VOICE ACTIVITY DETECTION USING 1D TIME-CHANNEL SEPARABLE CONVOLUTION

Nguyen Duy Khanh

khanh14ph@gmail.com

1. Introduction

Up to now, there are many system related to speech processing. However, in an audio, the speech is not always continuous, there will be some parts exist without the present of speech. Therefore, we need a system to detect the speech part in an audio (Voice activity detector - VAD). A VAD system is a frame-level classifier to check if there is speech in a short audio segment. So when we apply it to whole audio, we will know the speech part of it.

Recently, there are many speech-based applications across mobile and wearable devices (e.g., virtual assistant). However, their memory and computing power is limited, which require lightweight VAD models. Therefore, I propose a model with just 74k parameters but still achieve a good result based on the architecture of MarbleNet [4].

It is a CNN model using 1D time-channel separable convolution and skip connection. By using separable convolution,the number of parameters drop significantly.

2. Related work

In the past, LSTM is a popular choice for sequential modeling of VAD tasks. However it has been observed that LSTM suffer from state saturation problems when the utterance is too long. Therefore, Hebbar et al [3] proposed a Convolutional Neural Network-Time Distributed (CNN-TD) model (with 740K parameters [3] that outperformed existing models including Bi-LSTM (with 300K parameters), CLDNN [7] (with 1M parameters) on the benchmark evaluation dataset AVA-speech [6].

3. Data

The dataset used is the Ava-speech dataset[1]. The dataset include videos.Each part of the videos is labeled with 1 non-speech class or 3 speech class with different conditions(clean speechm, speech with noise, speech with music). After processing, I got 22250 audio file belongs to speech class, and 17617 audio file belongs to non-speech class.

3.1. Segment the audio

Each sample we use for trainging and testing will be an audio segment with length 0.63s. So that the input 64 dimensional melfilterbank features calculated from 25ms windows with a 10ms overlap would be the shape 64x64 when provided to CNN models, for fair comparison with Hebbar et al [3]. We will clip respectively each 0.63s part from aforementioned files, with stride = 0.5s Since each audio segment is equal in length, all audio file sample rate must be 16000 and monophonic so all sample will present a vector with the same length.

3.2. Handle imbalanced data

Since the number audio segment in each class is not quite equal, the data would be "Oversampling"

	Original Data	**Rebalanced data**
speech training set	106564	106564
speech valid set	12958	12958
speech test set	13832	13832
background training set	88487	106564
background valid set	11513	12958
background test set	12070	13832

4. Data augmentation

To increase the generalization and avoid bad performance when dealing with audio with noise, each audio segment would be augmented

4.1. Add white noise

White Noise perturbation is performed by the following steps :

1) Randomly sample the amplitude of the noise from a uniformly distributed range (defined in dB)

2) Sample Gaussian noise (mean = 0, std = 1) with same length as audio signal

3) Scale this Gaussian noise by the amplitude (in dB scale)

4) Add this noise vector to the original sample

For step 1, the range [-90, -46] dB is applied.

4.2. Time-shift perturbations

Time shift perturbation is performed by the following steps :

1) Randomly sample the shift factor of the signal from a uniformly distributed range (defined in milliseconds)

2) Depending on the sign of the shift, we shift the original signal to the left or the right.

3) The boundary locations are filled with zeros after the shift of the signal

For step 1, the range [-5, 5] ms is applied.

