Master's Thesis

Simple Depthwise Convolutional Neural Network for Efficient Keyword Spotting

효율적인 키워드 인식을 위한 간략 콘볼루션 신경망

AUGUST 2020

BY
QIAN XUE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Simple Depthwise Convolutional Neural Network for Efficient Keyword Spotting

Professor Sung, Wonyong

Submitting a Master's Thesis of Engineering

2020.08

Graduate School of Engineering
Seoul National University
Department of Electrical and Computer Engineering

QIAN XUE

Confirming the master's thesis written by
QIAN XUE
2020.08

Abstract

Keyword spotting (KWS) plays an important role in the current speech-based human-computer interaction, and is widely used on smart devices. With the rapid development of neural networks, various applications in speech related fields such as speech recognition, speech synthesis and speaker recognition have achieved great performances. Neural networks have become attractive choices for KWS architectures because of their good performance in speech processing.

However, since the application environment is mostly in small smart devices including smart phones, tablets and smart home devices, neural network architectures must consider the limited memory and computation capacity of these smart devices when designing a KWS system . At the same time, the KWS system should be able to maintain low latency in order to respond in real time. In addition, KWS is different from other tasks, because it needs to be always online and waiting for the call from the users, therefore, the power budget of the KWS application is also greatly restricted.

Among the mainstream neural network models, FCDNN (fully connected deep neural network), CNN (convolutional neural network), RNN (recurrent neural network) and the combination of them are mainly used for KWS in the past. Recently, attention-based models have become more and more popular. Among them, CNN is widely adopted in KWS, because of its excellent accuracy, robustness, and parallel processing capacity. Parallel processing capacity is essential for low-power implementations.

In this work, we present a neural network model-Simple Depthwise Convolutional Network, which supports an efficient keyword spotting. We mainly focus on a more compact Residual Network, and apply noise injection as an intermediate process to maintain high accuracy. Typically, ResNet always requires several hundred thousands parameters to achieve good performance. In our model, we employ depthwise convolutional neural networks to decrease the number of parameters, so that it can be more suitable for smart devices with limited resources. Finally, our model is tested on a real mobile device Samsung Galaxy S6 Edge, reality in the real inference time (that is, latency) of about 6.9ms, which is 17.5% faster than the state-of-the-art model TC-ResNet. The publicly available Google Speech Commands dataset is used to evaluate the models. The results show that we only use about one half of the parameters and at most 300 times fewer number of computations than the original base model, meanwhile, much smaller memory footprint yet maintain the 96.59% comparable high accuracy which outperforms the other state-of-the-art KWS models.

Keyword : Keyword Spotting (KWS), Convolutional Neural Network (CNN), Small Footprint

Student Number : 2018-24547

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Keyword Spotting System (KWS)

The rapid development of neural networks has made artificial intelligence possible, and has achieved good results in processing speech and images [1,2]. Neural network(NN) based KWS has achieved great popularity in the recent years [3,4,5,6,7,8]. The accuracy of machine recognition basically exceeds that of human recognition. As the most basic and direct way to interact with machines, speech plays an extremely critical role in artificial intelligence systems. In recent years, it has been used by major technology companies in the world for daily interaction on smart devices or smart homes. Speech recognition is mostly performed in the servers of service providers after the user's voice is transmitted. However, this server-based speech recognition has drawn attention regarding security and privacy. This is because the user's voice has been transmitted to the server, making it vulnerable to external attacks, and possibly leaking personal information to the outside [9]. In order to alleviate these concerns, on-device speech applications are needed. Before performing speech recognition, the device needs to be woken up and detect several predefined keywords. These predefined short contents consist of several characters called keywords. This process of detecting keywords by the device is called keyword spotting.

Keyword spotting is the first step of human-computer interaction based on speech. Thus, it is very important to detect the keywords very accurately, so that subsequent speech recognition can be activated. Then it is possible to perform interactive task operations. The process of keyword spotting is actually divided into these following steps: First, the device needs to stay online at all times, waiting for the user to give a call. When the user speaks out the keyword, the online device can receive audio signals in real time to quickly detect if it is recognized as a keyword, the device will wake up from the standby state to enter the interactive preparation state [10].

As introduced above, simply speaking, in some terms, keyword spotting is actually a simplified version of speech recognition. However, there is no decoding part like a language model, and the final task is to complete a classification task. In a typical process of keyword spotting using a neural network model, the entire system is roughly divided into two processes, as shown in Fig 1. The first one is the acoustic feature extractions, and the other one is the classification process based on the neural network model.

