFC), where the MFC represents the short-term power spectrum of a sound. The followings are general steps for calculating MFCC.

(1) Divide the signal into short frames

(2) Take the short-time fourier transform (STFT) of each frame to get the power spectrum

(3) Apply mel filterbank to the power spectra and add up the energy in each filter

(4) Take the logarithm of all filterbank energies

(5) Take the DCT of the log filterbank energies.

(6) Keep DCT coefficients in a specified range and discard the rest.

GFCC is an auditory feature first used in the speech domain. A bank of gammatone filters is used to improve the triangular filters conventionally used in mel-scale filterbanks and MFCC features. To compute GFCCs, we need to pass the signal through a gammatone filterbank to get

sub-band signals. And each sub-band signal is decimated to a specified frequency. Then the cubic root compression is performed on the magnitude of the decimated signals. Finally, take the DCT of the previous output to get GFCCs.

MRCG is a recently proposed acoustic feature compared to MFCC and GFCC. However, it has shown promising performance in many classification-based speech tasks, including speech separation [33] and VAD [17, 42]. The steps for MRCG calculation are described as follows

(1) Compute the cochleagram of the signal with specified frame length and frame shift, then take the logarithm of each time-frequency (T-F) unit to get CG1

(2) Compute CG2 in a similar process but with a larger frame length and the same frame shift

(3) Smooth each unit of CG1 by applying a square window to get CG2

(4) Similarly, smooth units of CG1 but with a larger size of the square window to get CG3

(5) Concatenate CG1, CG2, CG3 and CG4 to form a complete MRCG

CG1, CG2, CG3 and CG4 represent four cochleagrams at different resolutions. The high-resolution cochleagram CG1 captures the local information while three low-resolution cochleagrams, CG2, CG3 and CG4, provide spectro-temporal contexts at different scales.

In [33], a classification-based speech separation experiment was conducted to demonstrate the superior performance of the MRCG feature. In the experiment, the classification accuracy of the same method using different features was compared under -5 dB SNR with different noises. Table 2.1 gives the comparison results of MFCC, GFCC and MRCG.

Table 2.1: Classification accuracy (%) for six noises at -5 dB [33]

	Factory	Babble	Engine	Cockpit	Vehicle	Tank	Average
MFCC	86.5	77.5	90.2	91.1	88.8	88.6	87.1
GFCC	87.7	78.3	91.3	91.9	89.2	89.7	88.0
MRCG	**88.0**	**79.5**	**92.2**	**92.4**	**89.9**	**90.5**	**88.8**

The excellent performance of the MRCG feature on classification problems under low SNR conditions makes it a proper choice for the VAD task, which has also been proven in [17, 42].

Fig. 2.1 describes the calculation process of the MRCG feature. The first 32-D cochleagram feature is produced by inputting the time domain noisy speech into a 32-channel gammatone filterbank with frame length and frame shift set to 20 ms and 10 ms. The last 32-D cochleagram feature is generated using the same scheme with the frame length changed to 200 ms. Two different sizes of square windows are employed to smooth time-frequency units of the first cochleagram feature to get the other two 32-D cochleagram features. We concatenate these four 32-D cochleagram features to get a complete MRCG feature.

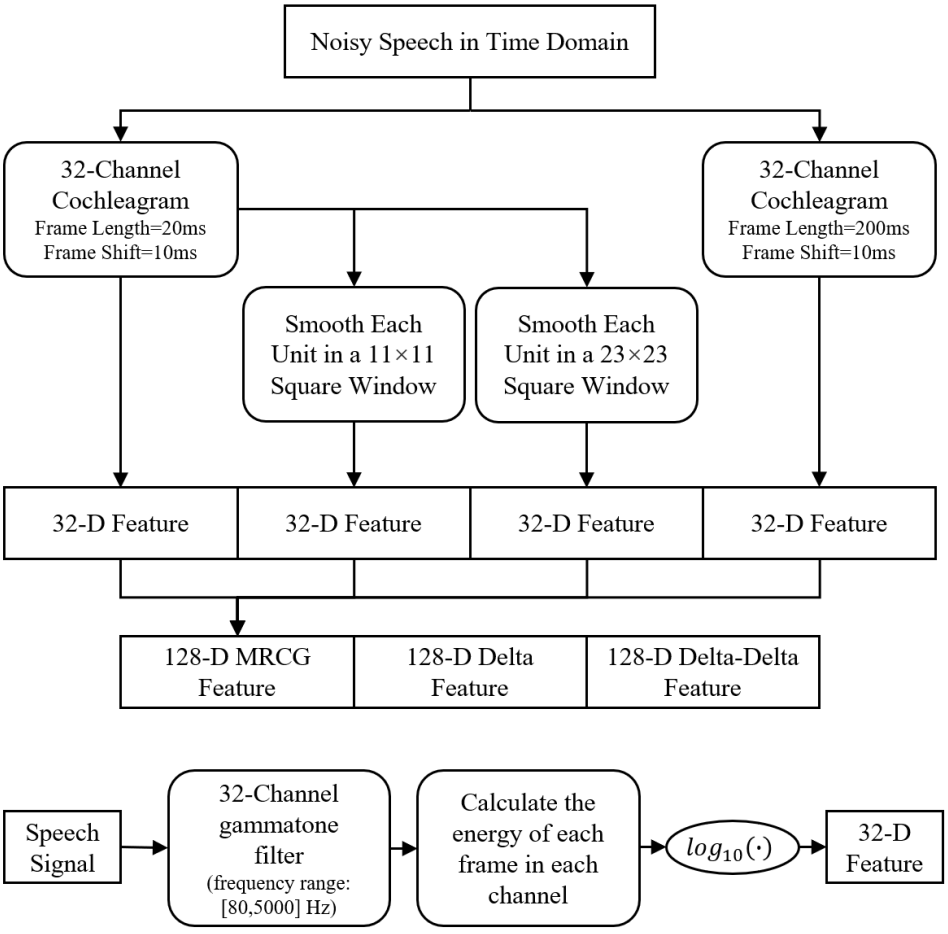

Figure 2.1: Generation of MRCG feature [32].

The MRCG only reflects the static characteristics of speech. However, it turns out that the combination of dynamic and static features can effectively yield better performance [46]. The dynamic

characteristics of speech can be calculated by the first-order and second-order differences of MRCG, delta (Δ) and delta-delta ($\Delta\Delta$). Delta is calculated by

$$\Delta x_n = \frac{(x_{n+1} - x_{n-1}) + 2(x_{n+2} - x_{n-2})}{10} \tag{3}$$

where x_n is the nth unit of MRCG in a given channel. The delta-delta can be derived by applying Eq. 3 to the obtained delta. The MRCG, delta and delta-delta are generally concatenated to form the complete feature of a frame, as shown in Fig. 2.2. The feature vector and ground truth of the frame are written as

$$\begin{aligned} x_n = [&MRCG_1, \ldots, MRCG_d, \Delta_1, \ldots, \Delta_d, \\ &\Delta\Delta_1, \ldots, \Delta\Delta_d] \end{aligned} \tag{4}$$

$$y_n = \begin{cases} 0, & nonspeech \\ 1, & speech \end{cases} \tag{5}$$

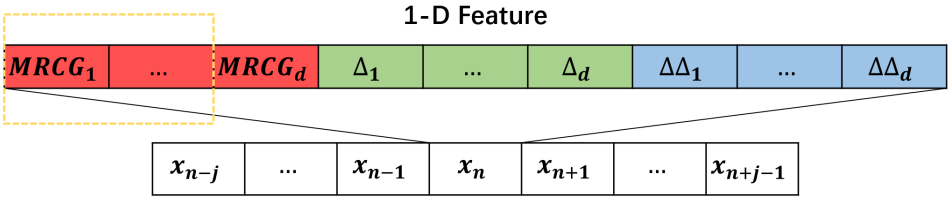

Figure 2.2: Features of one frame [47].

Contextual information effectively improves detection performance whether the classifier is statistical or deep learning-based [17, 40, 48]. This strategy is realized by concatenating a specified number of frames' features before and after the central frame features. For example, we can concatenate j frames before the central frame x_n and $j-1$ frames after x_n. The context-aware feature X_n of the central frame x_n is given as

$$X_n = [x_{n-j}, \ldots, x_{n-1}, x_n, x_{n+1}, \ldots, x_{n+j-1}] \tag{6}$$

the composition of X_n is shown in Fig. 2.2. Due to its limited size, the 1-D convolutional kernel can

only capture features of each frame during the sliding process, which means contextual information is ignored. Thus, researchers usually transform the context-aware feature X_n into a 2-D structure by stacking adjacent frames vertically rather than concatenating horizontally. After that, a single-channel 2-D feature can be obtained and represented in Fig. 2.3.

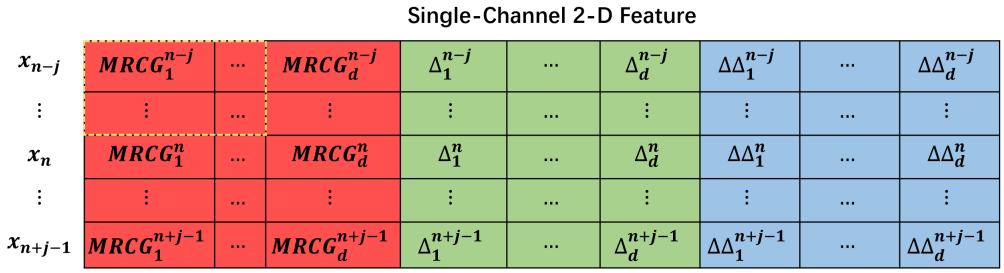

Figure 2.3: Single-channel 2-D features.

2.1.2 Standard convolution and depthwise separable convolutions

Fig. 2.4 depicts a simple standard convolution operation. The cuboid represents an input feature with the shape of (C_{in}, H_{in}, W_{in}), meaning that the feature has C_{in} channels and a (H_{in}, W_{in}) feature map in each channel. The gray $(C_{in}, K[0], K[1])$ cube is a convolutional kernel which usually has the same number of channels as the input feature but a much smaller size $(K[0], K[1])$. The yellow cuboid represents the output of the convolution operation. We start with the kernel sliding over the input feature and performing element-wise multiplication with the overlapped part of the input feature, as shown in the red line. Then we sum up the results into a single output pixel shown as the red square. We repeat this process for every step the kernel slides, finally converting a 3-D feature into a 2-D feature. The output feature is essentially weighted sums of the input feature while values of the convolution kernel are weights. The output feature size is calculated as

$$H_{out} = [\frac{H_{in} + 2 \times padding[0] - (K[0] - 1) - 1}{stride[0]} + 1] \tag{7}$$

$$W_{out} = [\frac{W_{in} + 2 \times padding[1] - (K[1] - 1) - 1}{stride[1]} + 1] \tag{8}$$

where the (H_{out}, W_{out}) is the height and width of the output feature map, $padding[0]$ and $padding[1]$ are the amount of padding applied on both sides of the input feature, and $stride[0]$ and $stride[1]$ mean the amount of steps kernel moves along the row and column directions.

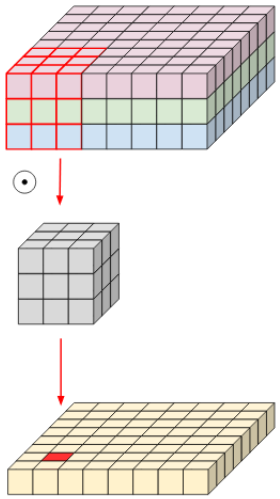

Figure 2.4: Standard convolution [49].

Depthwise separable convolutions were first proposed in [50] to reduce the computation complexity of standard convolution, and they have been adopted by several famous models [51–54]. Depthwise separable convolutions consist of a depthwise convolution and a pointwise convolution, which are both factorized from the standard convolution.

From Fig. 2.5, we can find that the input feature is divided into C_{in} channels and depthwise convolution also has the same number of kernels. We perform convolution on each channel of the input feature with the respective kernel. Then stack convolution outputs together to get output feature maps. In depthwise convolution, one C_{in}-channel kernel can deliver C_{in} output channels; however, it will cost C_{in} kernels with C_{in} channels in standard convolution.

The depthwise separable convolutions are so-called because it deals with not only the spatial dimensions but also the depth dimension — the number of channels. It is inspired by the idea that a filter's depth and spatial dimension can be separated. A concrete example can help us understand the concept behind depthwise separable convolutions. A $(4, 4)$ matrix contains 16 values, and we can represent it with two $(4, 1)$ vectors which only have eight values. The same idea is applied to separate the depth (channel) dimension from the horizontal $(Height, Width)$ dimensions for

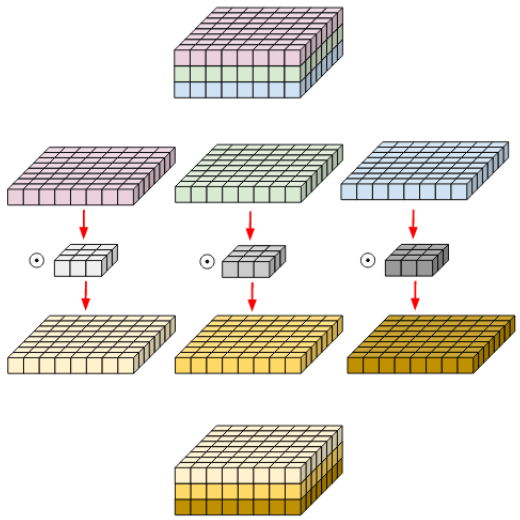

Figure 2.5: Depthwise convolution [49].

standard convolution. Depthwise separable convolutions consist of a depthwise and a pointwise convolution, where the former processes horizontal information and the latter deals with cross-channel information. In Fig. 2.6, the pointwise convolution has a kernel with the shape of $(C_{in}, 1, 1)$, and the kernel slides with stride 1. It will fuse channels' information of every point in the output of depthwise convolution. The output keeps the same horizontal size as the input but only has one channel.

In Fig. 2.4 and Fig. 2.6, though input and output are in the same shape, depthwise separable convolutions retain the computation advantage over standard convolution. We investigate it with the following formulas

$$K[0] \cdot K[1] \cdot C_{in} \cdot C_{out} \cdot H_{out} \cdot W_{out} \tag{9}$$

$$K[0] \cdot K[1] \cdot C_{in} \cdot C_{out} \tag{10}$$

Eq. 9 gives the computational cost of standard convolution. We can find that the computational cost depends on the size of convolutional kernels, the number of input channels, the number of output channels, and the output size. Eq. 10 shows the number of parameters required by standard convolution.

$$K[0] \cdot K[1] \cdot C_{in} \cdot H_{out} \cdot W_{out} + C_{in} \cdot C_{out} \cdot H_{out} \cdot W_{out} \tag{11}$$

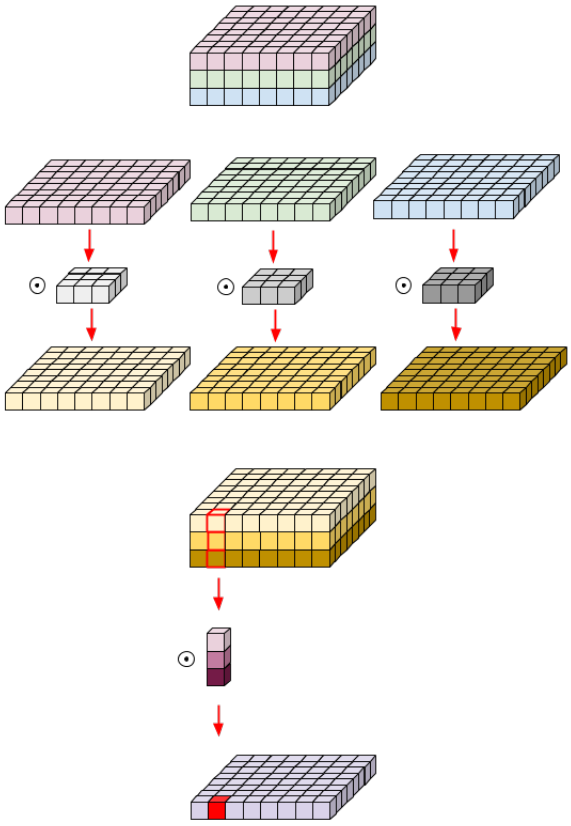

Figure 2.6: Depthwise separable convolutions [49].

$$K[0] \cdot K[1] \cdot C_{in} + 1 \cdot 1 \cdot C_{in} \cdot C_{out} \tag{12}$$

Eq. 11 shows the computational cost of depthwise separable convolution. The former and latter parts of the equation are computations of depthwise convolution and pointwise convolution, respectively. And Eq. 12 illustrates the number of parameters required by depthwise separable convolutions. We can make a more intuitive comparison using the division formula

$$\frac{K[0] \cdot K[1] \cdot C_{in} \cdot H_{out} \cdot W_{out} + C_{in} \cdot C_{out} \cdot H_{out} \cdot W_{out}}{K[0] \cdot K[1] \cdot C_{in} \cdot C_{out} \cdot H_{out} \cdot W_{out}} = \frac{1}{C_{out}} + \frac{1}{K[0] \cdot K[1]} \tag{13}$$

$$\frac{K[0] \cdot K[1] \cdot C_{in} + 1 \cdot 1 \cdot C_{in} \cdot C_{out}}{K[0] \cdot K[1] \cdot C_{in} \cdot C_{out}} = \frac{1}{C_{out}} + \frac{1}{K[0] \cdot K[1]} \tag{14}$$

Eq. 13 and Eq. 14 explicitly demonstrate that the depthwise separable convolution significantly reduced the computation and number of parameters from standard convolution.