5. Method

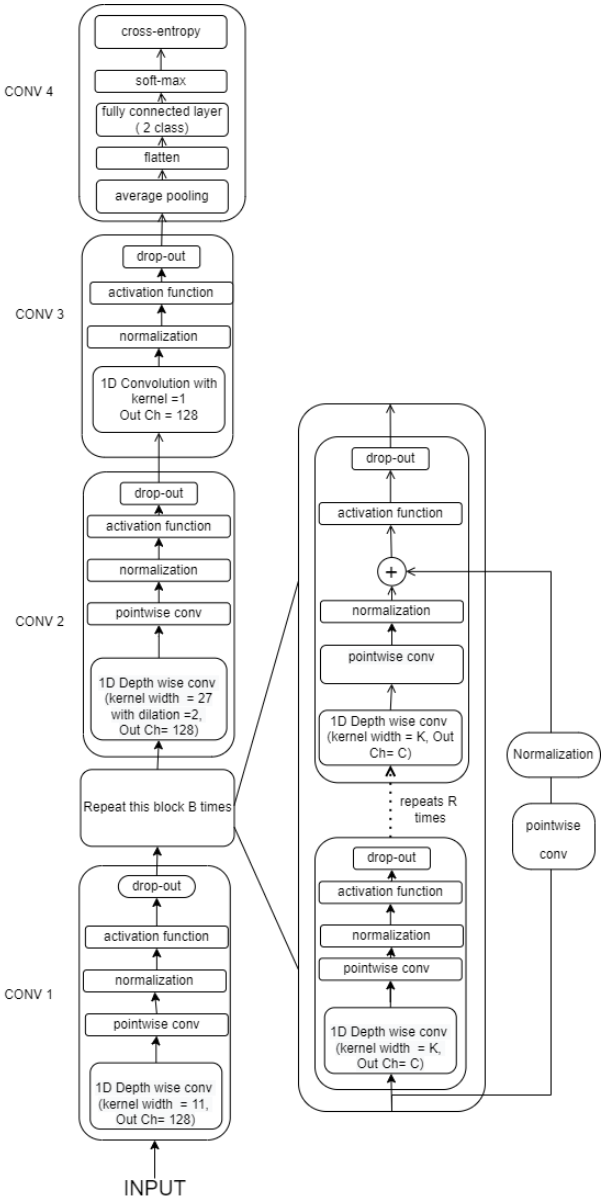
5.1. Feature extraction

The data is processed with 64 MFCC features since MFCC is a widely used technique for extracting the features from the audio signal .

5.2. Spectrogram Augmentation

After extracting feature, we applied SpecAugment [5] with 2 continuous time masks of size [0, 25] time steps, and 2 continuous frequency masks of size [0, 15] frequency bands. We also used SpecCutout [2] with 5 rectangular masks in the time and frequency dimensions

5.3. Network Architectures



The main purpose of the model is to reduce the number of parameters. Therefore the model is constructed of 1D time-channel separable convolution. 1D time-channel separable convolutions can be separated into a 1D depthwise convolutional layer with kernel length K that operates on each channel individually but across K time frames and a pointwise convolutional layer that operates on each time frame independently but across all channels.

A regular 1D convolutional layer with kernel size K, cin input channels, and cout output channels has K × cin × cout weights. The time-channel separable convolutions use K × cin + cin × cout weights (K × cin weights for the depthwise layer and cin × cout for the pointwise layer).It is clear that the number of parameters in two kind of convolution differ significantly.

The model includes B=2 residual blocks with R=2 sub-blocks for each block. All sub-blocks within each block have C=64 output channels.A basic sub-block consists of a 1D-time-channel separable convolution, 1x1 pointwise convolutions, batch norm, ReLU, and dropout. The 1D time-channel separable convolution has C filters with a kernel of the size K. All models have four additional sub-blocks: 'Conv1' before the first block, and three sub-blocks ('Conv2', 'Conv3', and 'Conv4')

block	sub blocks	output channel	kernel width
Conv1	1	128	11
B1	2	64	13
B2	2	64	15
Conv2	1	128	27,dilation=2
Conv3	1	128	1
Conv4	1	classes	1

6. Experiment

6.1. Loss function

Since this is just a binary classification problem, we will use Cross-entropy loss. It would measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. A perfect model would have a loss of 0.

6.2. Optimizer

After training the model with two kind of optimizer, "SGD with momentum" and "Adam". I found that "Adam" help the model converge faster. Therefore, Adam is the optimizer used in this research

6.3. Normalization method

By experimenting with four types of normalization: batch normalization, layer normalization, instance normalization , and group normalization, I found that models with batch normalization give the best results.

6.4. Metrics

For evaluation, denote non-speech class as the negative class, speech class as the positive class

6.4.1. Accuracy

Accuracy is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

6.4.2. Precision

Precision is defined as the ratio between all the instances that were correctly classified in the positive class against the total number of instances classified in the positive class. In a practical sense, precision tells you how much you can trust your

classifier when it tells you an instance belongs to the positive class. A high precision value means there were very few false positives and the classifier is very strict in the criteria for classifying something as positive. When precision equals to 1, it means that all the samples predicted true are correct, no negative sample blend in.

$$Precision = \frac{TP}{TP+FP}$$

6.4.3. Recall

Recall is defined as the ratio between all the instances that were correctly classified in the positive class against the total number of actual members of the positive class. A high recall value means there were very few false negatives and that the classifier is more permissive in the criteria for classifying something as positive.

$$Recall = \frac{TP}{TP+FN}$$

6.4.4. F1 score

F1 score is harmonic mean of precision and recall (supposed that these two value are not zero) The higher F1 score, the better classifier

$$F1score = \frac{2*Precision*Recall}{Precision+Recall}$$

6.5. Training

The model is build using Nemo toolkit of Nvidia (pytorch lightning based) on Google Colab with GPU Tesla T4. Input is a vector with size (10080,) (equivalent to 0,63s of 16000 sample rate audio).