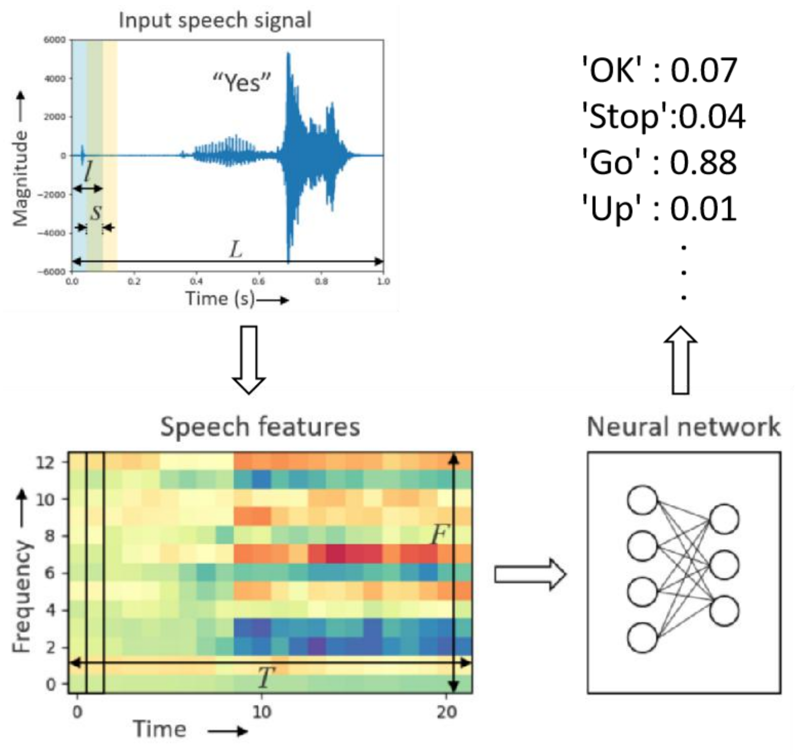


Fig 1. End to End keyword spotting system

The first step is the feature extractions, which is actually the same as that in speech recognition. The arriving speech signal is passed to the feature extraction module. If the speech signal length is L, a window function of length w is added, and s is the stride size. T frames are always obtained. Each frame extracts F-dimensional speech features through Mel-Frequency Cepstral Coefficients (MFCC) or Mel-Frequency Cepstral Banks (MFFB). Then, the entire input speech signal is converted into $T\text{x}F$ feature graph. In the second step, the two-dimensional feature matrix obtained above is transmitted to the classifier module. Finally the probability of the output category is obtained through the neural network model.

In addition to the end-to-end neural network-based KWS system described above, the traditional method also uses the keyword/filler hidden Markov model

(HMM) for recognition [11， 12], as shown in Fig 2. The key to this type of system is the decoding module on the lower side of Fig. 2. It is similar to the decoder in the HMM based speech recognizer. It also utilizes the Viterbi algorithm to obtain the optimal path, but it is similar to LVCSR (large-scale vocabulary continuous speech recognition). The difference from the speech recognition system is the specific construction of the decoding network. The decoding network in speech recognition contains all the words in the dictionary, while the wake-up decoding network contains the keyword and filler words on the upper side of Fig.2. The words excluding the keywords are all included in the filler path, and not every word will have a corresponding path. Such a network will be much smaller than a typical speech recognition network. When decoding keywords in a targeted manner, there are fewer optional paths, allowing the improvement of decoding speed. All of the other decoded candidates follow the same method to complete the overall framework. Although this method has achieved a reasonable performance in accuracy, it is still difficult to train, and it also requires a lot of computation process. Other technologies, such as RNN, are significantly better than HMM-based KWS in terms of accuracy [13]. Since RNNs have to wait for the previous steps, the structure demands a large delay, which is not ideal for KWS requiring a real-time response. Therefore, this article implements the variant network of CNN to perform KWS tasks.