2.1.3 Channel attention mechanism

The channel attention mechanism is first proposed in [55]. It is realized by a squeeze-and-excitation (SE) block which can be added to any CNN architecture to improve performance with negligible computational complexity. We will underline the importance of the channel attention mechanism and introduce the structure of the SE block.

In CNN architectures, convolutional layers produce output feature maps with several channels. To keep consistency with previous formulas, we denote the shape of output feature maps as $(C_{out}, H_{out}, W_{out})$, where C_{out} refers to the number of channels and the (H_{out}, W_{out}) refers to the spatial dimensions. Convolutions of input features and kernels produce channels, and each kernel can learn specific features from the input. However, channels might not have the same representative importance, meaning that some channels might be more important than others. Therefore, applying weights based on their importance makes sense before propagating feature maps to subsequent layers. Thus, we need to learn which channels are more important during training and then pay more attention to those channels at the test stage.

Fig. 2.7 depicts the architecture of the SE block, which consists of a squeeze module, excitation module and scale module. First, the squeeze module reduces the spatial dimensions of each feature

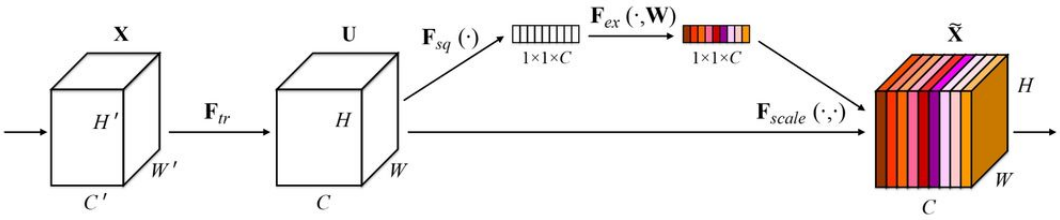

Figure 2.7: Architecture of SE block [55].

map to a singular value since we only concern with the channel information. Global average pooling (GAP) and global max pooling (GMP) are two commonly-used methods to reduce the spatial size of feature maps. The former is selected due to its better performance and smooth characteristic in experiments. Thus, if input feature maps have the shape (C_{in}, H_{in}, W_{in}), the output shape of the GAP operator will be $(C_{in}, 1, 1)$, essentially a vector of length C_{in}.

The next part is to learn the adaptive scaling weights for these channels. The author opts for an MLP to map scaling weights. This MLP has a bottleneck structure with one hidden layer, as shown

in Fig. 2.8. The input is first encoded to a low-dimension representation by a reduction factor r, then decoded back to the original size.

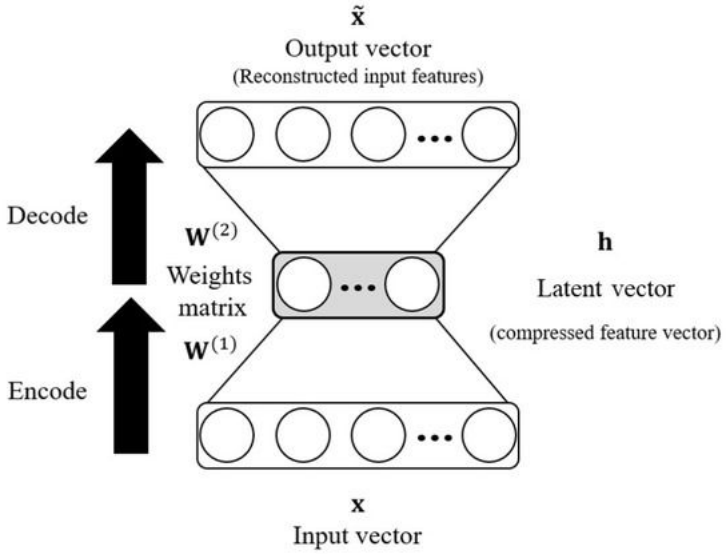

Figure 2.8: Excitation module [55].

In the scale module, a sigmoid activation layer scales the output of the excitation module to a range of zero to one. Then the output is broadcasted and element-wise multiplied with the original input of the SE block.

2.2 Proposed VAD neural network using multi-channel features

The general trend of VAD research has been to develop complicated networks to reach higher detection accuracy. However, the improvement in performance does not make these networks necessarily computational-friendly. Furthermore, sophisticated networks have limitations on both platforms with limited computing resources and tasks with real-time requirements. This section proposes a computation-efficient VAD neural network using multi-channel features to explore a trade-off between performance and computation.

In Section 2.2.1, we overview the proposed method with a flowchart that consists of convolutional layers, bottlenecks, an average pooling layer and a fully connected layer. Section 2.2.2 describes our proposed multi-channel features with positional information and explains ideas of

feature designing. We elaborate on the composition of bottlenecks in Section 2.2.3. Detailed parameter settings for the proposed model are given in Chapter 4.

2.2.1 Proposed method

Block diagram Fig. 2.9 shows the workflow of our proposed method. The well-designed multi-channel features are input to a 2-D convolutional layer. The 2-D convolutional layer extracts local details of features over time and frequency domains, yielding output feature maps with half-size of input features to reduce computation cost. Bottleneck layers learn underlying patterns of speech and non-speech by further processing feature maps. The following module is a 2-D convolutional layer with a $(1, 1)$ kernel, called the pointwise convolutional layer. We employ it to lessen computation by reducing the number of feature maps. The subsequent average pooling layer reduces the size of feature maps before inputting them into the fully-connected layer. At last, the fully-connected layer produces prediction results of frames as 1 representing speech or 0 for non-speech.

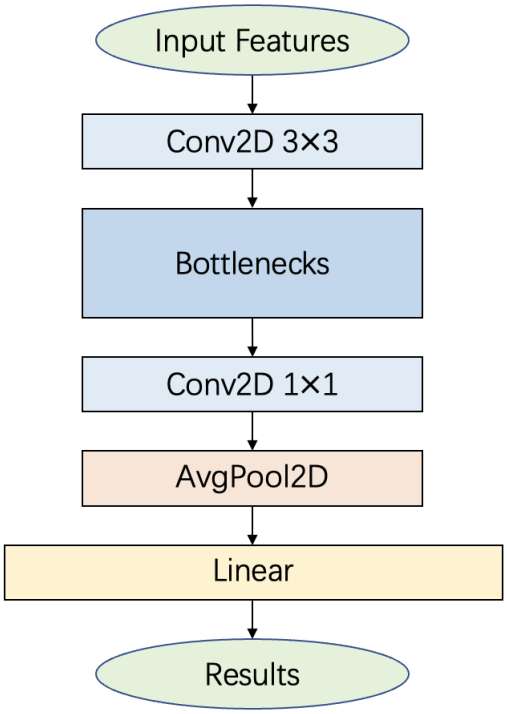

Figure 2.9: Flowchart of the proposed method.

2.2.2 Proposed multi-channel features with positional information

In Section 2.1.1, we have introduced two commonly-used arrangements of features, the 1-D arrangement and single-channel 2-D arrangement in Fig. 2.2 and Fig. 2.3. Although the single-channel 2-D arrangement has enabled convolutional kernels to capture contextual information, kernels can not acquire MRCGs with their dynamic information Δ and Δ simultaneously. To solve this problem, we reorganize the obtained features into a multi-channel arrangement by splitting and stacking MRCGs with their Δ and $\Delta\Delta$, as illustrated in Fig. 2.10.

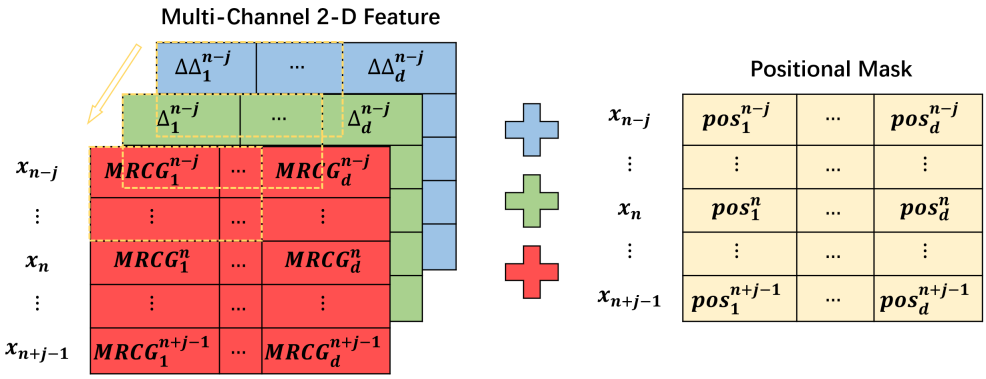

Figure 2.10: Multi-channel 2-D features with positional information.

CNN has kernels sliding over input feature maps, and kernels can collect MRCGs with dynamic information from neighbouring frames. However, kernels cannot capture long-dependency information due to their limited sizes compared to the size of feature maps. Thus, it is necessary to inject sequence information explicitly into the multi-channel features. RNN, LSTM and GRU are usually employed to learn sequence information, but their training is an iterative process. Specifically, the processing of subsequent input elements requires networks to complete the processing of all previous elements. Therefore, the calculation of RNN-based neural networks will be inefficient when the input sequence is lengthy. Here, we utilize the positional encoding (PE) [56] technique to compute a 2-D positional mask which remains the same size as the multi-channel features. The mask will assign a unique encoding value for every position in the multi-channel features. In Fig. 2.10, each channel of the features shares the same positional mask since the delta and delta-delta reflect the trend of MRCG in the same position. This approach takes advantage of no training required, achieving reductions in trainable parameters and the amount of computation. The positional mask

is calculated as

$$PM_{(n,2i)} = sin(n/10000^{2i/d})$$
$$PM_{(n,2i+1)} = cos(n/10000^{2i/d}) \tag{15}$$

where n represents frame index and i is the dimension index.

2.2.3 Proposed channel-attention inverted block

Each bottleneck layer in our proposed model contains m concatenated inverted blocks [57] and channel-attention inverted blocks. The structure of the inverted block is depicted in Fig. 2.11. Each inverted block consists of a pointwise convolutional layer, a depthwise convolutional layer and a linear pointwise convolutional layer. Except for the linear pointwise convolutional layer, every convolutional layer is followed by batch normalization and nonlinear activation function ReLU6. The first pointwise convolutional layer increases the number of channels by t times, and t is the expansion ratio. After depthwise convolution, the last linear pointwise convolution layer reduces the number of channels to c.

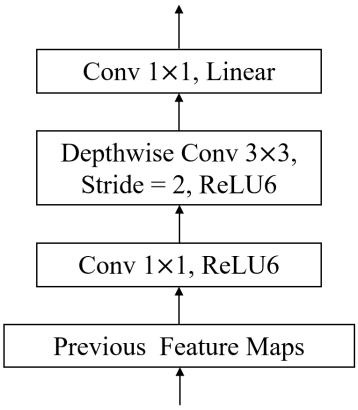

Figure 2.11: Inverted block.

The inverted block is so-called to distinguish itself from the residual block proposed in [58]. Fig.2.12 compares the residual block and inverted block, where a classical residual connects the layers with a high number of channels while the inverted residual connects the bottlenecks. Sufficient experimental results in [57] show that inverted blocks outperform residual blocks.

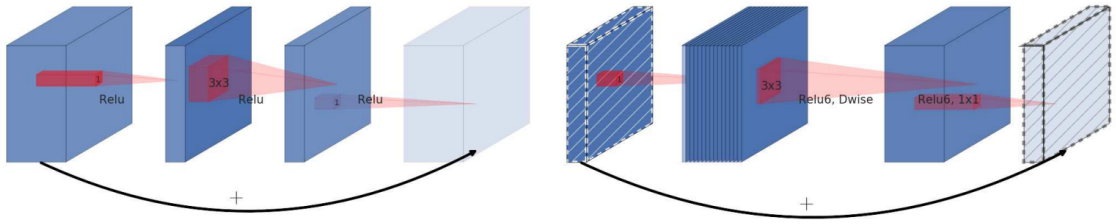

Figure 2.12: Comparison between the residual block and the inverted block [57].

Suppose the parameter s of the bottleneck is 2, the first inverted block has stride 2 in its depth-wise convolutional layer, and the rest $m-1$ inverted blocks have stride 1.The $m-1$ inverted blocks are added with the SE module to achieve channel attention. Here, we denote the inverted block with the SE module as a channel-attention inverted block (CAIB). As shown in Fig. 2.13, CAIB replaces the residual connection in the inverted block with the SE module to add channel attention. The channel attention mechanism adds weights to channels of feature maps to make networks pay more attention to important channels.

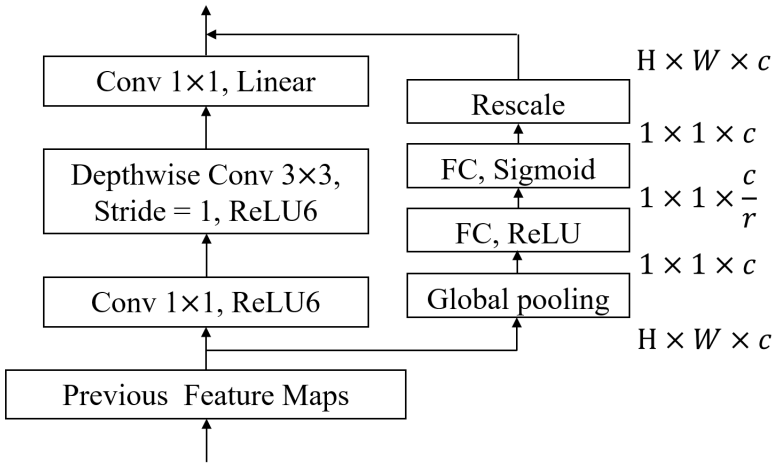

Figure 2.13: Channel-attention inverted block [47].

2.3 Summary

This chapter first reviews the background knowledge of CNN-based VAD methods. Three commonly-used acoustic features for classification-based speech tasks are introduced: MFCC, GFCC and MRCG. Experimental results indicates MRCG outperforms other features under low SNR conditions. We then illustrate the commonly-used arrangements of features with their contextual and dynamic information. Because contextual information and dynamic information are indispensable for improving the performance of models. Depthwise separable convolutions are introduced to replace the standard convolution to reduce a considerable amount of computation in convolutional layers. The channel attention mechanism is explained, which can bring performance improvements to CNN-based models with insignificant computation increments.

We show the overview framework of the computation-efficient VAD neural network and elaborate on our contributions in the following. The first contribution is the proposed multi-channel features which effectively incorporate contextual, dynamic and sequence information for convolutional kernels to learn. Another is to employ depthwise separable convolutions and the channel attention mechanism to build CAIBs to effectively and efficiently learn patterns of speech and non-speech from feature maps.

Chapter 3

End-to-end voice activity detection based on transformer

End-to-end learning is a technique where the model learns all the steps between the initial input phase and the final output result. Specifically, all different parts of neural networks are simultaneously trained instead of sequentially. The characteristic of end-to-end learning requires VAD models to take raw audio data as input and produce desirable classification results in the end. Although most state-of-the-art VAD methods would calculate manual features before inputting them into models, these features are based on human's existing knowledge of acoustic. However, it has been proven that neural networks have the ability to extract features from massive data, and neural networks may acquire knowledge beyond human cognition.

Given the advantages of end-to-end learning, we propose an end-to-end VAD model, which consists of a feature extractor, a dual-attention transformer encoder and a classifier. Section 3.1 introduces background knowledge related to our end-to-end model, including CondenseNet, Transformer and Linformer. Section 3.2 first provides an overview of the proposed model, then elaborates on the three modules mentioned above.