Due to training, I found that the loss function tend to converge after 25 epochs.

Drop-out rate is set to 0.1 and batch size is set to 256(which is quite stable)

7. Result

Figure 1: Train loss value during training

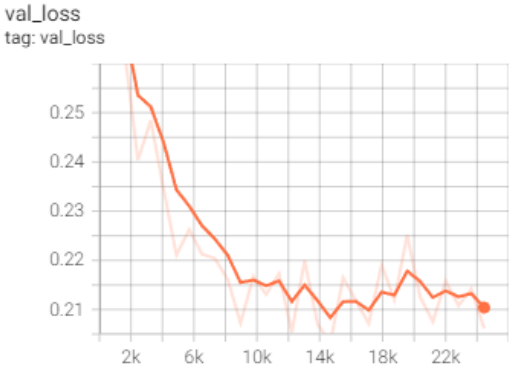


Figure 2: validation loss during training

Figure 3: train accuracy after training each batch

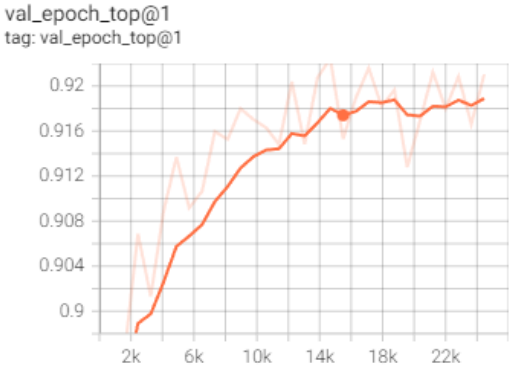


Figure 4: validation accuracy after each epoch

	All data	clean speech	with music	with noise
Accuracy	93.09%	96.1%	91.27%	92.9%
Precision	96.33%	96.28%	86.3%	91.3%
Recall	91.17%	96.35%	95.18%	93.5%
F1 score	93.67%	96.31%	90.52%	92.39%

7.1. Comparing performance with other model

In the paper, Hebbar et al [3] only show True positive rate (which is recall) result

	All data	Clean	Music	Noise
CNN-TD	94.5%	98.3%	91.7%	93.9%

It is clear that the CNN-TD model is slightly better.

7.2. Further testing

I used the model for testing a small part of Vin Bigdata's Vietnamese speech dataset. Since the data is quite clean, trivial in noise, the model achieve 97% TPR

7.3. Error analysis

Even though the model have a quite good result, it still have some limitation. When predicting the audio segment which has human sound (not speech) like laughing, crying, screaming, coughing, the model tend to label it as speech. This may result because they are all human sound like speech. To handle this, we can make a third class, named "non-speech human sound" in order to remove the mistaken of the model.

8. Conclusion

In this research, I build a light-weight model but still have a good performance. I hope this model can facilitate not only VAD but also other speech system on mobile and embedded devices.

Acknowledgement: I would like to thank mentor Đỗ Văn for his guidance and advice while I was doing this project

9. References

[1] Sourish Chaudhuri, Joseph Roth, Daniel PW Ellis, Andrew Gallagher, Liat Kaver, Radhika Marvin, Caroline Pantofaru, Nathan Reale, Loretta Guarino Reid, Kevin Wilson, et al. Ava-speech: A densely labeled dataset of speech activity in movies. *arXiv preprint arXiv:1808.00606*, 2018.

[2] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[3] Rajat Hebbar, Krishna Somandepalli, and Shrikanth Narayanan. Robust speech activity detection in movie audio: Data resources and experimental evaluation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4105–4109. IEEE, 2019.

[4] Fei Jia, Somshubra Majumdar, and Boris Ginsburg. Marblenet: Deep 1d time-channel separable convolutional neural network for voice activity detection. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6818–6822. IEEE, 2021.

[5] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

[6] Nicholas Wilkinson and Thomas Niesler. A hybrid cnn-bilstm voice activity detector. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6803–6807. IEEE, 2021.

[7] Rubén Zazo Candil, Tara N Sainath, Gabor Simko, Carolina Parada, et al. Feature learning with raw-waveform cldnns for voice activity detection. 2016.