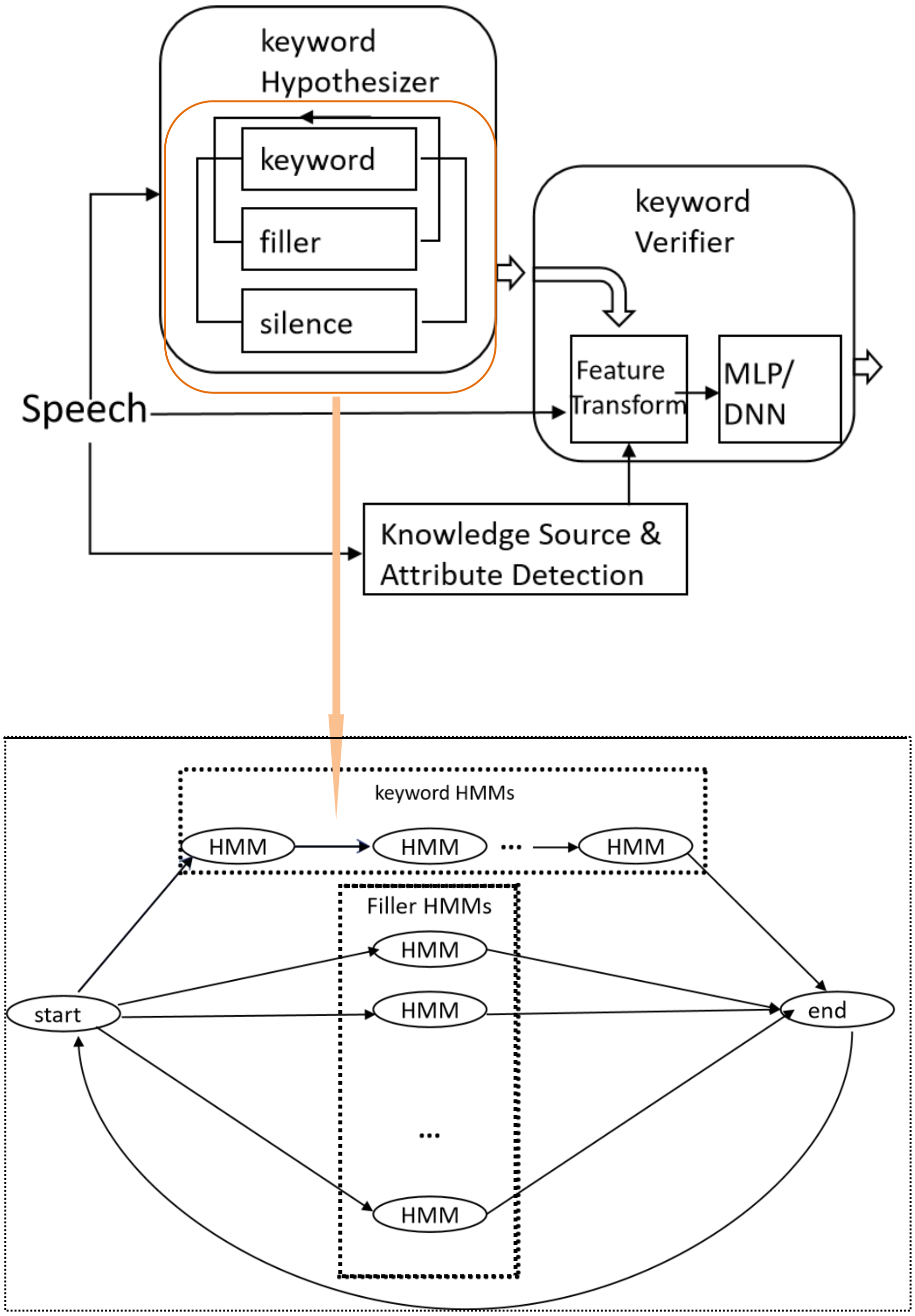


Fig 2. The Topology of HMM based keyword spotting system

1.2 Challenges in Keyword Spotting

As introduced in chapter 1.1, keyword spotting is usually considered as the first step of the human-machine interaction, mostly used on smart devices. There are basically four metrics for KWS.

The recall rate recalls to the number of times that it was correctly awakened as a percentage of the total number of times the keyword was detected. This value is better when it is larger.

The false alarm rate refers to the probability of keywords that should not be detected. A better the performance can be achieved with a lower value.

The real time factor is also one of the four metrics for KWS, which represents the response speed of the equipment.

Lastly, the power metrics is another metric for KWS. It is essential for portable devices.

Regarding the four metrics described above, there were some notable challenges. That is the trade-off between high accuracy and low power consumption, or high accuracy and low latency. In this thesis, we not only focus on the accuracy, but also pay attention on the latency. Usually power consumption is mainly affected by the capacity of hardware of devices and architecture of models we designed, which requiring us to deeply compress our model, so that less parameters and computations are demanded. However at the same time, this model should be able to maintain a high accuracy and faster speed comparable to that of the state-of-the-art models.

1.3 Neural Network Architecture for Small-Footprint KWS

There are some neural network architectures that are suitable for the on-device small footprint KWS. Among the mainstream neural network architectures, convolutional neural network (CNN) based models and ResNet based models show fairly great performances, especially the models outperform in accuracy [14,15] and showing low latency [15]. However, all of these ResNet models in previous studies consume quantities of parameters, doing lots of computations as the cost of pursuing high accuracy. As a result, the response speed is slowed by a considerable amount. The trade-off of these metrics is crucial for KWS, since KWS is commonly used on resource restrained devices. In this section, several latest researches will be explored including the architecture using self-attention which gained popularity in speech recognition, time delay neural network, temporal convolution combined with ResNet, and lastly with depthwise convolution.

1.3.1 TDNN-SWSA

This network is the time delay network with shared weight self-attention (TDNN-SWSA) [16]. TDNN is known as a classic network architecture and has achieved great success in recent speech recognition tasks [17]. In this study, they used TDNN here to capture local features, and shorten the length of the input before feeding it into the self-attention module. In addition, three matrices in the self-attention module [18] share the same matrix and are projected into the same single space. In this way, the number of parameters diminished sharply.

The schematic of the TDNN based subsampling is as shown in Fig 3. The length of the input is shortened to $(Tin - w + 1)/k$, where w is the length of the TDNN window Eq.(1) and Eq.(2) show the differences between the two different attention methods.

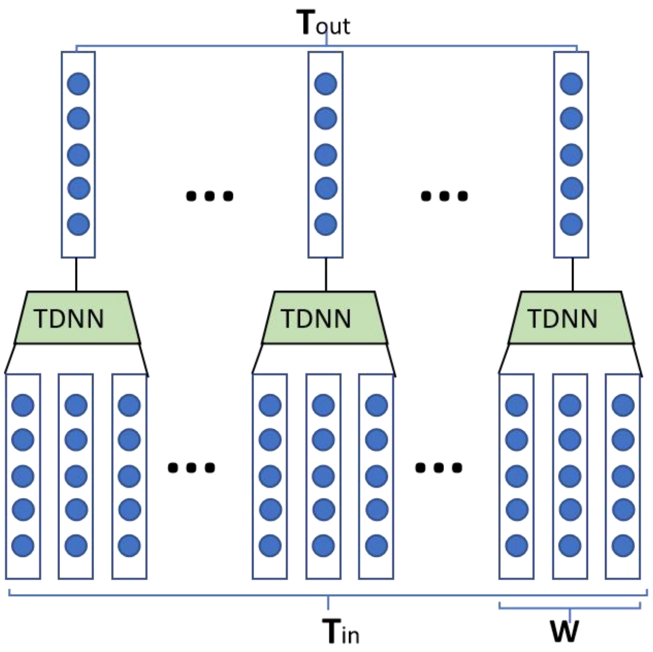


Fig.3 The schematic of TDNN-SWSA

The innovation of TDNN is the usage of the self-attention to share weights. In order to reduce the total number of parameters.

The traditional way of self attention is as follows:

$$\begin{aligned} &\text{Attend}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_k}}\right)\mathbf{V}, \\ &\text{where } \mathbf{Q} = \mathbf{U}\mathbf{W}_q,\ \mathbf{K} = \mathbf{U}\mathbf{W}_k,\ \mathbf{V} = \mathbf{U}\mathbf{W}_v. \end{aligned} \tag{1}$$

While the SWSA is represented as:

$$\begin{aligned} &\text{Attend}(\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{V}\mathbf{V}^T}{\sqrt{D}}\right)\mathbf{V}, \\ &\text{where } \mathbf{V} = \mathbf{U}\mathbf{W}. \end{aligned} \tag{2}$$

A shared weight matrix replaces three different matrices which correspond to queries, keys and values. In this way, the number of parameters are reduced sharply into 12k, only 1/20 of ResNet15, although there is some accuracy sacrifice.