3.1 Introduction

The end-to-end VAD model is supposed to extract features from raw audio data, encode extracted features (learn and understand features), and classify frames as speech or non-speech. Our proposed feature extractor's core is the condense block introduced in Section 3.1.1. Section 3.1.2 introduces the transformer neural network, including explanations of the encoder-decoder structure and multi-head attention mechanism. To improve the efficiency of transformer neural networks, Section 3.1.3 presents Linformer that achieves the attention mechanism with linear complexity.

3.1.1 CondenseNet

CondenseNet [59] is a computation-efficient CNN architecture and it is also the improved version of DenseNet [60]. Therefore, we will first introduce the DenseNet. Dense blocks are cores of the DenseNet, and they have a massive contribution to the strong performance of DenseNet. Dense blocks employ novel dense connectivity rather than the resnet connectivity in [58]. Fig. 3.1 shows that resnet connectivity promotes gradient propagation by identity mapping.

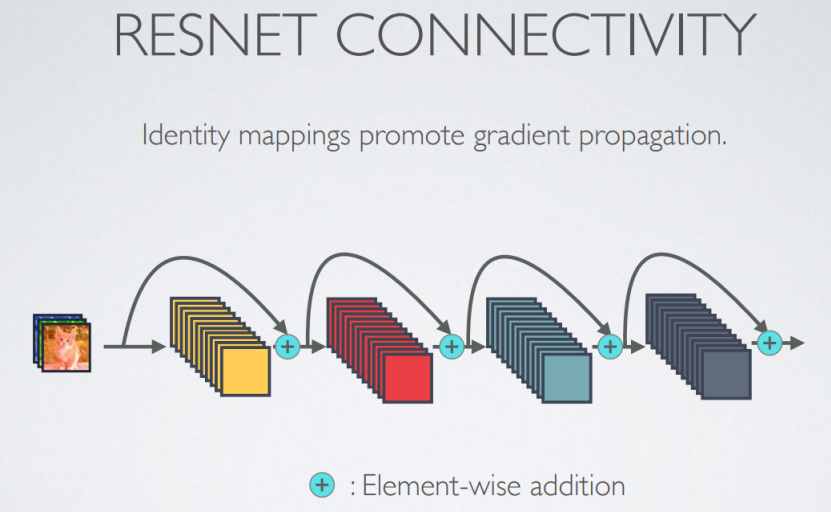

Figure 3.1: Resnet connectivity [60].

Fig. 3.2 presents the dense connectivity that allows each layer to obtain inputs from all preceding layers and pass on its feature maps to all subsequent layers. The dense connectivity is achieved by channel-wise concatenation, which means that each layer can receive "collective knowledge"

from all previous layers. It is also worth noting that only k new channels are produced by each convolution operation, where k represents the growth rate. It differs from the standard convolution, where the number of channels increases exponentially.

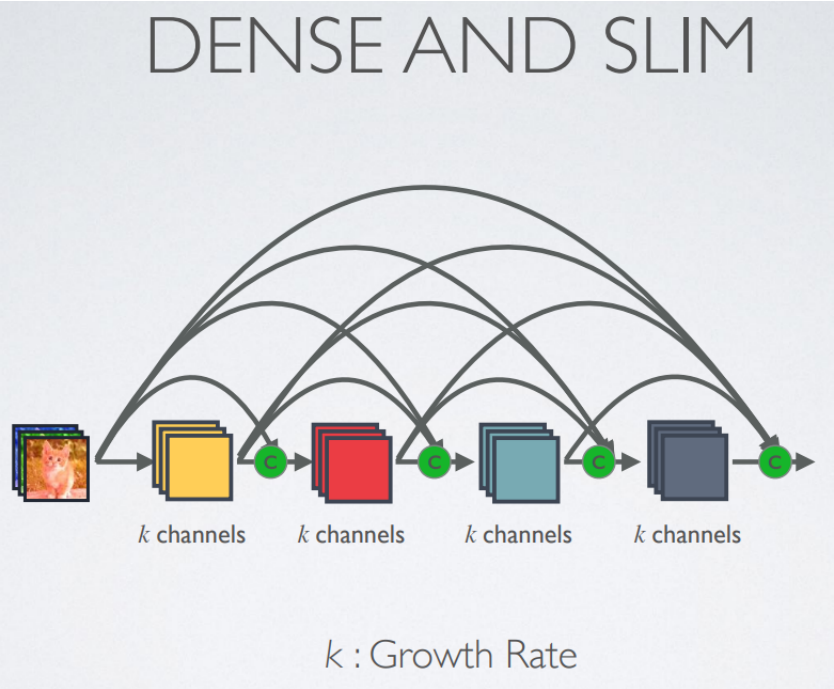

Figure 3.2: Dense connectivity [60].

Fig. 3.3 depicts the forward propagation process of a dense block, where $x_0, x_1, ..., x_4$ represent feature maps in each layer and $h_1, ..., h_4$ mean a composite operation of batch normalization, ReLU activation and convolution. When layers in one dense block become deeper, the computational cost of convolution grows drastically. To solve the problem, the author employs pointwise convolution to reduce the channel size before the (3×3) convolution, as shown in Fig. 3.4.

Although DenseNet has introduced dense connectivity that achieves great performance improvement, the dense connectivity will lead to network inefficiency due to its redundant structure. CondenseNet addresses the problem of DenseNet occupying extensive memory footage and requiring large amounts of computation. CondenseNet retains the same performance as DenseNet but only with one-tenth computation cost of DenseNet. It can also save half the computation cost when keeping the same performance as MobileNet and ShuffleNet [61]. This advancement can be attributed to the following techniques proposed in CondenseNet, and we will introduce them later.

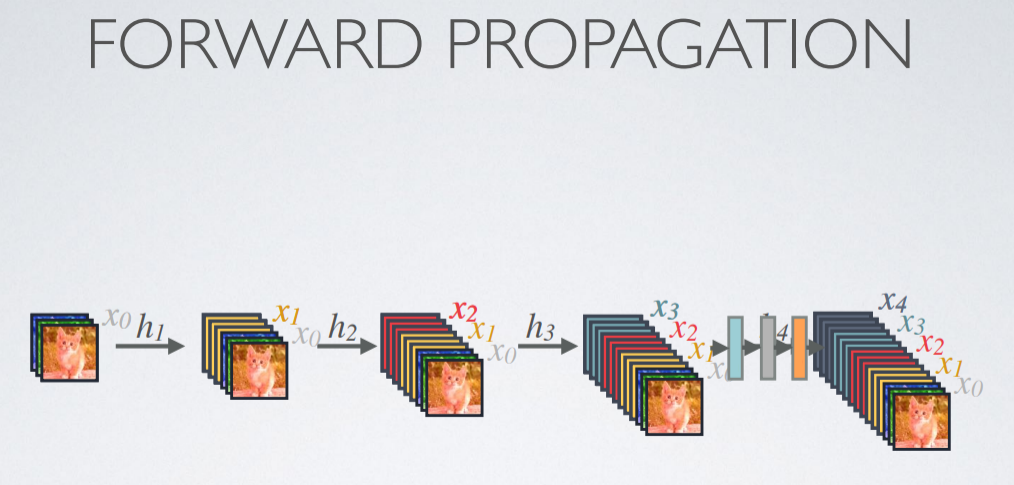

Figure 3.3: The forward propagation of a dense block [60].

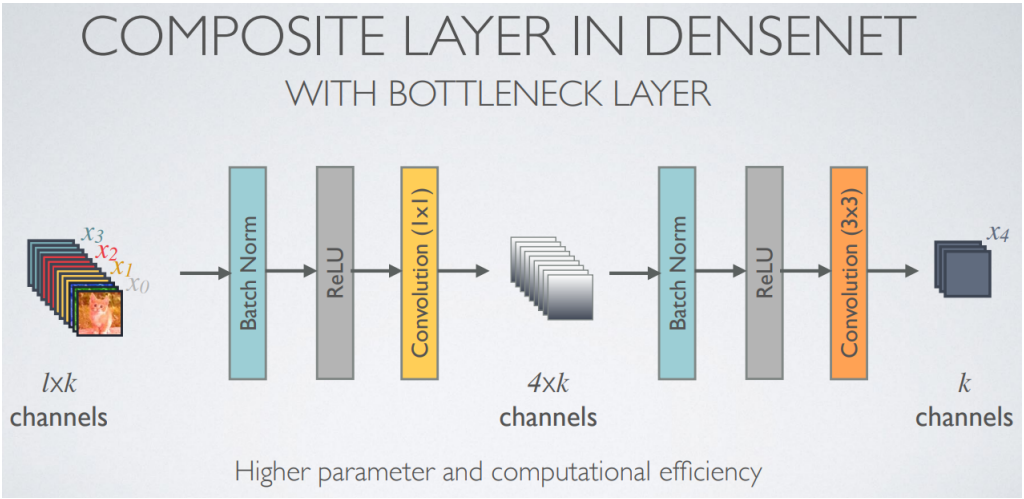

Figure 3.4: Bottleneck layer in a dense block [60].

(1) Learned group convolution

(2) Index layer

(3) Exponential-increasing growth rate and other structure improvements

Fig. 3.5 shows structures of a dense layer, condense layer at training time, condense layer at test time. We can find training-phase condense layer replaces 1×1 Conv and 3×3 Conv with 1×1 L-Conv (Learned Group Convolution) and 3×3 G-Conv (Group Convolution). For the condense layer at test time, the index layer selects feature maps based on the learned group convolution result

at training time. Besides, the 1×1 L-Conv is changed to a normal 1×1 G-Conv.

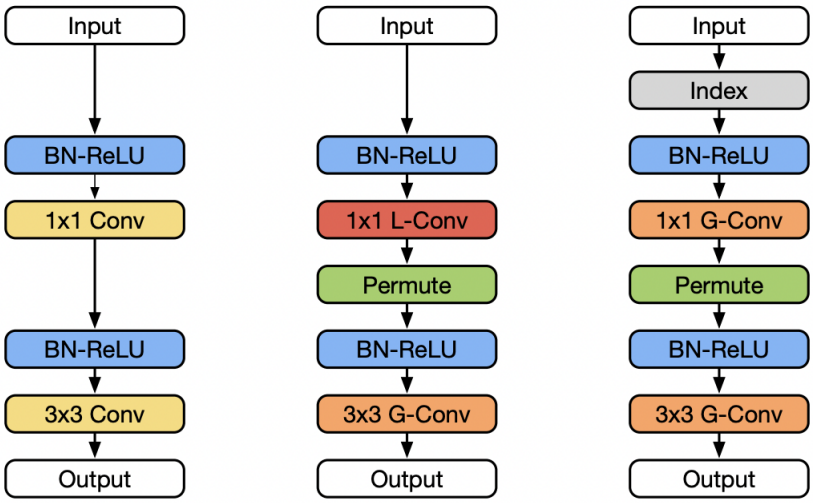

Figure 3.5: Dense layer(left), condense layer during training (middle) and condense layer during test(right) [59]

The group convolution is first applied to a dual-GPU architecture in AlexNet [62], and it is first proposed as a performance-improving strategy in ResNeXt [63]. Fig. 3.6 describes a standard convolution and a group convolution. We observe that group convolution is a sparse representation of standard convolution, and groups are preset. Even if the sparse connection is efficient, manually-specified groups can not learn which connections are more important and which can be ignored. Furthermore, the ShuffleNet in [61] supposes group convolution has problems of weak connections between channels and insufficient feature diversity. CondenseNet addresses this issue by letting networks learn sparse representations of convolutional networks during training and decide weights to keep, which is called learned group convolution.

The learning process contains two stages: condensing and optimizing. Fig. 3.7 illustrates the process of a learned group convolution with three groups and condensation factor 3. If we assume the condensation factor is C, there will be $C - 1$ condensing stages in training. Therefore, there are two condensing stages and one optimization stage in Fig. 3.7. The first condensing stage is a standard convolution but added with sparse regularization by utilizing group lasso, which makes learned parameters present in sparse distribution so that subsequent weights pruning will not result in much accuracy loss. Because cutting out a connection with a zero weight does not affect accuracy.

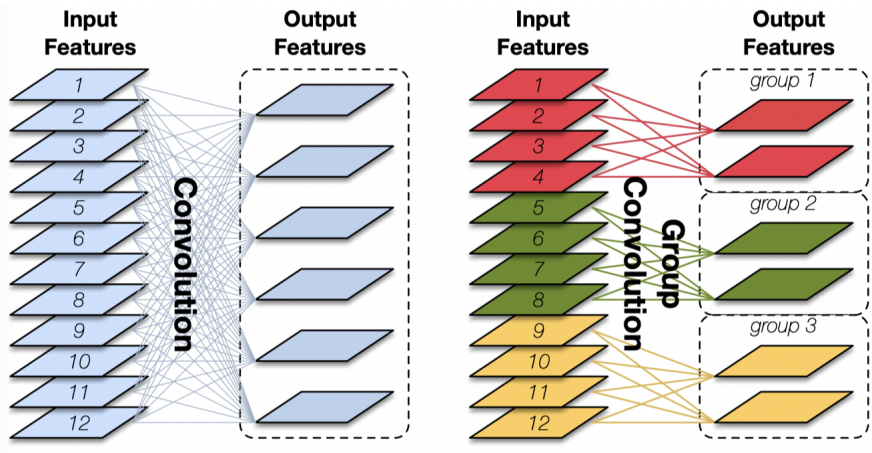

Figure 3.6: Standard convolution and group convolution [59].

Then weights pruning is performed channel-wise in the following condensing stage. The same number of connections are pruned from each group. Here, every condensing stage except the first one will prune $\frac{1}{C}$ connections; there will be another $\frac{1}{C}$ connections pruned at the optimization stage. Thus, only $\frac{1}{C}$ connections remain in the end.

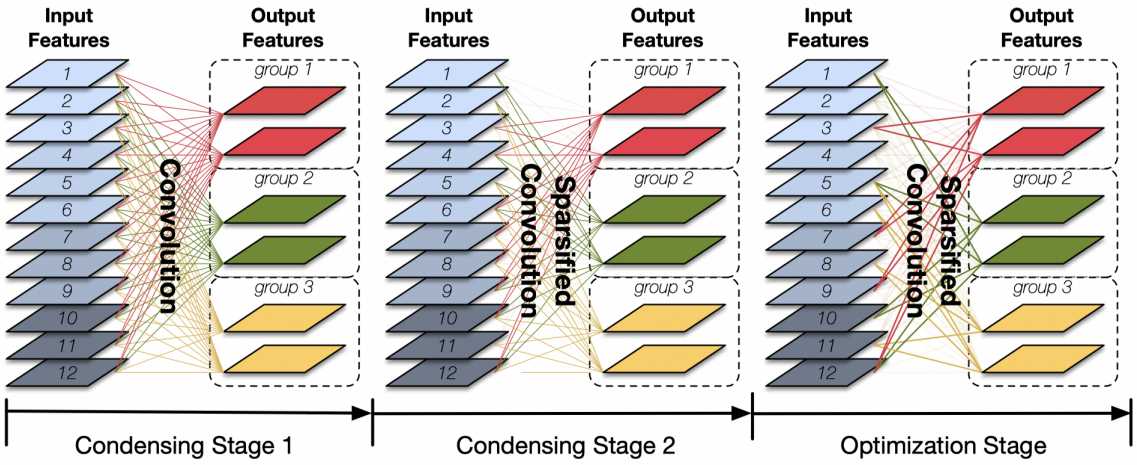

Figure 3.7: Illustration of learned group convolutions [59].

The next novel technique introduced in CondenseNet is the index layer used at the test stage. We have learned to preserve important connections using learned group convolution during training. The index layer rearranges input features based on the previously learned results. For example, the learned group convolution has decided that feature maps used by the group 1, 2 and 3 are

$(3, 7, 9, 12)$, $(1, 5, 10, 12)$ and $(5, 6, 8, 11)$, as shown in Fig. 3.7. In Fig. 3.8, the index layer then arranges input features into $(3, 7, 9, 12, 1, 5, 10, 12, 5, 6, 8, 11)$ so that features can be directly applied to a standard group convolution.