1.3.2 TC-ResNet

TC-ResNet refers to Temporal Convolutional - Residual Networks [15]. Convolutional neural network (CNN) based KWS tasks have shown outstanding accuracy. This model applied temporal convolution, i.e. 1D convolution along the temporal dimension and took MFCC features as the input, as shown in Fig 4. Compared with 2D convolution, the output feature map size of temporal convolution is much smaller, which contributes to the drastic reduction of the computational burden in the next layers and its fast implementation. Another CNN architecture adopted is ResNet, specifically, ResNet8 and ResNet14. They changed the kernel size into 3x1 and 9x1, and expanded some channels in some of the model experiments.

By using temporal convolution, the burden of computation was lessened and the kernel looked at the whole range of frequency to improve the performance. TC-ResNet achieved the best accuracy in the model TC-ResNet14-1.5 with 96.6%. However the downsides of using large size of ResNet are consuming hundreds of thousands of parameters and requiring lots of computations, which leads to a large model size and increase in the inference time.

1.3.3 DS-CNN

DS-CNN refers to the depthwise separable convolutional neural network (DS-CNN) [19]. Depthwise separable CNN here is based on the implementation of stacking n many pure depthwise separableconvolutioins. A DS-CNN is composed of one depthwise convolution decomposes 3-D convolutions into 2-D convolutions, and one pointwise convolution which follows the depthwise convolution. Each convulotion was followed by a batch normalization [20] and the (Relu) activation function. N many DS-CNNs were stacked together to build up a DS-CNN model. The usage of depthwise and pointwise convolution made wider and deeper systems possible, even in the resource-constrained devices. Furthermore, 8bit quantization was performed here to compress the model size. We can take the advantage of efficiency in number of parameters, operations and model size.

1.4 Simple Depthwise Convolutional Neural Network for Efficient KWS

Through the previous study, we recommend a simple depthwise convolutional neural network for footprint KWS. Simple depthwise separable convolutional neural network is the simplest form of depthwise convolution, combined with residual neural network and we utilized this architecture in training our model.

Simple Depthwise Convolutional Network consists of 1-D depthwise convolution part and a ResNet part. ResNet-based KWS systems showed great performance. In order to keep a high accuracy, we used ResNet. The issue in utilizing ResNet that required solving was that it consumed too many parameters. To resolve this, we applied another way of implementing the 1-D depthwise separable convolutional neural network. Depthwise convolution is advantageous since it requires less parameters, making it different from the traditional convolutions. However, the lack of pointwise convolution makes it difficultto expand the feature map, resulting in an accuracy not as good as before. In order to make it learn neighboring channels, we chose simple depthwise convolution by looking through K channels (in our model, it means K features) and using T length as one input. In addition, we applied noise injection into weights, which improved accuracy at some level, since training with noise injection was helpful in finding a wide range of local minima in loss surface, and also avoid overfitting. Lastly, we trained on three different KWS datasets to make our system more robust.

Through this method, our simple depthwise convolutional neural network was able to maintain the best accuracy, while occupying much less parameters, smaller model size and faster speed .

1.5 Outline of the Thesis

More details about our recommended model will be introduced in the following pages. And the rest of this dissertation are organized as follows. In Chapter 2, simple depthwise convolution is illustrated, including the structure of simple depthwise convolution, its contribution and the results of some experiments when choosing the basic settings. Chapter 3 focuses on simple depthwise convolution with noise injection, the experiments on three different datasets, and the comparisons with the state-of-the-art models, especially on the accuracy, speed, the number of parameters and the computations. The last chapter gives a conclusion of the entire thesis.

Chapter 2

Simple Depthwise Convolutional Neural Network

This chapter includes four sections, and the first three are the introduction of basic models : the traditional depthwise separableconvolution, simple expanded depthwise convolution, and recommended network combined. The last part is composed of experiments and results of this model compared with other various networks .

2.1 Depthwise ConvNet

Depthwise ConvNet is the variant of traditional convolutional network, which is popular and has been employed in various fields such as in machine translation, computer vision and speech recognition [21,22].

Depthwise convolution as shown in Fig.5 is a part of depthwise separable convolution which was inspired from MobileNet [22]. The other part is pointwise convolution, as shown in Fig.6. Unlike conventional convolution operations, one convolution kernel of Depthwise Convolution single channel kernel is responsible for only one input channel. If the number of input channels is N, there must be N many channels or multiple of N many channels. Besides, each kernel is a single-channel. Every time, the single-channel kernel convolutes with the corresponding single channel of the input. Therefore, a N-channel input is processed to generate N feature maps (if there is same padding, the size is the same as the input layer). While in the conventional convolution, each convolution kernel operates simultaneously with every channel of the input.