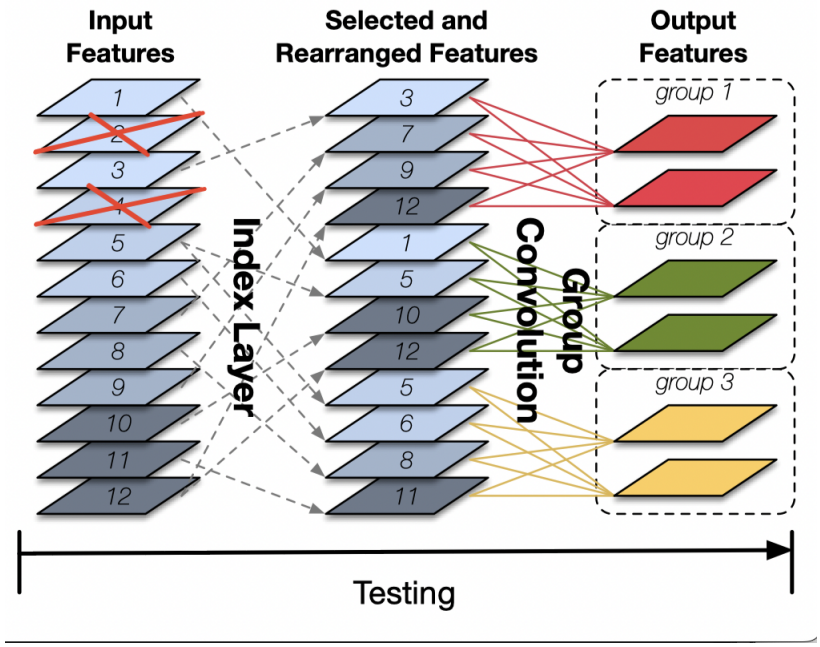

Figure 3.8: Index layer [59].

Other than previous techniques, CondenseNet also proposes two improvements to DenseNet. The first is adjusting the constant growth rate k in DenseNet to an exponential growth rate. In a dense block, the number of input channels increases k layer by layer, which means each layer receives k channels' information from each previous layer. However, deeper convolutional layers depend little on the information from the first several layers of the same dense block. CondenseNet adopts an exponential-increasing growth rate for different blocks while the growth rate remains the same in each block. This helps the network strengthen the connection of adjacent blocks and increase computation efficiency. The second improvement is CondenseNet introduces the dense connectivity to the blocks level, as shown in Fig. 3.9. Pooling layers are used to downsample feature maps to the same size, which realizes the concatenation of feature maps with different sizes.

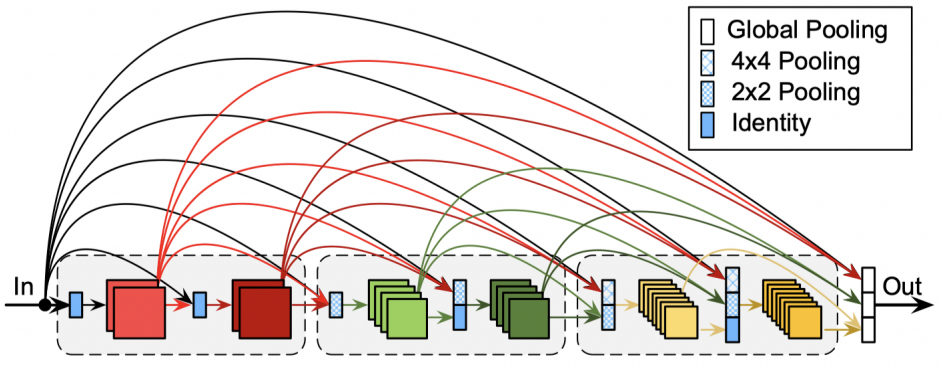

Figure 3.9: Condense connectivity [59].

3.1.2 Transformer

The encoder-decoder framework has been widely used in sequence-based tasks, such as machine translation and text summarization. The encoder-decoder is a process that simulates human cognition. In psychology, cognition refers to the process of acquiring knowledge through mental activities such as the formation of concepts, perceptions, judgments or imaginations, that is, the psychological function of individual thinking for information processing. The encoder memorizes, understands and refines information to form a low-rank vector. In this process, the encoder can only rely on the input information, or we can add a prior knowledge or attention mechanism when building the whole model. Similar to the encoder, humans usually use common sense and prior knowledge to understand and abstract new information they are exposed to. The encoding format corresponds to the memory in the human brain. The decoder mixes the encoded low-rank vector with other information and decodes them into a required form. The encoder-decoder training process is similar to the human brain's learning process of information cognition and application. For example, in an encoder-decoder-based machine translation system, the machine first understands a sentence in one language and then translates the understood intrinsic information into another language.

Before the invention of the transformer, a popular choice for sequence-based tasks was LSTM-based models. Because LSTM-based models can give meaning to the sequence while remembering or forgetting the parts it finds important or unimportant. However, RNN and LSTM-based models are always limited by their characteristics of sequential computation. In recent years, transformer

and its variants have been proven effective for many sequence-based tasks, such as natural language processing [56] and speech signal processing [42, 64].

Fig. 3.10 depicts the encoder-decoder architecture of the transformer. Since the transformer is initially designed for the machine translation task, the input embedding and positional encoding techniques are introduced to preprocess text inputs. The former provides embedding vectors to represent words, and the latter assigns positional information to words.

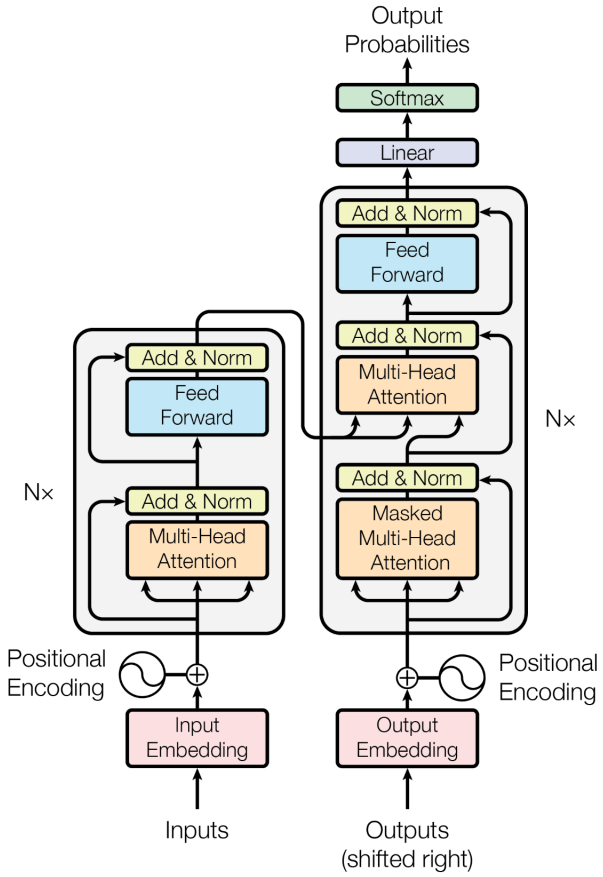

Figure 3.10: Architecture of transformer [56].

Transformer encoder and decoder

The transformer encoder contains N identical layers and each layer consists of two sub-layers, multi-head attention (MHA) and feed-forward network (FFN). Each sub-layer is added with a residual connection and layer normalization.

$$sublayer\_output = LayerNorm(x + (SubLayer(x))) \tag{16}$$

Similarly, the transformer decoder consists of N identical layers but each involves one more sub-layer, the masked MHA. By blocking future information, the masked MHA makes the output of the current time step only be determined by itself and past information, which keeps the auto-regressive characteristic of the transformer. Following MHA and FFN decode the encoded high-level information by processing outputs from the encoder and masked MHA together. Each sub-layer of the decoder is also accompanied by a residual connection and layer normalization.

Multi-head attention

The attention mechanism looks at an input sequence and decides which other parts of the sequence are important at each step. The intuition of the attention mechanism is that we always focus on the word we read but at the same time, our mind still holds important keywords of the whole text in memory to provide context. In Fig. 3.11, the attention mechanism is shown on the left and it can be described by the following equation

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{17}$$

$$Q = W^q I \quad K = W^k I \quad V = W^v I \tag{18}$$

where Q, K, V are query, key and value matrices. They can be calculated as Eq. 18, where the I is the inputs matrix, W^q, W^k and W^v represent transformation matrices that will be learned during training. The dimension of vectors in K is denoted as d_k.

The MHA mechanism is depicted as the right part of Fig. 3.11. The MHA adopts h times of different and learnable linear transformations, which produces h pairs of projected Q, K, V. The h times scale dot-product attentions are performed in parallel, where each is called one-head attention. The h-head attention is then concatenated and input into a linear layer to produce the MHA results. The MHA provides multiple representation subspaces, which allows the model to focus on information from different representation subspaces in different positions. That is, the MHA enables models to capture richer information about features. The procedure of MHA can be

formulated as follows

$$\begin{aligned} MultiHead(Q,K,V) &= Concat(head_1,...,head_h)W^O \\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{19}$$

where h is the number of heads. W_i^Q, W_i^K, W_i^V denote ith linear transformation matrix for Q, K, V, respectively, and the i ranges from 1 to h. The W^O represents the linear transformation matrix of the linear layer.

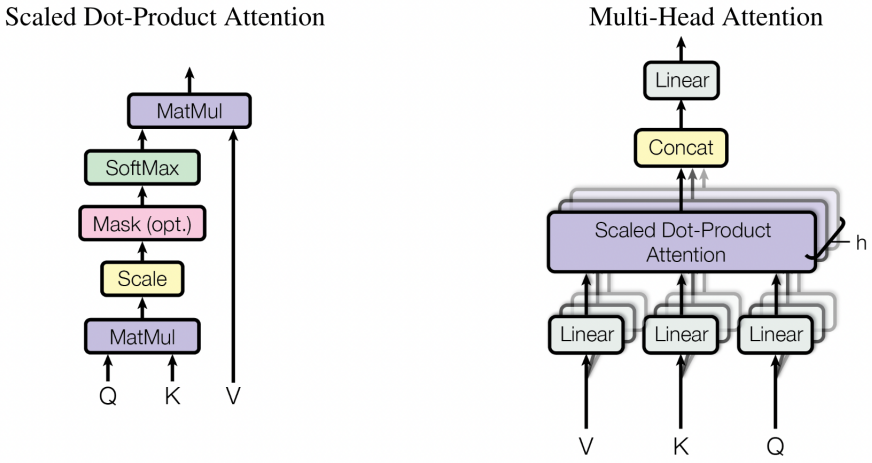

Figure 3.11: Attention mechanism (left) and multi-head attention mechanism (right) [56].

3.1.3 Linformer

Although the vanilla transformer has achieved extraordinary success in many tasks, training and deploying it for long sequences is very expensive and time-consuming due to the quadratic space and time complexity of the scaled-dot product attention. A vanilla transformer has time and space complexity of both $O(n^2)$, where n is the sequence length. To solve the problem, authors of Linformer [65] suggest an approximate way of calculating self-attention with linear space and time complexity $O(n)$. Linformer achieves the same performance as the vanilla transformer with much less complexity.

Linformer is based on the observation that self-attention is low-rank, and it can be proved both

theoretically and empirically that a matrix of much lower rank can closely approximate the scale-dot product attention matrix. Fig. 3.12 provides spectrum analysis of the self-attention matrix with $n = 512$. The Y-axis is the normalized cumulative singular value of the context mapping matrix P, and the X-axis is the index of the largest eigenvalue. The context mapping matrix P in MHA is defined as

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) = \underbrace{softmax[\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d_k}}]}_{P}VW_i^V \quad (20)$$

Authors performed singular value decomposition (SVD) on P across different layers and different heads of the vanilla transformer and averaged the normalized cumulative singular values over more than ten-thousand sentences. We observe that the curve tends to be saturated before 128, which implies that most of the information of matrix P can be recovered from the first few largest singular values. The heatmap in Fig. 3.12 shows the normalized cumulative singular value at the 128th largest singular value. The spectrum distribution in higher layers is more skewed than in lower layers, which means that, in higher layers, more information is concentrated in the largest singular values, and the rank of P is lower.

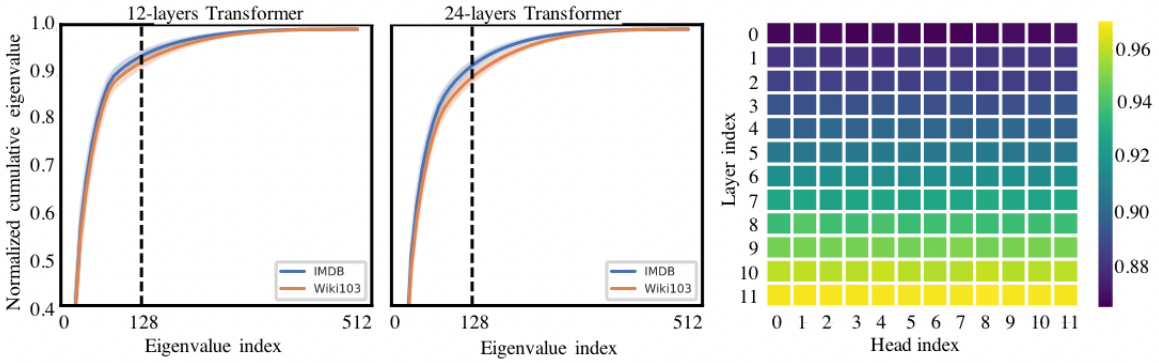

Figure 3.12: Proof of low-rank property of self-attention [65].

The low-rank matrix is obtained by calculating the SVD of the key and value matrices and only using a fixed number of the largest singular values. But in practice, since SVD calculation also adds computational complexity, the authors replace it with linear projection layers, as shown in

Fig. 3.13. These projection layers serve to reduce the dimensions of matrices. For simplicity, the trainable parameters of projection layers can be shared by attention heads and layers. Compared to the standard MHA, the Linformer computes attention by

$$\begin{aligned} \bar{head}_i &= Attention(QW_i^Q, E_iKW_i^K, F_iVW_i^V) \\ &= \underbrace{softmax[\frac{QW_i^Q(E_iKW_i^K)^T}{\sqrt{d_k}}]}_{\bar{P}:n\times k} \cdot \underbrace{F_iVW_i^V}_{k\times d} \end{aligned} \tag{21}$$

The first difference between Eq. 20 and Eq.21 is the latter constructs a similarity matrix $\bar{P} \in R_{n\times k}$, which is implemented by adding a linear projection matrix E to K. The second difference is the latter adds a linear projection matrix F to V that makes the original $n \times d$ matrix into $k \times d$. These two changes reduce the time and space complexity of the self-attention mechanism from $O(n^2)$ to $O(n \times k)$. If k is much less than n, the time and space complexity are approximately equivalent to $O(n)$.

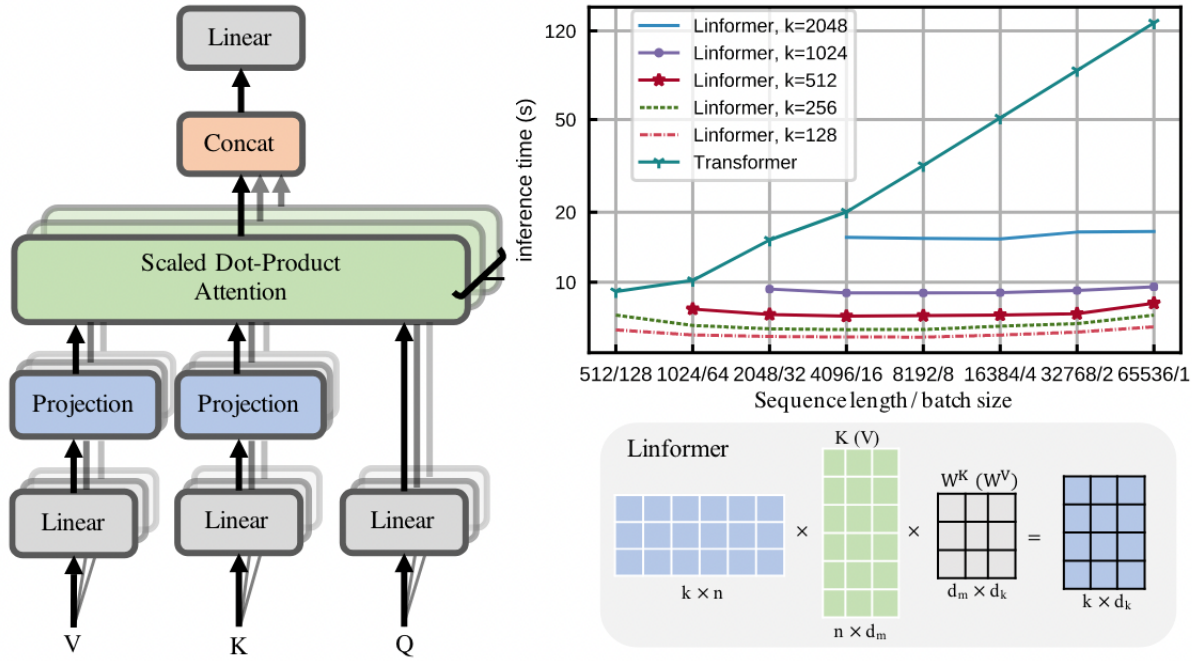

Figure 3.13: Linear-complexity attention mechanism of Linformer [65].