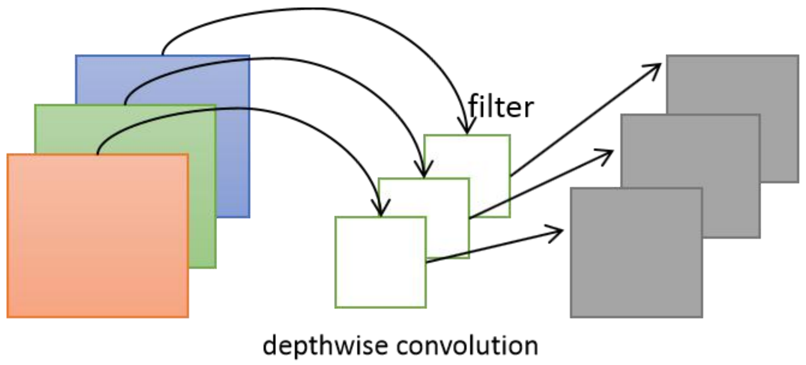


Fig.4 Depthwise convolution in depthwise separableconvolution

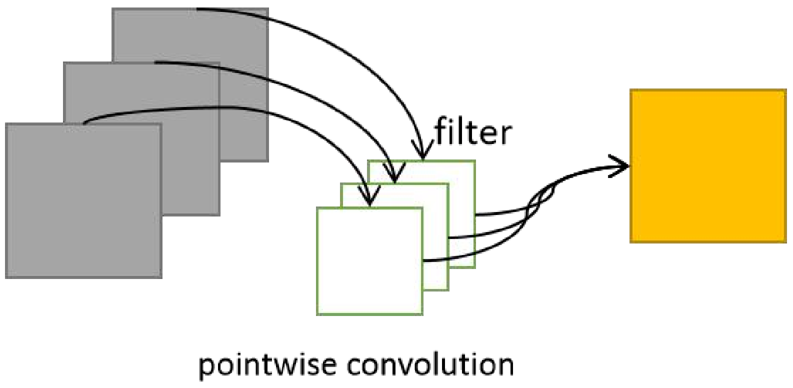


Fig.5 Pointwise convolution in depthwise separableconvolution

The output feature map of depthwise convolution cannot be extended. It is the input of pointwise convolution, and pointwise convolution operates the same convolution as depthwise convolution does, after which combining the output feature maps together as one new feature map. In this way, the model is able to effectively use feature information of different channels at the same spatial position.

2.2 Simple Depthwise ConvNet

First is the 1-D temporal convolutional neural network, rather than the 2-D convolutin in the left side of Fig.6. TC-ResNet chose to do 1-D convolution along the time axis in the right side of Fig.6 . If the kernel size is $3\text{x}3\text{x}1\text{x}C$ for 2-D convolution, $3\text{x}1\text{x}D\text{x}C'$ for 1-D temporal convolution, when they keep the same number of parameters, the MACs of 2-D convolution is is $3\text{x}C/\text{f}$ (C is 160, f is 40, here) times of 1-D temporal convolution. Thereby, 1-D convolution needs much less parameters and computations than 2-D convolution.

Simple Depthwise Convolutional Network is illustrated as the 1-D depthwise convolution as shown in the right side of Fig.7, which is different from the conventional depthwise convolution part of depthwise separableconvolution, and is slightly different from the 1-D temporal convolution mentioned above. 1-D depthwise convolution also convolutes along the time axis. Although the height of the kernel is not 1 but k, it is still 1-D convolution, because we set the kernel size to $K\text{x}T\text{x}1$. The reason why we stacked K here is that pure depthwise convolution cannot obtain the information from other channels, which may result in bad performance. Therefore, after padding, there are D input channels (from the first channel to Dth channel) on the first layer, D channels (from the second channel to (D+1)th channel) on the second layer and so on until k layers. K neighboring channels were stacked together, then each channel of the new stacked input was made sure it contained k neighboring channels and was convoluted by one kernel at the same time. If the kernel size is $3\text{x}1\text{x}D\text{x}C$ for the 1-D temporal convolution, $K\text{x}T\text{x}1\text{x}D$ for the 1-D depthwise convolution, while keeping the same number of parameters, the MACs of 1-D temporal convolution is $C'/9$ (assume k=3 t=9

C'=12, but all of the kernel size in TC-ResNet are $9\text{x}1$, therefore the value of this is about 4) times of 1-D depthwise convolution.

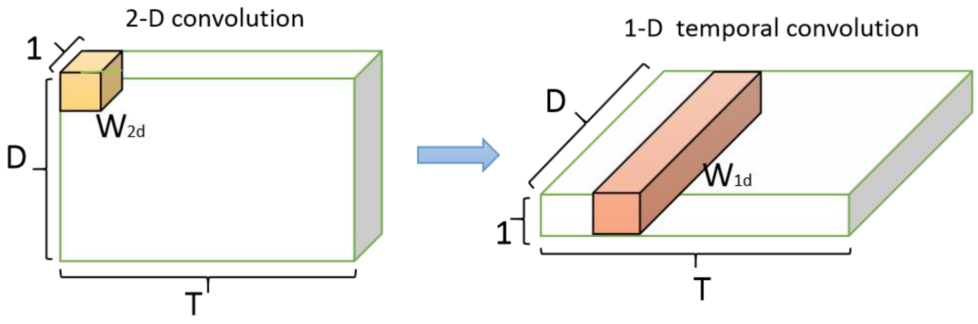


Fig.6 Compare traditional 2-D convolution with
1-D temporal convolution.

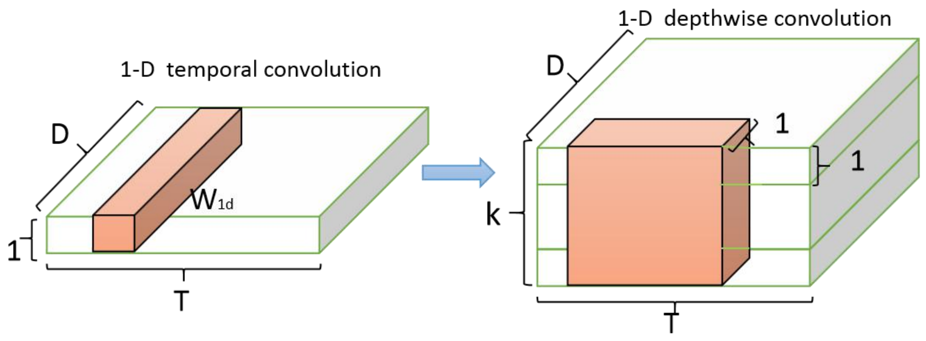


Fig.7 Compare 1-D temporal convolution with
1-D depthwise convolution

2.3 Residual Simple Depthwise ConvNet

As mentioned-above in section 1.3.3, the model merely used depthwise convolution neural network (DS-CNN), and performed better than the mainstream recurrent neural networks, including LSTM and GRU. However, the accuracy was not that high enough compared to state-of-the-art model. The newly proposed model, TC-ResNet in section 1.3.2, achieved the highest accuracy with 96.6% using ResNet14 with multiplier 1.5 (multiplier is used to expend channels in each block). It proved that ResNet architecture improves the accuracy, the only problem was that ResNet consumes a lot of parameters and memory. Therefore, we adopted the advantages of depth-wise convolution and residual network. The whole architecture of our model is shown in Fig 8. The audio signal after the preprocessing produced a speech feature representation, which is the input of the neural network. The first layer is a 1-D depthwise convolution followed by 3(for DC-ResNet8) or 6(for DC-ResNet14) blocks in which there are two 1-D depthwise convolutions connected with one residual connection. If the channels between the previous layer and the following layer are different, the stride of the first layer inside a block would be set to 2 and stride-2 1x1 kernel sized normal convolution module would be added into residual connection.

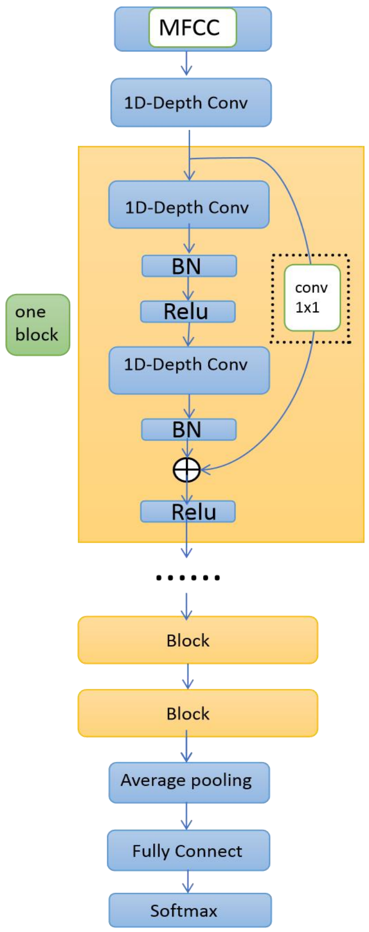


Fig.8 The Whole Architecture of DC-ResNet

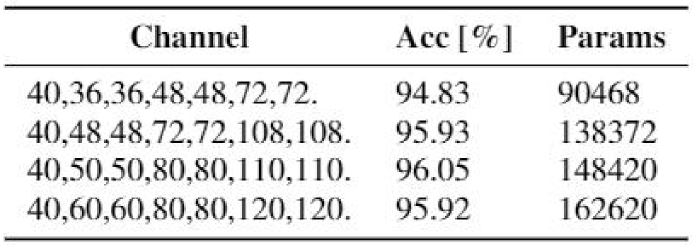
2.4 Experiments and Results

Dataset we used in this task is Google Speech Command Dataset(GSC) [23], which is specially designed for device controlling tasks. There are about 65000 recordings of 30 words, each of them being a second long. Among them, 10 of them are keywords and the rest are fillers, and all are labeled as '_unknown_'. In this experiment, there is one more class, which is the background, necessitating a total of 11 to 12 classifications. This dataset is used for training, validation and test, and the proportion of these three parts are 8:1:1 respectively.

The preprocessing of the data followed the previous study. Each one-second raw audio is decomposed into a sequence of frames by a window with length of 30 ms and shifted by 10ms stride at each time for feature extraction. We utilized 40 dimension MFCC feature representations for each frame, and stack them over time-axis.

First of all, we found the best base channels, without multiplier, according to the settings in the previous study. We set k=5 t=9 to find the best base model. N_channels = [40,50,80,110] was chosen as the base model as shown in Table1.

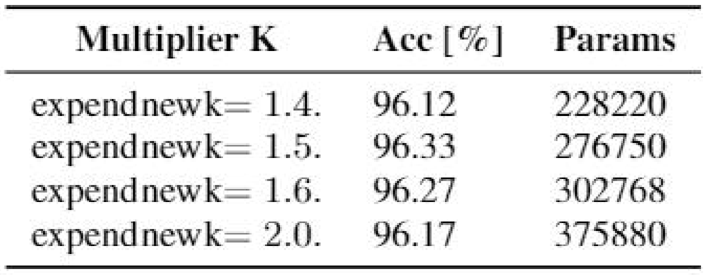
Table 1 Finding the base channel list.



Channel	Acc [%]	Params
40,36,36,48,48,72,72.	94.83	90468
40,48,48,72,72,108,108.	95.93	138372
40,50,50,80,80,110,110.	96.05	148420
40,60,60,80,80,120,120.	95.92	162620

Secondly, we used the multiplier to extend the number of channels. The experiments indicated that setting k as 1.5 improves accuracy the most, with relatively fewer parameters.

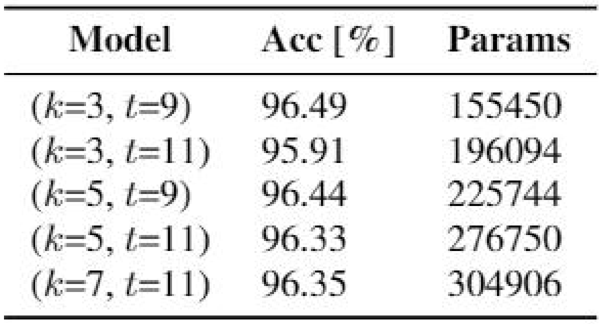
Table 2 Various multiplier K attempts.