3.2 Proposed end-to-end VAD neural network

Existing DNN-based approaches for VAD were generally developed to learn acoustic features that were not designed specifically for the VAD task initially. Although end-to-end methods are rarely researched, they have the advantage that networks for feature extraction, encoding and classification can be trained simultaneously. This characteristic allows the overall model to have the best performance.

In Section 3.2.1, we overview the proposed model composed of a feature extractor, dual-attention transformer encoder, and classifier. Section 3.2.2 describes every module of the feature extractor and explains their design intuitions. We then elaborate on the dual-attention transformer encoder in Section 3.2.3. Section 3.2.4 finally presents a classifier without fully-connected layers. Parameter settings for the proposed end-to-end model are given in Chapter 4.

3.2.1 VAD framework using transformer

The block diagram Fig. 3.14 shows the framework of the end-to-end model. This model comprises a feature extractor, dual-attention transformer encoder, and classifier. The input is sample points of mixture audio generated by adding clean speech with different types and intense noise, which is then segmented into frames, as shown in Fig. 1.1. We assume the feature extractor is a learnable filter that continuously adjusts its parameters during training, which can learn to extract suitable acoustic features for the VAD task. It is not necessary to implement a whole encoder-decoder framework for classification problems. We only employ the dual-attention transformer encoder part to understand and encode features to other representations, allowing a simple classifier to discriminate speech segments from non-speech segments. Instead of using fully-connected layers with many trainable parameters, we employ an uncomplicated convolutional layer with a pooling layer as the classifier to produce labels of frames directly.

3.2.2 Feature extractor

Unlike calculating the manual features in chapter 2, we directly input sample point values of frames to the feature extractor to learn the acoustic feature. As shown in Fig. 3.15, the proposed

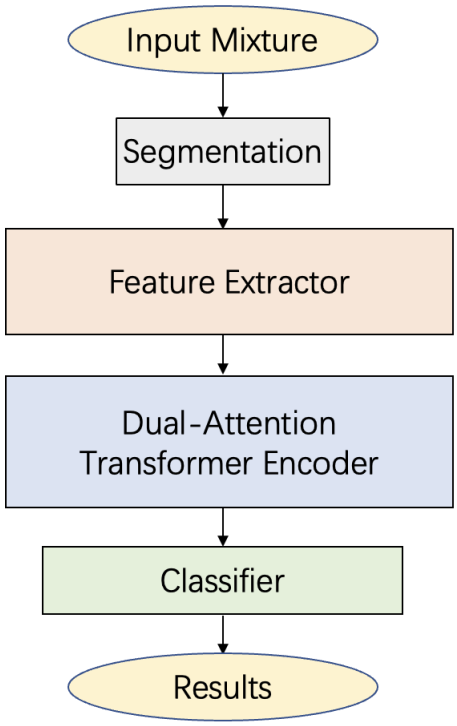

Figure 3.14: Framework of the end-to-end VAD model.

feature extractor can be divided into three parts: preprocessor, condense block and postprocessor.

The preprocessor includes a 2-D convolutional layer, a layer normalization (LN) [66] and a parametric rectified linear unit (PReLU) [67] nonlinear activation function. We know CNN is more efficient than MLP since the former only learns weights of local kernels with a finite spatial size. It means convolution kernels only know the information in their current receptive field but not where they are located in feature maps. However, through a comprehensive set of experiments, researcher of [68] sheds light on the fact that convolutional layers can implicitly encode positional information. Thus, we aim to use a single 2-D convolutional layer to encode the positional information of input sample points. The convolutional layer is configured with kernel size $(1, 2)$ and stride 2. The kernel will slide horizontally through all sample points of one frame, then repeat the same process for all frames. Referring to Eq. 7 and Eq. 8, the output feature maps have the half size of the original input, which helps reduce computation in the next layers. The following LN operates along the width dimension of output feature maps, which normalizes values of each frame to $[0, 1]$ to avoid the negative effects of extremely large or small values. The PReLU activation can adaptively learn

the parameters of the rectified linear unit and improve the accuracy with negligible computation.

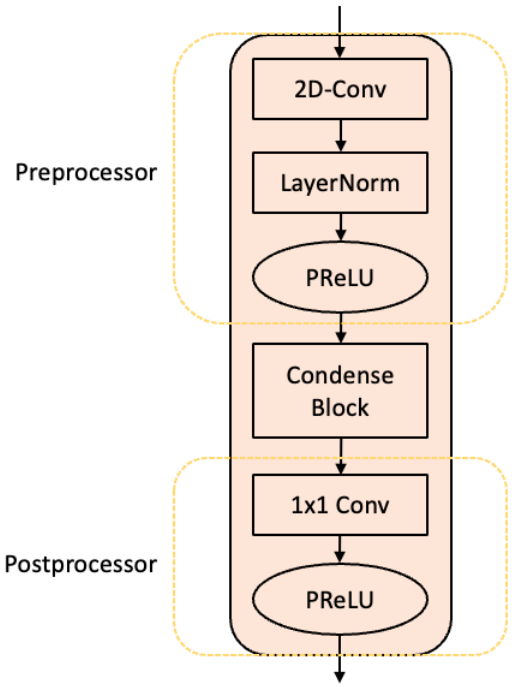

Figure 3.15: The composition of feature extractor.

The condense layer has been introduced in Section 3.1.1 and shown in Fig. 3.5. We employ 14 condense layers in the condense block to learn acoustic features since it has the following advantages: previous feature maps can be propagated to deeper convolutional layers through condense connectivity, which allows every condense layer to have collective knowledge from previous layers; the learned group convolution can learn sparse representations of networks, making networks more efficient.

The postprocessor comprises a pointwise convolutional layer and a PReLU activation function. The pointwise convolutional layer reduces the number of feature maps to lessen the computation in the following dual-attention transformer encoder.

3.2.3 Dual-attention transformer encoder

We are inspired by TSTNN [64] to propose the dual-attention transformer encoder. The two-stage transformer block in Fig. 3.16 is the core component of TSTNN, which consists of a local transformer and a global transformer. The local and global transformers have the same structure

but have different inputs. The input of the local transformer is sliced into frames, enabling the local transformer to learn features from individual frames. The input of the global transformer is sliced along features dimension so that information across frames can be learned.

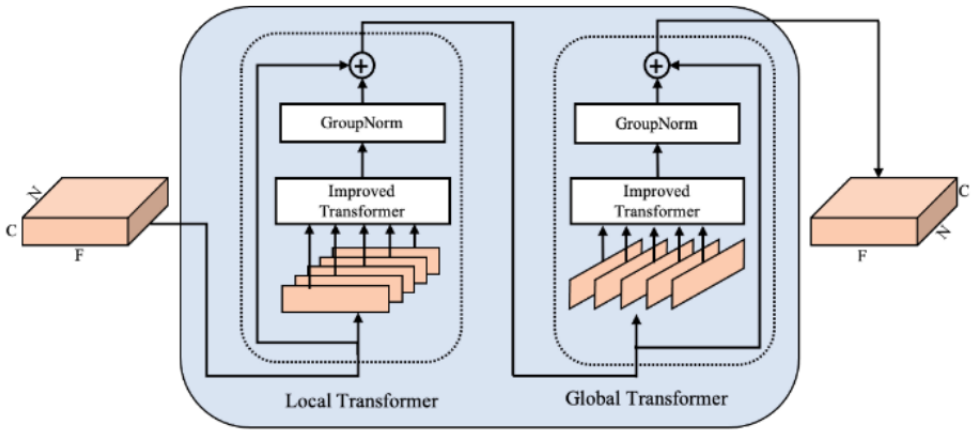

Figure 3.16: Two-stage transformer block [64].

The improved transformer of the two-stage transformer block is shown in Fig. 3.17. The improved transformer replaces one linear layer in FFN with GRU to learn sequence information. However, the employment of GRU will slow the inference speed of networks, a common problem of sequential models. Moreover, the cascading structure of local and global transformers requires the FFN twice, exacerbating the slow inference problem. Therefore, we make the following improvements to the two-stage transformer block to enhance performance while reducing time and space complexity:

(1) Employ a convolutional layer in the feature extractor to encode sequence information rather than using GRU in the FFN.

(2) Adopt linear-MHA rather than standard MHA to achieve linear complexity in time and space.

(3) Suggest a parallel dual-attention structure to learn local and global information, which only needs the feed-forward network once.

The structure of our proposed dual-attention transformer encoder is depicted in Fig. 3.18. The input is a three-dimension tensor with the shape of (C, N, F), where C is the number of channels, N denotes the number of features, and F represents the number of frames. The input is sliced

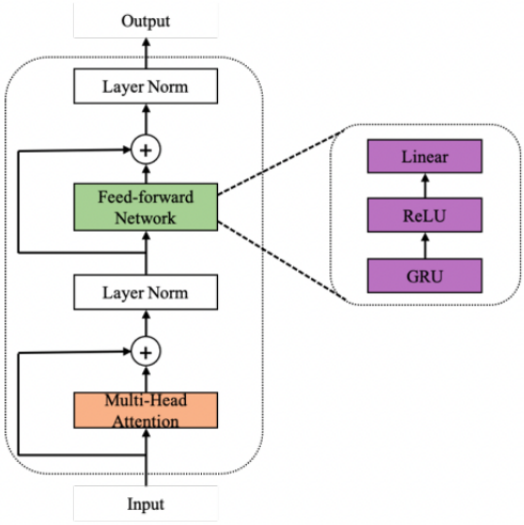

Figure 3.17: Improved transformer in the two-stage transformer block [64].

along N and F dimensions to get local and global input, respectively. We replace the MHA of the vanilla transformer with the linear-MHA to achieve linear complexity in time and space, where the linear-MHA has been introduced in Section 3.1.3. We then use the permute modules to reshape the outputs after adding attention so that they can directly add up after group normalization. FFN in the dual-attention transformer encoder maintains the same structure as FNN in the vanilla transformer.

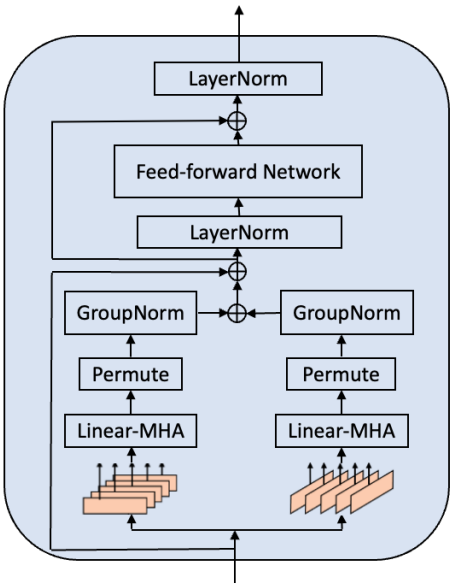

Figure 3.18: Dual-attention transformer encoder.

3.2.4 Classifier

The classifier comprises a PReLU activation function, pointwise convolutional layer and adaptive average pooling layer, as shown in Fig. 3.19. There are no fully-connected layers in the classifier since they bring many trainable parameters and much computation to the overall model. Here, we assume the shape of the dual-attention transformer encoder output as (C, N, F). The pointwise convolutional layer compresses the number of feature maps from C to 1 while retaining their salient features, which can be regarded as a channel-wise pooling operation. Max pooling and average pooling are two commonly-used pooling operations, as shown in Fig. 3.20. They enable the model to focus on the presence or absence rather than the locations of certain features. Average-pooling calculates an average value of a finite area to preserve the overall characteristics of data while max-pooling picks the largest value to preserve the local details of data. We can find that average pooling emphasizes the downsampling of the overall information, which has a more significant contribution to feature dimension reduction. Therefore, we use the adaptive average pooling layer to shrink the size of the feature map. The final output of the classifier labels every frame as 1 for speech or 0 for non-speech.

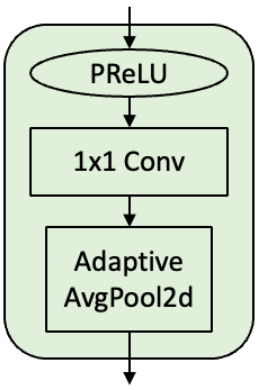

Figure 3.19: Proposed classifier.

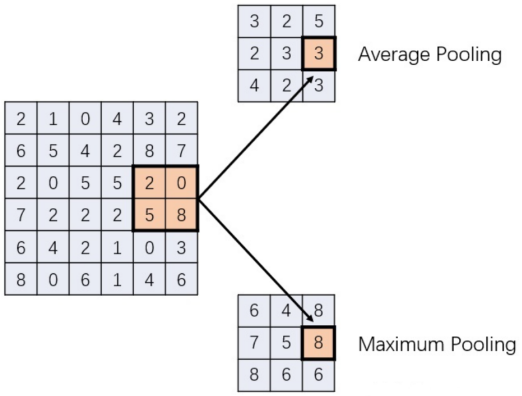

Figure 3.20: Max-pooling and average-pooling.

3.3 Summary

This chapter proposes an end-to-end VAD model, including the feature extractor, dual-attention transformer encoder and classifier. In the feature extractor, the standard convolutional layer first encodes sequence information to inputs, and the condense block learns to extract acoustic features. The next pointwise convolutional layer reduces the number of feature maps to lessen the computation in the following networks. Then, extracted features are input into the dual-attention transformer encoder. Features are divided locally and globally for dual-path linear-MHA to capture local-term and long-term information, respectively. After permuting dimensions, the two outputs are added and passed through the FNN. The proposed encoder greatly reduces the time and space complexity by replacing MHA with linear-MHA. Finally, the pointwise convolutional layer in the classifier compresses previous output into a single feature map, and the subsequent adaptive average pooling layer compresses features of frames smoothly into scalars as final prediction results.

Chapter 4

Experiments and results

In this chapter, we first introduce the experimental setup used for model evaluation, including datasets, comparison methods, parameter settings, performance and complexity metrics. Then, we discuss the performance difference between the proposed and baseline methods based on experimental results. Apart from that, we illustrate how different settings of models will affect their performance. Finally, we visualize the number of parameters, amount of computation and detection performance of all models in a ball chart.

4.1 Experimental setup

Section 4.1.1 presents datasets we used for training and testing and introduces the preprocessing of these datasets. Then baseline methods for comparison are presented in Section 4.1.2. Section 4.1.3 describes the settings of proposed models and the settings of hyperparameters during training. We introduce several evaluation metrics and their calculation process in Section 4.1.4.

4.1.1 Datasets

The TIMIT corpus is the source of clean utterances, which contains 6300 sentences recordings spoken by 630 speakers from 8 major dialect regions [69]. Every native speaker contributes 10 recordings by reading ten phonetically rich sentences. The TIMIT corpus is designed for acoustic-phonetic research and ASR systems development. It provides a clear directory hierarchy, including

pre-separated training and test sets, dialect directory, speaker directory, and word-level sample point annotations for each audio. Therefore, TIMIT is a qualified dataset for training and testing VAD models. Based on the observation of audio's ground truth labels, we find that speech segments make up the vast majority of audio segments. This leads to the problem of class imbalance causing deep learning-based models to learn much knowledge of human speech but little of non-speech. To alleviate the negative impact of this phenomenon, we added 1-second-long blank segments both before and after the original audio segments. We also revised the ground truth labels of audio segments after adding these blanks.

To achieve data augmentation, we mixed the training set of TIMIT with all types of additive noises from NOISEX-92 [70]. NOISEX-92 contains 15 different noises: white noise, pink noise, HF channel noise, speech babble, factory floor noise 1 and 2, jet cockpit noise 1 and 2, destroyer engine and operation rooms noises, F-16 cockpit noise, military vehicle noise, tank noise, machine gun noise, and car interior noise. For each additive noise, the generated noisy audios have five levels of SNR, i.e., -10, -5, 0 and 10 dB, which simulate different noisy conditions. We used 90% of the augmented training set for model training and 10% for model validation. In the test stage, we use the TIMIT test set as a clean speech source and AURORA [71] noise set as the unseen noise source. The AURORA contains eight types of noises: babble, airport, restaurant, exhibition, street, car, subway, and train. Similar to the data augmentation process in the training stage, we corrupted the clean utterances with all types of noises from AURORA. The test levels of SNR are still -10, -5, 0, 5, and 10 dB. The TIMIT test set contains different speakers' recordings compared to the training set, and the AURORA noise set is composed of unseen noises at the training stage. Therefore, our test can evaluate the generalization ability of different models.