Multiplier K	Acc [%]	Params
expendnewk= 1.4.	96.12	228220
expendnewk= 1.5.	96.33	276750
expendnewk= 1.6.	96.27	302768
expendnewk= 2.0.	96.17	375880

Thirdly, the accuracy and the number of parameters were computed with different values of K and T. Table 3 displays the results with different values of K and T. Nearly 10 cases with k = 3,5,7 and t = 7,9,11,13 were tested. These results display the best ones chosen in each K, which used the values of K=3 and T=9. It had an accuracy of 96.49%, while TC-ResNet had 96.6%(only 0.11% dropout). However, merely half the parameters were required.

Table 3 Results of different 1-D depthwise convolution kernel size.



Model	Acc [%]	Params
(*k*=3, *t*=9)	96.49	155450
(*k*=3, *t*=11)	95.91	196094
(*k*=5, *t*=9)	96.44	225744
(*k*=5, *t*=11)	96.33	276750
(*k*=7, *t*=11)	96.35	304906

Chapter 3

Robustness of Efficient Keyword Spotting

3.1 Weight Noise Injection

KWS is considered as a neural network based classification task, and we used the cross entropy for Keyword Spotting training. As proved in [16, 10], appropriately injecting noise into weights in training stage helps toavoid over fitting and being easily stuck in narrow local minima in the losssurface. Thereby, it helps to achieve a higher accuracy more or less. We were able to achieve the improvement of performance in our experiment. There are several different ways to operate weight injections such as Gaussian noise, uniform noise and smooth out. In this experiment we chose the uniform noise injection.

Traditional stochastic gradient descent method to update parameters:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta/B \sum_{i=1}^{B} \nabla \mathbf{L}_i(\mathbf{w}_t)$$

After the noise was injected :

$$\hat{\mathbf{w}}_{t+} = \mathbf{w}_t + \alpha \mathbf{n}_t$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta / B \sum_{i=1}^{B} \nabla \mathbf{L}_i(\hat{\mathbf{w}}_t +)$$

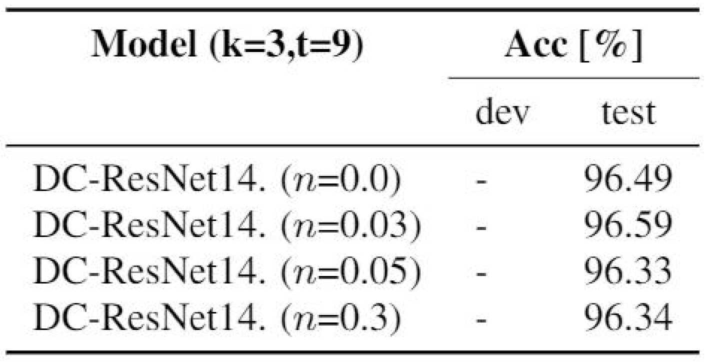
Where B is the batch size, and ‘η’ is the learning rate. In this noise injection, ‘α’ is the scale factor of noise injected into weights, and ‘n’ is exactly the noise vector which is randomly produced via uniformly distribution. The value of this noise was determined by the standard deviation of all the parameter weights. Through these efforts, our simple depthwise convolutional neural network was able to maintains a comparable accuracy, much less parameters, smaller model size, and faster speed.

In this part, I utilized the same dataset as the GSC mentioned above. Noise injection did improve the accuracy by 0.1% in our task, as shown in Table 4. Through a lot of experiments, we decided to use 0.03 as the scale factor ‘α’ in Table 5.

Table 4. Results of the models trained with the single weight injection. SW denotes the single weight injection, Acc denotes the accuracy.

Model	**Acc [%]**	
	dev	test
DC-ResNet14. (k=3, t=9)	-	96.49
DC-ResNet14. (k=3, t=9) + SW.	-	96.59
DC-ResNet14. (k=5, t=9)	-	96.44
DC-ResNet14. (k=5, t=9) + SW.	-	96.48

Table 5. Results of different noise scales. n denotes the noise scale, k,t represent the height and the weight of kernel size.



Model (k=3,t=9)	Acc [%]	
	dev	test
DC-ResNet14. (n=0.0)	-	96.49
DC-ResNet14. (n=0.03)	-	96.59
DC-ResNet14. (n=0.05)	-	96.33
DC-ResNet14. (n=0.3)	-	96.34

3.2 Experiments on Two Different GSCs

We trained and evaluated our model on an English dataset Google Speech Commands (GSC) [17], which is specially designed for controlling tasks on device. Three are two versions of this, and we experimented on each group. There are two versions of this dataset, and we performed two groups of experiments on these.

10 keywords of these two versions are the same: ‘yes’, ‘no’, ‘left’, ‘right’, ‘on’, ‘off’, ‘go’, ‘stop’, ‘up’, ‘down’.

3.2.1. Standard GSC

This version of GSC dataset is composed of 64752 recordings from various people for total 30 words, including 10 keywords, and the rest 20 words noted as fillers. Each recording is a one second long speech signal, and composed of only one word.Therefore, the classification classes in this experiment are 11 or10 keywords, all the other 20 fillers are collectively referred to as ”unknown”. According to the .txt files of validation and testing, the whole dataset is divided into three parts: 51088 recordings for training, 6798 recordings are used as validation set, and the rest 6833 recordings for evaluation.