To add noises to clean speech with different levels of SNR, we first calculated the root mean square (RMS) of clean speech and noise and denoted them as $speech_{rms}$ and $noise_{rms}$. The SNR is defined as

$$SNR = \frac{P_{speech}}{P_{noise}} = \left(\frac{speech_{rms}}{noise_{rms}} \right)^2 \tag{22}$$

where the P_{speech} is the average power of a clean speech signal and the P_{noise} is the average power

of a noise signal. It can also be expressed in decibels as

$$SNR_{dB} = 10\log_{10}[\left(\frac{speech_{rms}}{noise_{rms}}\right)^2] = 20\log_{10}\left(\frac{speech_{rms}}{noise_{rms}}\right) \tag{23}$$

we can adjust the SNR level of the noisy audio by multiplying a scalar to the $speech_{rms}$. We denote the expected SNR level as SNR_{exp}. Then, the scalar can be expressed as

$$scalar = 10^{\frac{SNR_{exp}}{20}}\left(\frac{noise_{ams}}{speech_{rms}}\right) \tag{24}$$

the noisy speech with a specified SNR level can be obtained by

$$speech_{noisy} = \frac{scalar \times speech + noise}{2} \tag{25}$$

4.1.2 Baseline methods

We adopt several unsupervised and supervised methods as baselines, and they are compared with the proposed methods in terms of performance and complexity. They are rVAD [72], WebRTC, CNN, 2-D CNN, CRNN, 2-D CRNN [73].

A. Robust VAD

The robust VAD (rVAD) method was proposed by Tan in [72] and it is one of the state-of-the-art unsupervised VAD methods. Fig. 4.1 shows the workflow of rVAD method. It contains three processing stages: the first pass of denoising using a posteriori SNR weighted energy difference measure [74], and the second pass of denoising using speech enhancement techniques and voice activity detection. Unlike other statistical VAD methods, rVAD performs two passes of denoising to enhance speech quality in the beginning. Because first-pass denoising can remove high-energy burst noise to avoid noise overestimation when applying a noise estimator in the second pass; moreover, if high-energy non-speech parts are not detected, it will be difficult for conventional VAD methods to deal with.

In the first pass of the denoising process, a posteriori SNR weighted energy difference is computed. It can be used to detect high-energy segments that contain speech and high-energy noise

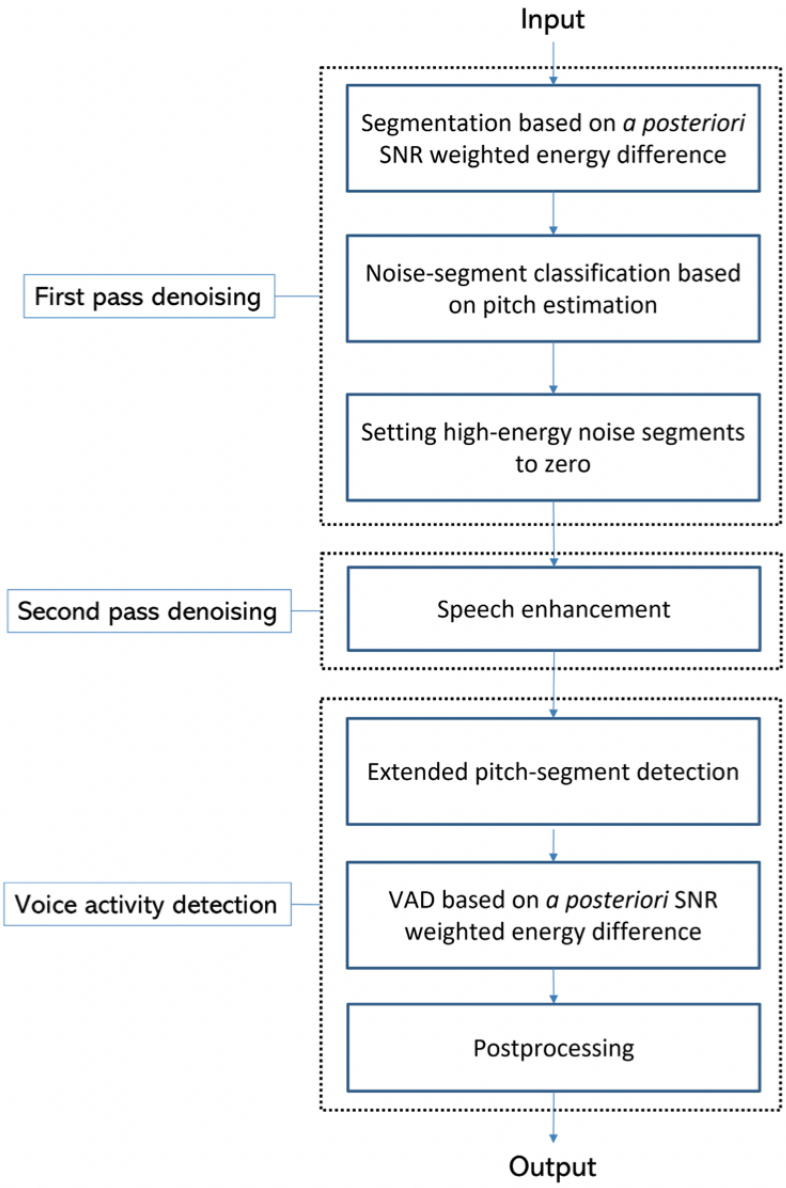

Figure 4.1: Workflow of rVAD method.

segments. Since human speech has pitch characteristics, pitch estimation is then performed to filter out non-pitch segments, which removes high-energy noise segments.

The author explored several speech enhancement methods for the second pass of the denoising stage. They are all spectral subtraction based, such as minimum mean-square error (MMSE) [75], minimum statistics noise estimation (MSNE) [76]. The author then proposed and employed a modified version of MSNE (MSNE-mod) for the second pass of denoising. If more than half of the energy is concentrated in the first few frequency bins, the second pass denoising process will set values of these frequency bins to zeros so that low-frequency noise can be removed.

In the last VAD stage, under the assumption that speech segments contain consecutive speech frames with pitch, neighbouring frames with pitch are grouped to form pitch segments. Pitch segments are then extended 60 frames from beginning and end so that voiced, unvoiced sounds and likely non-speech parts can be included. Finally, voice activity will be detected by comparing the computed smoothed a posteriori SNR weighted energy difference with the preset threshold.

B. WebRTC

WebRTC is an open-source visual-audio communication project proposed by Google company. It contains various speech-related applications, such as noise cancellation, automatic echo cancellation, and voice activity detection. WebRTC VAD is an unsupervised method based on GMM, and it has been widely accepted and used in industrial projects since its satisfactory performance and real-time running property.

C. CRNN and 2-D CRNN

The convolutional recurrent neural network (CRNN) was proposed to achieve VAD. The intuition of CRNN is to utilize CNN layers to capture audio information from both time and frequency domains. The RNN is then employed to identify time intervals of speech and non-speech, especially for long sequences. The author suggests that the combination of CNN and RNN suits the VAD task.

Fig. 4.2 depicts the structure details of CRNN. It contains five convolutional layers in the CNN part. The first convolutional layer has filter size of 16 and the following filter sizes of subsequent layers are increased as a power of two. They are 32, 64, 128 and 256, respectively. The first layer has kernels' size of 1×3, and there is a 1×2 max pooling layer following every convolutional layer to subsampling feature maps. Also, batch normalization and ReLU nonlinear activation follow each

convolution operation. In the RNN part of CRNN, there is two bi-directional gate recurrent unit (GRU) networks with all filter sizes of 126. Bi-directional GRUs can learn contextual information based on future and past values.

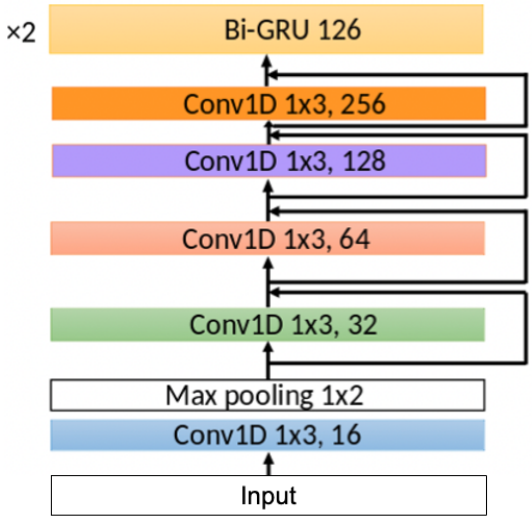

Figure 4.2: CRNN structure diagram [73].

The 2-D CRNN architecture is shown in Fig. 4.3. There are five convolutional layers and two bi-directional GRUs in this model. The first convolutional layer has 16 filters and uses 7×7 kernels. The second layer has 32 filters and uses 5×5 kernels. The rest convolutional layers have 32 filters and their kernels are all 3×3. Max pooling layers with 3×3 kernels and 2×1 strides are concatenated to every convolutional layer. Their 2×1 strides allow them to only sub-sample the features while reserving time information to be processed by the bi-direction GRUs. Batch normalization and ReLU follow every convolutions layer. Feature maps produced by the CNN part will be permuted and reshaped to fit the bi-directional GRUs which have filter size of 126 for each.

D. CNN and 2-D CNN

CNN and 2-D CNN models strictly follow the CNN part designs of CRNN and 2-D CRNN. But we reshape the output of CNN networks to a vector and then use fully-connected layers to produce the binary classification results.

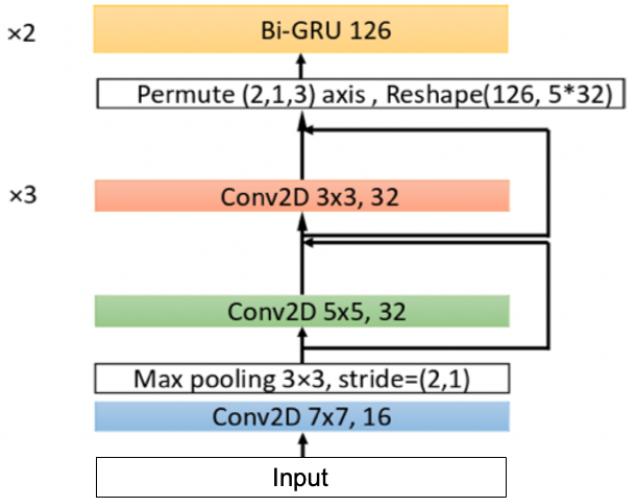

Figure 4.3: 2-D CRNN structure diagram [73].

4.1.3 Parameter setting

In this section, we present parameter settings for proposed models and hyperparameter settings for model training as follows:

A. Parameter setting for proposed computation-efficient model

The structure of the proposed computation-efficient model is shown in Fig.2.9. Here, we state the proposed model setting in Table 4.1. We hope to explore whether an even lightweight model can also achieve satisfactory performance. The parameter setting of the lite model is shown in Table 4.2, where t is the expansion ratio in an inverted block, c represents the number of output channels, m denotes the number of inverted blocks and s is stride.

Table 4.1: Parameter setting for the proposed computation-efficient model

Input	Operator	t	c	m	s
32×128×3	conv2d	-	32	1	2
16×64×32	bottleneck	1	16	1	1
16×64×16	bottleneck	6	24	2	2
8×32×24	bottleneck	6	32	3	2
4×16×32	bottleneck	6	64	4	2
2×8×64	bottleneck	6	96	3	1
2×8×96	bottleneck	6	128	1	1
2×8×128	conv2d 1×1	-	1280	1	1
2×8×1280	avgpool	-	-	1	-
1×1×1280	linear	-	2	1	-

Table 4.2: Parameter setting for the proposed computation-efficient lite model

Input	Operator	t	c	m	s
32×128×3	conv2d	-	32	1	2
16×64×32	bottleneck	1	16	1	1
16×64×16	bottleneck	6	24	2	2
8×32×24	bottleneck	6	32	3	2
4×16×32	bottleneck	6	64	3	2
2×8×64	bottleneck	6	96	1	1
2×8×96	conv2d 1×1	-	960	1	1
2×8×960	avgpool	-	-	1	-
1×1×960	linear	-	2	1	-

B. Parameter setting for proposed end-to-end model

Fig. 3.14 has given an overview of the end-to-end model. Here, the detailed parameter setting is given in Table 4.3, where k and s mean the kernel size and stride of convolutional layers and n means the number of operators. The condense block has 14 condense layers with a growth rate of 8. In each dense layer, the 1×1 learned group convolution has group size 4, condense factor 4 and dropout rate 0.2; the 3×3 group convolution has group size 4, and its padding and stride are both 1. In the dual-attention transformer encoder, we set the number of heads in the linear-MHA to 4 and the FNN first expands inputs by 4 times and then compresses back. We will discuss the impact of the number of layers in the dual-attention transformer encoder on the performance and complexity of the model later.

Table 4.3: Parameter setting for the proposed end-to-end model

Module	Operator	k	s	n	Output
Segmentation	-	-	-	-	1×299×320
Feature extractor	conv2d	(1,2)	(1,2)	1	16×299×160
	condense block	-	-	1	128×299×160
	conv2d 1×1	(1,1)	(1,1)	1	64×299×160
Dual-attention transformer encoder	-	-	-	1/2/3	64×299×160
Classifier	conv2d 1×1	(1,1)	(1,1)	1	1×299×160
	avgpool	-	-	1	299

C. Hyperparameter setting for model training

First, all audio segments, including clean speech and noise, are resampled to 16kHz before

augmentation. After the data augmentation, audio is segmented into 20ms-long frames, and the overlap between frames is 10 ms. The rVAD and the proposed end-to-end model directly take raw audio data (framed sample points) as inputs. The proposed computation-efficient model and other baseline models use different arrangements of MRCG and its dynamic information as inputs, where the dimension parameter d for each feature component is set as 128, and the parameter j for context range control is set to 16.

All baseline and proposed models are trained with SGD optimizer using momentum 0.9 and weight decay $4e^{-5}$, and cross-entropy is selected as the loss function. The maximum training epoch is 50. The learning rate varies dynamically with epochs [77]. Specifically, it linearly increases from 0.01 to the initial learning rate of 0.1 in 10 warm-up epochs. It then decreases to the final learning rate of 0.0001 in another 30 epochs using cosine decay strategy [78]. The learning rate remains at 0.0001 for the rest 10 epochs. The training and validation batch sizes are 2 and 1 for the proposed end-to-end method, while they are both 64 for the rest deep learning-based methods. We employ gradient clipping with a maximum L2-norm of 5 to avoid gradient explosion. The weights of all convolutional layers and linear layers are initialized by Kaiming normal initialization [67], while the initial weights of batch normalization layers are set as 1.

4.1.4 Evaluation metrics

To quantitatively evaluate the performance and complexity of all methods, we introduce several metrics here: the area under the ROC curve (AUC) and F1-Score for performance evaluation, the floating point operations (FLOPs) and the number of parameters for complexity evaluation.

A. Performance metrics

VAD is a binary classification problem, which means the performance of different methods can be measured by comparing prediction results with ground truth labels. There are four cases when comparing predictions with labels, and we depict all possible outcomes in Table. 4.4.

where TP, TN, FP, and FN represent true positive, true negative, false positive and false negative, respectively. TP and TN are outcomes where the model correctly predicts positive and negative classes, while FP and FN are outcomes where the model incorrectly predicts positive or negative classes.

Table 4.4: Confusion matrix

		Ground truth	
		Positive	Negative
Prediction	Positive	TP	FP
	Negative	FN	TN

Before introducing the AUC and F1-Score, we need to illustrate concepts of accuracy, precision and recall first. Accuracy is defined as

$$\begin{aligned} \text{Accuracy} &= \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \\ &= \frac{TP + TN}{TP + TN + FP + FN} \end{aligned} \tag{26}$$

where accuracy reflects what extent a model can correctly predict results. However, in some specific cases, FP and FN are more than critical, even if we have sufficiently high accuracy values. Also, accuracy is not a good metric when the dataset is class-imbalanced.

Precision attempts to illustrate to what extent the model's positive predictions are actually correct, so it is defined as

$$\text{Precision} = \frac{TP}{TP + FP} \tag{27}$$

while recall depicts the proportion of actual positives identified correctly by a model. It is defined as

$$\text{Recall} = \frac{TP}{TP + FN} \tag{28}$$

we can find that a model's precision is 1 if it produces no false positive, and its recall will be 1 if it produces no false negative results.

The F1-Score is the harmonic mean of the precision and recall, which can be calculated by Eq. 29. Its highest possible value is 1, indicating that the model has perfect precision and recall; its

lowest possible value is 0 when either precision or recall is zero.

$$\begin{aligned} \text{F1-Score} &= \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} \\ &= 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \\ &= \frac{2TP}{2TP + FP + FN} \end{aligned} \tag{29}$$

AUC is the area under the receiver operating characteristic (ROC) curve that plots two parameters: true positive rate (TPR) and false positive rate (FPR). The TPR is a synonym for recall. It can be calculated by Eq. 28. The FPR is defined as follows

$$\text{FPR} = \frac{FP}{FP + TN} \tag{30}$$

The ROC curve plots TPR vs. FPR at different classification thresholds. Fig. 4.4 depicts a typical ROC curve. AUC measures the entire two-dimensional area underneath the ROC curve from $(0, 0)$ to $(1, 1)$. The more the value of AUC is close to 1, the better the model's performance.

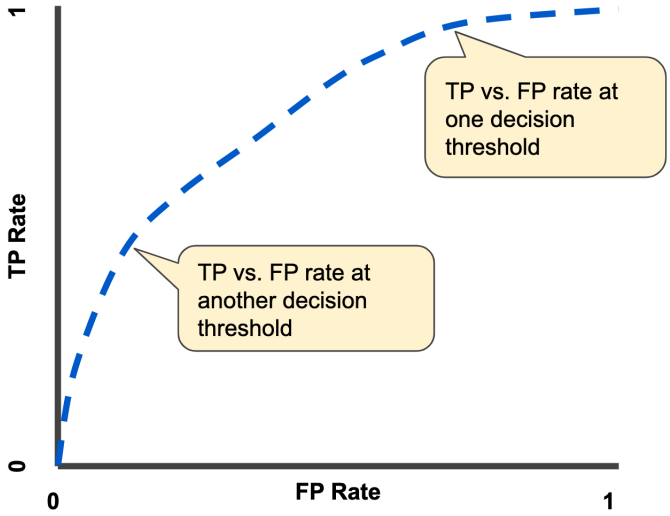

Figure 4.4: A typical ROC Curve.

B. Complexity metrics

As for measurements of model complexity, we employ two metrics: the number of parameters and FLOPs. The number of parameters reflects the size of a model, and FLOPs are used to describe

the theoretical amount of multiply-add operations required to run a single instance of a given model. When deep-learning models run on computationally-constrained devices, these two metrics are as important as model performance.

4.2 Results and discussion

In this section, we first quantitatively compare the performance of the proposed methods with baseline methods under different SNRs and scenarios. The comparison between the two proposed methods is also analyzed. We then investigate the impact of different parameter settings on models' performance. Finally, we visualize all deep learning-based models' performance and complexity in a ball chart.

4.2.1 AUC and F1-Score results comparison

Table 4.5 lists the AUC performance of baseline methods and two proposed methods in all eight different scenarios with five levels of SNRs, where the best-performing item is shown in bold. Here, we use CE-VAD to represent our proposed computation-efficient VAD model and DATE-VAD to denote our end-to-end VAD model using the dual-attention transformer encoder. From this table, we can intuitively find that our DATE-VAD method achieves the best performance in almost all test scenarios.

Table 4.6 ignores effects caused by scenarios and makes statistics on the performance of models under different SNR levels. Items in bold represent they have the best performance, while underlined items denote the second-best performance. From our observation, rVAD achieves surprisingly better performance than other baseline methods in 10 dB and 5 dB test environments, although it is a training-free method. However, its performance declines rapidly as the SNR decreases, which makes it incompetent for VAD in severe environments. Based on analysis of Table 4.6 and Table 4.5, although it is straightforward that WebRTC delivers the worst performance among all methods, it still has comparable performance in some specific scenarios, such as airport, car, exhibition, restaurant and subway.

After the analysis of unsupervised methods, we first compare the CNN and CRNN. The only

Table 4.5: AUC(%) comparison

Scenario	SNR(dB)	rVAD	WebRTC	CNN	2-D CNN	CRNN	2-D CRNN	CE-VAD	DATE-VAD
Airport	10	94.24	94.04	88.86	88.76	91.66	94.47	96.02	**99.18**
	5	93.95	89.28	80.65	86.11	87.68	90.85	93.91	**98.43**
	0	76.94	79.02	69.74	82.45	72.57	82.73	89.06	**97.97**
	-5	55.46	66.27	66.15	80.53	65.33	72.88	79.39	**97.87**
	-10	51.33	53.74	64.39	78.66	62.87	66.79	74.26	**96.36**
Babble	10	94.89	50.00	96.39	97.48	97.05	97.31	97.28	**99.45**
	5	94.02	50.00	96.03	97.32	96.78	96.74	97.12	**98.75**
	0	83.83	50.00	95.19	96.96	96.13	94.61	96.83	**98.81**
	-5	62.00	50.00	93.46	96.35	93.71	90.64	95.37	**98.37**
	-10	54.50	50.00	90.78	93.66	89.54	87.97	92.85	**97.19**
Car	10	92.66	84.87	96.92	94.06	97.20	95.64	97.35	**97.97**
	5	93.39	71.75	96.98	93.77	97.28	94.16	**97.34**	96.86
	0	92.46	57.48	95.34	91.89	95.74	91.01	95.16	**95.85**
	-5	89.09	50.50	95.49	91.47	**95.60**	89.42	93.89	93.40
	-10	85.74	50.00	93.42	89.86	**93.47**	88.57	89.78	89.06
Exhibition	10	92.32	92.92	95.34	87.27	95.61	93.57	96.16	**98.45**
	5	91.57	86.74	88.55	85.78	93.13	89.10	95.58	**97.98**
	0	87.42	71.55	75.07	81.03	79.07	81.82	90.69	**97.62**
	-5	81.09	56.09	69.21	77.33	63.90	76.74	78.32	**97.78**
	-10	73.38	50.92	64.95	75.20	59.38	74.98	70.13	**94.84**
Restaurant	10	94.14	91.55	72.89	73.62	74.60	70.64	92.76	**97.11**
	5	83.50	79.98	70.51	72.96	72.09	65.13	86.56	**95.57**
	0	71.74	65.68	65.68	69.05	68.37	56.65	78.28	**92.85**
	-5	63.06	52.78	62.67	67.10	64.62	54.45	71.96	**90.04**
	-10	58.87	50.13	60.85	65.76	61.87	53.61	67.85	**85.69**
Street	10	94.42	59.16	94.91	89.16	96.99	92.88	97.49	**99.07**
	5	91.96	59.12	91.90	87.53	95.89	90.24	97.02	**98.55**
	0	79.35	58.17	89.28	85.44	93.67	84.81	96.12	**98.64**
	-5	76.82	57.65	87.34	83.48	89.37	81.28	93.93	**97.44**
	-10	70.63	55.53	85.23	82.06	82.55	78.52	91.82	**97.36**
Subway	10	92.95	93.50	90.65	93.70	94.13	92.62	96.71	**97.30**
	5	93.94	88.10	88.18	92.38	88.82	89.94	95.45	**98.98**
	0	89.46	72.63	86.66	89.93	83.19	85.12	90.62	**96.42**
	-5	81.55	56.55	83.86	86.83	80.92	79.91	86.06	**95.86**
	-10	74.37	50.37	81.70	85.53	79.89	77.33	82.85	**92.35**
Train	10	93.43	53.04	96.41	96.44	97.19	95.82	97.76	**99.31**
	5	93.46	52.50	95.73	96.01	96.41	94.80	97.00	**99.14**
	0	92.54	52.58	95.30	95.79	94.66	93.12	95.51	**99.34**
	-5	91.34	52.83	94.47	95.28	92.82	92.34	92.87	**97.88**
	-10	81.80	52.59	93.96	94.73	91.91	92.69	90.39	**97.35**

difference between them is the RNN part, and they both employ the 1-D contextual-aware features. From Table 4.6, we find that adding RNNs improves the performance when the SNR level is above 0 dB, but performance worsens when SNR becomes severe. Then the comparison between CNN and 2-D CNN confirms that contextual-aware features in the single-channel 2-D arrangement can slightly improve the overall performance, especially under severe conditions. The worse performance of 2-D CRNN than that of CRNN may indicate RNNs can not properly encode the sequence information of convoluted feature maps. Except for the slightly lower AUC in -10 dB SNR environments, our proposed CE-VAD performs better than other baseline methods. The CE-VAD uses positional encoding to inject sequence information into features rather than to employ RNN to learn sequence information of feature maps, which greatly reduces computation cost. In addition, the multi-channel 2-D arrangement enables convolutional layers to capture features with their dynamic and contextual information simultaneously, further improving the model's performance.

Finally, we compare the two proposed methods: CE-VAD and DATE-VAD. DATE-VAD is an end-to-end method and it does not rely on prior knowledge of acoustic features. Specifically, it employs neural networks to learn acoustic features from raw audio data. In contrast, the CE-VAD takes the designed multi-channel features to emphasize characteristics of human speech. In Table 4.6, it is clearly shown that DATE-VAD improves the performance of CE-VAD by 2% to 11%, which is especially noticeable under low SNR conditions. The enormous improvement demonstrates that neural networks can learn acoustic features by themselves and achieve even better performance.

Table 4.6: AUC ($\%$) comparison under different SNRs

SNR(dB)	rVAD	WebRTC	CNN	2-D CNN	CRNN	2-D CRNN	CE-VAD	DATE-VAD
10	93.63	77.39	91.55	90.06	93.05	91.62	<u>96.44</u>	**98.48**
5	91.97	72.18	88.57	88.98	91.01	88.87	<u>95.00</u>	**98.03**
0	84.22	63.39	84.03	86.57	85.43	83.73	<u>91.53</u>	**97.19**
-5	75.05	55.33	81.58	84.80	80.78	79.71	<u>86.47</u>	**96.08**
-10	68.83	51.66	79.41	<u>83.18</u>	77.69	77.56	82.49	**93.78**

Table 4.7 provides the F1-Score comparison results of models, which provides consistent results with Table 4.6 and suggests that our proposed CE-VAD and DATE-VAD have better generalization ability than baseline models under most SNR conditions.

Table 4.7: F1-Score (%) comparison under different SNRs

SNR(dB)	rVAD	WebRTC	CNN	2-D CNN	CRNN	2-D CRNN	CE-VAD	DATE-VAD
10	93.54	82.46	93.40	92.20	94.52	93.50	96.82	**98.74**
5	92.25	74.92	91.29	91.43	93.02	91.71	95.80	**98.38**
0	85.97	58.79	88.28	89.81	89.12	88.20	93.12	**97.65**
-5	80.17	38.61	86.70	88.63	86.20	85.62	89.65	**96.49**
-10	76.19	27.66	85.05	87.47	83.63	84.53	86.93	**94.62**

Fig. 4.5 gives us an insight into how scenarios affect models' performance. The restaurant and street are two typical indoor and outdoor test environments. Though they are tested under the same SNR condition, the overall performance of models in the restaurant scenario is worse than in the street scenario. We suggest that outdoor scenarios mainly contain stationary and low-frequency noise that is much easier for models to handle. In contrast, indoors are often condensed with babble and other high-frequency noise that are difficult to deal with.

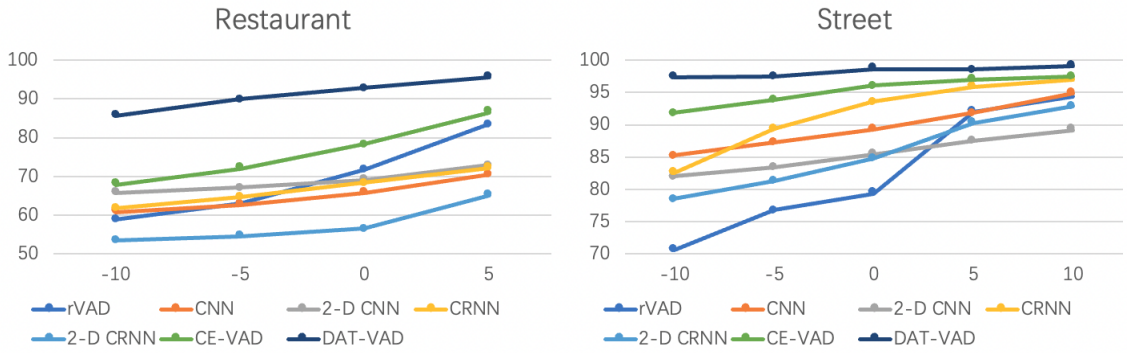

Figure 4.5: AUC (%) comparison under specific scenarios.

4.2.2 Performance comparison with different settings

A. Different settings for CE-VAD

Since the utilization of contextual-aware features, one of the most critical parameters is the number of neighbouring frames. To investigate how the number of neighbouring frames will affect the model performance, we conducted four experiments with 40, 32, 24 and 16 neighbouring frames including the current frame. The parameter j is set as 20, 16, 12, and 8. We tested the four cases with eight types of noise under 10 dB SNR and then averaged them to get the final AUC result. Table 4.8

describes test results that indicate the positive correlation between the number of neighbouring frames and FLOPs. Here, we used a public package in [79] to count the number of FLOPs. However, the performance does not follow the same rule, where 16 frames even can be comparable to 40 frames. But it shows a significant improvement after changing from 16 to 32 frames. Because the feature map size is always reduced by half, it will lead to information loss when the number is not the power of 2. We finally chose 32 neighbouring frames as a performance-computation trade-off solution.

Table 4.8: Discussion on different number of neighbour frames

# of neighbor frames	40	32	24	16
FLOPs(M)	66	48	42	24
AUC(%)	93.22	96.44	90.80	93.19

Apart from comparing different neighbour frames, we also conducted experiments on the lite-version model of CE-VAD. Based on our observations in Table 4.9, the lite-version model achieves noticeable performance improvements compared to baseline methods, especially in environments with SNR levels above 0. Although its performance declines drastically under -5 dB and -10 dB SNRs, the up to dozens of times FLOPs reduction demonstrates the effectiveness of our proposed CE-VAD structure.

Table 4.9: Comparison between CE-VAD lite model and other deep learning models

SNR(dB)	CNN	2-D CNN	CRNN	2-D CRNN	CE-VAD(lite)
10	91.55	90.06	93.05	91.62	**96.81**
5	88.57	88.98	91.01	88.87	**93.89**
0	84.03	86.57	85.43	83.73	**87.24**
-5	81.58	**84.80**	80.78	79.71	79.75
-10	79.41	**83.18**	77.69	77.56	74.03
FLOPs(M)	294	120	642	324	**30**

B. Different settings for DATE-VAD

The proposed DATE-VAD is an end-to-end model, and its core is the dual-attention transformer encoder. The number of layers in the encoder is critical for learning hidden patterns of extracted features. Therefore, we conducted experiments on the proposed model with one to three layers in

the dual-attention transformer encoder, and the results are shown in Table 4.10.

Table 4.10: Comparison between DATE-VAD models with different settings

SNR(dB)	DATE1-VAD	DATE2-VAD	DATE3-VAD
10	96.93	98.48	**99.16**
5	94.18	98.03	**98.76**
0	90.76	97.19	**97.85**
-5	83.76	**96.08**	95.85
-10	74.75	**93.78**	92.31

We have not tested the DATE-VAD with more layers because DATE-VAD will consume an enormous graphical memory footprint during the training stage. Limited by computer resources, we only take up to three-layer DATE for comparison. Even if the number of test results is limited, it is obvious that more layers in DATE bring promising advantages to model performance. From DATE1-VAD to DATE3-VAD, the overall performance is growing but the difference between DATE2-VAD and DATE3-VAD is not as noticeable as the difference between DATE1-VAD and DATE2-VAD. Thus, we choose to use the two-layer DATE in the end-to-end model.

4.2.3 AUC-operation-parameter ball chart

In this section, we intend to use a more intuitive way to show the performance-complexity trade-off process. The ball chart is a commonly used representation for deep learning-based model comparison. It takes advantage of simultaneously presenting the performance, size, and computation of multiple models in one figure. We first average deep learning-based models' performance under all SNR conditions to get the average AUC. The average AUC, the number of parameters and FLOPs are presented in Table. 4.11, which are the data source of the ball chart. In Fig. 4.6, the vertical axis represents the average AUC, the horizontal axis represents the amount of computation, and the balls' sizes represent the number of parameters. The ideal model is supposed to place in the left-up corner while having as small as possible size, meaning that the model has an acceptable size and achieves high performance with a fewer amount of computation.

In Fig. 4.6, we observe that our computation-efficient neural network CE-VAD places at the left-up corner of the ball chart and has competitive ball size, demonstrating that CE-VAD is a sweet

Table 4.11: Average AUC, number of parameters and FLOPs comparison

	CNN	2-D CNN	CRNN	2-D CRNN	CE-VAD	CE-VAD(lite)	DATE-VAD
Avg AUC (%)	85.03	86.72	85.59	84.30	90.39	86.34	96.71
# of params (K)	329	399	935	732	854	349	573
FLOPs (M)	294	120	642	324	48	30	2124

point of the performance-complexity trade-off. As for the proposed end-to-end model DATE-VAD, it advances other models in performance by a significant margin. In addition, it also contains a competitive number of parameters. However, its large amount of computation is notable, which can be attributed to the following reasons: the amount of computation of calculating manual features is not included in other models; if we use extracted features to represent audio, there would be less data than using original sample points of the audio; the DATE requires the largest amount of computation in the end-to-end model, which needs to be optimized in the future.

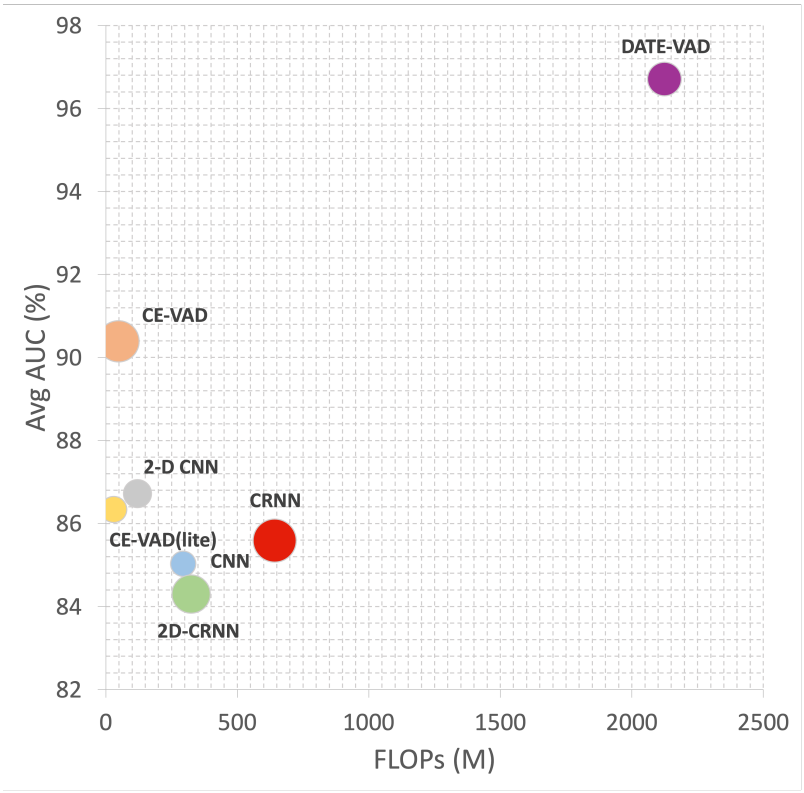

Figure 4.6: Ball chart for model comparison.

Chapter 5

Conclusion

5.1 Summary of the work

In this thesis, we thoroughly studied VAD in two directions: improving the detection performance of VAD in low SNR environments and reducing the complexity of VAD methods. Our proposed computation-efficient VAD neural network using multi-channel features has been proven to have superior detection performance than baseline methods while maintaining a few times smaller model size and the amount of computation. The proposed end-to-end VAD method significantly improves detection accuracy and robustness compared to other baseline methods.

Chapter 2 presented a computation-efficient VAD neural network using multi-channel features. We have introduced several features commonly used in speech-related classification tasks, such as MFCC, GFCC, PNCC and MRCG. After comparing the performance of adapting different speech features, we chose MRCG with its dynamic information as a feature for VAD. Then we investigated three arrangement methods of contextual features: 1-D, 2-D and multi-channel 2-D. The multi-channel 2-D arrangement allows CNN kernels to capture features' contextual and dynamic information simultaneously, contributing to better performance. The channel-attention inverted blocks are the core of the proposed computation-efficient neural network. Inverted blocks expand inputs to high-dimension representations to learn hidden patterns in extracted features, and the channel attention mechanism helps inverted blocks focus on important feature maps.

In Chapter 3, we have proposed an end-to-end VAD method. In contrast to current deep learning methods, our method allows neural networks to learn features from raw audio data rather than using manual features directly. The end-to-end model consists of a feature extractor, dual-attention transformer encoder and classifier. The feature extractor employs a condense block for feature extraction, a standard convolutional layer for raw data preprocessing and a pointwise convolutional layer for reducing the dimension of learned features. The following dual-attention transformer encoder processes extracted features from two different perspectives and uses the linear-MHA instead of the MHA to reduce significant amounts of computation. Finally, the classifier comprises a pointwise convolutional layer and an adaptive average pooling layer, which has excellent classification performance but only brings a few parameter increments.

Chapter 4 provided the proposed and baseline models' training and evaluation details. We compared all methods from performance and complexity perspectives. The AUC and F1-Score comparison consistently show that our proposed methods both outperform baseline methods. Especially the end-to-end method achieves tremendous performance advancements compared to other methods. In complexity comparison, our computation-efficient VAD neural network requires a fewer amount of computation while maintaining superior performance. Both of our proposed methods keep their models with a small number of parameters. Besides, we have discussed how models' parameter settings will affect their performance.

5.2 Future work

In this thesis, we mainly focused on developing deep learning-based VAD methods that are efficient and effective. The nature of supervised deep learning requires massive labelled data for model training. On the one hand, labelling new data for training is very labour-intensive and expensive. On the other hand, labelled datasets are always limited compared to real-life datasets. This limits supervised deep learning models to obtain even better generalization ability. A new concept of self-learning suggests that artificial intelligence can train itself using unlabeled data. It works by analyzing datasets and learning patterns of them to make conclusions. Self-learning can enable VAD models to continuously update themselves by learning new data without labels directly,

which makes it easy to enhance the generalization ability of VAD models. Besides, most of the state-of-the-art VAD models and ours only accept single-channel audio signals as input. However, real-world applications generally collect voices through multiple microphones, which requires us to develop VAD models capable of handling multi-channel audio signals.

In addition, the actual deployment of deep learning-based VAD models on computationally-constrained devices will be an exciting and promising topic. It involves both deep-learning model optimization for efficiency improvement and hardware knowledge for optimizing energy consumption.

Bibliography

[1] Z. Chen and B. Xu, "Optimization of speech endpoint detection base on sub-band energy feature," *Acta Acustica*, pp. 171–176, 2005.

[2] A. Benyassine, E. Shlomot, H.-Y. Su, D. Massaloux, C. Lamblin, and J.-P. Petit, "Itu-t recommendation g. 729 annex b: a silence compression scheme for use with g. 729 optimized for v. 70 digital simultaneous voice and data applications," *IEEE Communications Magazine*, vol. 35, no. 9, pp. 64–73, 1997.

[3] L. R. Rabiner and M. R. Sambur, "An algorithm for determining the endpoints of isolated utterances," *Bell System Technical Journal*, vol. 54, no. 2, pp. 297–315, 1975.

[4] L. Lamel, L. Rabiner, A. Rosenberg, and J. Wilpon, "An improved endpoint detector for isolated word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 4, pp. 777–785, 1981.

[5] K. Srinivasan and A. Gersho, "Voice activity detection for cellular networks," in *Proceedings., IEEE Workshop on Speech Coding for Telecommunications,*, pp. 85–86, IEEE, 1993.

[6] J. Ramırez, J. C. Segura, C. Benıtez, A. De La Torre, and A. Rubio, "Efficient voice activity detection algorithms using long-term speech information," *Speech communication*, vol. 42, no. 3-4, pp. 271–287, 2004.

[7] Y. Kobayashi and Y. Niimi, "Word boundary detection by pitch contours in an artificial language," in *ICASSP'80. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 900–903, IEEE, 1980.

[8] G. R. Rao and J. Srichland, "Word boundary detection using pitch variations," in *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, vol. 2, pp. 813–816, IEEE, 1996.

[9] J.-l. Shen, J.-w. Hung, and L.-s. Lee, "Robust entropy-based endpoint detection for speech recognition in noisy environments.," in *ICSLP*, vol. 98, pp. 232–235, 1998.

[10] A. Vahatalo and I. Johansson, "Voice activity detection for gsm adaptive multi-rate codec," in *1999 IEEE Workshop on Speech Coding Proceedings. Model, Coders, and Error Criteria (Cat. No. 99EX351)*, pp. 55–57, IEEE, 1999.

[11] G. Evangelopoulos and P. Maragos, "Multiband modulation energy tracking for noisy speech detection," *IEEE Transactions on audio, speech, and language processing*, vol. 14, no. 6, pp. 2024–2038, 2006.

[12] E. Chuangsuwanich and J. Glass, "Robust voice activity detector for real world applications using harmonicity and modulation frequency," in *Twelfth Annual Conference of the International Speech Communication Association*, Citeseer, 2011.

[13] J. Haigh and J. 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, pp. 321–324, IEEE, 1993.

[14] P. K. Ghosh, A. Tsiartas, and S. Narayanan, "Robust voice activity detection using long-term signal variability," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 3, pp. 600–613, 2010.

[15] I. V. McLoughlin, "The use of low-frequency ultrasound for voice activity detection," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[16] G. Aneeja and B. Yegnanarayana, "Single frequency filtering approach for discriminating speech and nonspeech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 4, pp. 705–717, 2015.

[17] X.-L. Zhang and D. Wang, "Boosting contextual information for deep neural network based voice activity detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 2, pp. 252–264, 2015.

[18] J. Sohn, N. S. Kim, and W. Sung, "A statistical model-based voice activity detection," *IEEE signal processing letters*, vol. 6, no. 1, pp. 1–3, 1999.

[19] D. Ying, Y. Yan, J. Dang, and F. K. Soong, "Voice activity detection based on an unsupervised learning framework," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2624–2633, 2011.

[20] S. Gazor and W. Zhang, "A soft voice activity detector based on a laplacian-gaussian model," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 5, pp. 498–505, 2003.

[21] J.-H. Chang, N. S. Kim, and S. K. Mitra, "Voice activity detection based on multiple statistical models," *IEEE Transactions on Signal Processing*, vol. 54, no. 6, pp. 1965–1976, 2006.

[22] Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 32, no. 6, pp. 1109–1121, 1984.

[23] S. O. Sadjadi and J. H. Hansen, "Unsupervised speech activity detection using voicing measures and perceptual spectral flux," *IEEE signal processing letters*, vol. 20, no. 3, pp. 197–200, 2013.

[24] P. Teng and Y. Jia, "Voice activity detection via noise reducing using non-negative sparse coding," *IEEE Signal Processing Letters*, vol. 20, no. 5, pp. 475–478, 2013.

[25] S. Mousazadeh and I. Cohen, "Voice activity detection in presence of transient noise using spectral clustering," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 6, pp. 1261–1271, 2013.

[26] B. Luo, Z. Pei, L. Xu, and D. L. Hu, "A new method based on hmms and k-means algorithms for noise-robust voice activity detector," in *Applied Mechanics and Materials*, vol. 128, pp. 461–464, Trans Tech Publ, 2012.

[27] T. Ng, B. Zhang, L. Nguyen, S. Matsoukas, X. Zhou, N. Mesgarani, K. Veselỳ, and P. Matějka, "Developing a speech activity detection system for the darpa rats program," in *Thirteenth annual conference of the international speech communication association*, 2012.

[28] J. Wu and X.-L. Zhang, "Efficient multiple kernel support vector machine based voice activity detection," *IEEE Signal Processing Letters*, vol. 18, no. 8, pp. 466–469, 2011.

[29] H. Veisi and H. Sameti, "Hidden-markov-model-based voice activity detector with high speech detection rate for speech enhancement," *IET signal processing*, vol. 6, no. 1, pp. 54–63, 2012.

[30] B. Carolina, "Multilayer perceptron explained with a real-life example and python code: Sentiment analysis." [Online]. Available: `https://carolinabento.medium.com/`.

[31] X.-L. Zhang and J. Wu, "Deep belief networks based voice activity detection," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 4, pp. 697–710, 2012.

[32] X.-L. Zhang and D. 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.

[33] J. Chen, Y. Wang, and D. Wang, "A feature study for classification-based speech separation at low signal-to-noise ratios," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1993–2002, 2014.

[34] T. Hughes and K. Mierle, "Recurrent neural networks for voice activity detection," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7378–7382, IEEE, 2013.

[35] S.-Y. Chang, B. Li, G. Simko, T. N. Sainath, A. Tripathi, A. van den Oord, and O. Vinyals, "Temporal modeling using dilated convolution and gating for voice-activity-detection," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5549–5553, IEEE, 2018.

[36] F. Eyben, F. Weninger, S. Squartini, and B. Schuller, "Real-life voice activity detection with

lstm recurrent neural networks and an application to hollywood movies," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 483–487, IEEE, 2013.

[37] X.-L. Zhang, "Unsupervised domain adaptation for deep neural network based voice activity detection," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6864–6868, IEEE, 2014.

[38] N. Moritz, T. Hori, and J. Le Roux, "Triggered attention for end-to-end speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5666–5670, IEEE, 2019.

[39] X. Wang, R. Li, S. H. Mallidi, T. Hori, S. Watanabe, and H. Hermansky, "Stream attention-based multi-array end-to-end speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7105–7109, IEEE, 2019.

[40] J. Kim and M. Hahn, "Voice activity detection using an adaptive context attention model," *IEEE Signal Processing Letters*, vol. 25, no. 8, pp. 1181–1185, 2018.

[41] Y. Lee, J. Min, D. K. Han, and H. Ko, "Spectro-temporal attention-based voice activity detection," *IEEE Signal Processing Letters*, vol. 27, pp. 131–135, 2019.

[42] Y. R. Jo, Y. K. Moon, W. I. Cho, and G. S. Jo, "Self-attentive vad: Context-aware detection of voice from noise," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6808–6812, IEEE, 2021.

[43] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," *Pattern recognition and artificial intelligence*, vol. 116, pp. 374–388, 1976.

[44] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

[45] Y. Shao and D. Wang, "Robust speaker identification using auditory features and computational auditory scene analysis," in *2008 IEEE international conference on acoustics, speech and signal processing*, pp. 1589–1592, IEEE, 2008.

[46] Y. Wang, K. Han, and D. Wang, "Exploring monaural features for classification-based speech segregation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 2, pp. 270–279, 2012.

[47] R. Wang, I. Moazzen, and W.-P. Zhu, "A computation-efficient neural network for vad using multi-channel feature," in *2022 30th European Signal Processing Conference (EUSIPCO)*, pp. 170–174, IEEE, 2022.

[48] J. Ramírez, J. C. Segura, C. Benítez, L. García, and A. Rubio, "Statistical voice activity detection using a multiple observation likelihood ratio test," *IEEE Signal Processing Letters*, vol. 12, no. 10, pp. 689–692, 2005.

[49] S. Hashemifar, "Depth-wise convolution explained in tensorflow." [Online]. Available: `https://soroushhashemifar.medium.com/`.

[50] L. Sifre and S. Mallat, "Rigid-motion scattering for texture classification," *arXiv preprint arXiv:1403.1687*, 2014.

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

[52] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[53] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[54] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and

H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[55] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[57] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[59] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2752–2761, 2018.

[60] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[61] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

[62] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[63] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for

deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[64] K. Wang, B. He, and W.-P. Zhu, "Tstnn: Two-stage transformer based neural network for speech enhancement in the time domain," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7098–7102, IEEE, 2021.

[65] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[66] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

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

[68] M. A. Islam, S. Jia, and N. D. B. Bruce, "How much position information do convolutional neural networks encode?," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[69] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, p. 27403, 1993.

[70] A. Varga and H. J. Steeneken, "Assessment for automatic speech recognition: Ii. noisex-92: A database and an experiment to study the effect of additive noise on speech recognition systems," *Speech communication*, vol. 12, no. 3, pp. 247–251, 1993.

[71] H.-G. Hirsch and D. Pearce, "The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *ASR2000-Automatic speech recognition: challenges for the new Millenium ISCA tutorial and research workshop (ITRW)*, 2000.

[72] Z.-H. Tan, N. Dehak, *et al.*, "rvad: An unsupervised segment-based robust voice activity detection method," *Computer speech & language*, vol. 59, pp. 1–21, 2020.

[73] A. Vafeiadis, E. Fanioudakis, I. Potamitis, K. Votis, D. Giakoumis, D. Tzovaras, L. Chen, and R. Hamzaoui, "Two-dimensional convolutional recurrent neural networks for speech activity detection," International Speech Communication Association, 2019.

[74] Z.-H. Tan and B. Lindberg, "Low-complexity variable frame rate analysis for speech recognition and voice activity detection," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 5, pp. 798–807, 2010.

[75] T. Gerkmann and R. C. Hendriks, "Unbiased mmse-based noise power estimation with low complexity and low tracking delay," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1383–1393, 2011.

[76] R. Martin, "Noise power spectral density estimation based on optimal smoothing and minimum statistics," *IEEE Transactions on speech and audio processing*, vol. 9, no. 5, pp. 504–512, 2001.

[77] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 558–567, 2019.

[78] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[79] S. Vladislav, "Flops counter for convolutional networks in pytorch framework." [Online]. Available: `https://github.com/sovrasov/flops-counter.pytorch`.