3.2.2. Augmented GSC

This version of GSC dataset is composed of 64727 recording, and each of them is one second long and composed of 30 words. Experiments performed in section 2.4 and section 3.1 used this dataset. Furthermore, following Google's implementation [17], data was augmented with background noise. In addition to the 11 classes above, the 12th class, 'silence' was also produced. According to the .txt files of SHA-1 hashed name, the dataset is split into training, validation, and test datasets with 22246, 3093 and 3081 files respectively.

3.2.3 Experiments and Results

Experimental settings followed the setup in the previous work [15]. The window size and the stride was set up as 30ms and 10ms, respectively. Finally, the 40-dimension MFCC features were extracted. When training models, we set the dropout probability was 0.5, weight decay was 0.001, and applied 0.03 scaled weigh injection to optimize the loss. Learning rate started from 0.1 and dropped by 1/10 every 10k iterations, each models trained for 30k iterations in total.

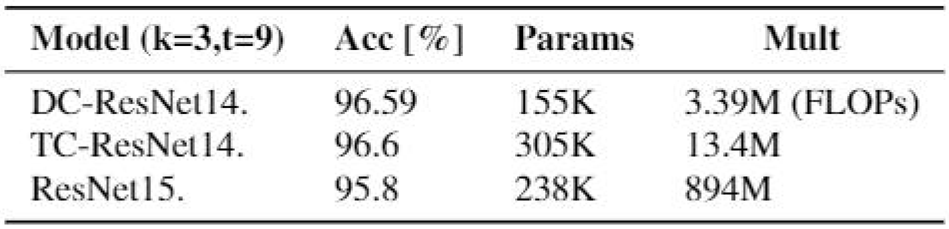
According to the different datasets, the experimental results were separated into two groups. First group was the comparisons among DC-ResNet, TC-ResNet, ResNet15and DS-CNN, using the augmented GSC dataset. Second group was the comparisons among DC-ResNet, TDNN-SWSA, ResNet15 experimented on standard GSC dataset.

Table 6 illustrates several different architectures. We found that our model (DC-ResNet14) keeps a comparable accuracy with the state-of-the-art model, consuming the least parameters, and the merely 3.39M FLOPs [24] instead of Multipliers displayed by the other two models. Table 7 focuses on the latency between two models. We tested the real inference time using Tensorflow Lite Android benchmark tool. The value 8.4 is different from 5.7 in paper because we tested the inference time on other mobile device, the Samsung Galaxy S6 Edge.

In Table 7, DC-ResNet responds 17.5% faster than the latest model, with the best performances so far, and maintains a high accuracy, and at the same time, our model only employed 1/2 of the parameters.

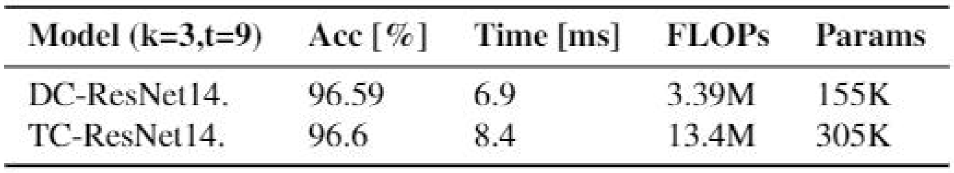
Table 8 displays diverse architectures tested on Standard GSC. Compared to the ResNet15, the accuracy of DC-ResNet dropped slightly. However when we traded off these three metrics illustrated on the table together, it was still fairly acceptable. Particularly, the FLOPs. FLOPs of DC-ResNet is half of TC-ResNet, 1/6000 of ResNet15.

Table 6. Results of various models on augmented GSC



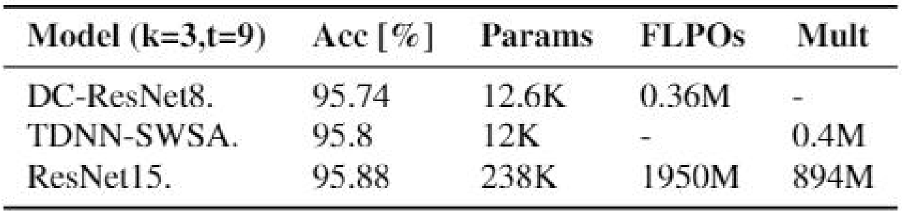
Model (k=3,t=9)	Acc [%]	Params	Mult
DC-ResNet14.	96.59	155K	3.39M (FLOPs)
TC-ResNet14.	96.6	305K	13.4M
ResNet15.	95.8	238K	894M

Table 7. The performance results for DC-ResNet and TC-ResNet



Model (k=3,t=9)	Acc [%]	Time [ms]	FLOPs	Params
DC-ResNet14.	96.59	6.9	3.39M	155K
TC-ResNet14.	96.6	8.4	13.4M	305K

Table 8. Results of various models on a standard GSC



Model (k=3,t=9)	Acc [%]	Params	FLPOs	Mult
DC-ResNet8.	95.74	12.6K	0.36M	-
TDNN-SWSA.	95.8	12K	-	0.4M
ResNet15.	95.88	238K	1950M	894M

3.3 FRR and FAR in a Third Dataset

3.3.1 FRR and FAR

In KWS system, besides the four main metrics introduced above, receiver operating characteristic (ROC) curve, and area under the curve (AUC) are also crucial. In KWS, x-axis of ROC is the false alarm rate (far), y-axis is the false reject rate (frr), AUC is the area under the ROC curve.

False alarm rate is the same as the false positive rate (fpr). False reject rate is also same as the false negative rate (fnr). Summing the false negative rate up with true positive rate is 1, where true positive rate is the recall rate. i.e.:

False alarm rate = False positive rate = FP / N = FP / (FP + TN)

False reject rate = False negative rate = FN / P = FN / (FN + TP)

= 1- True positive rate = 1- recall rate

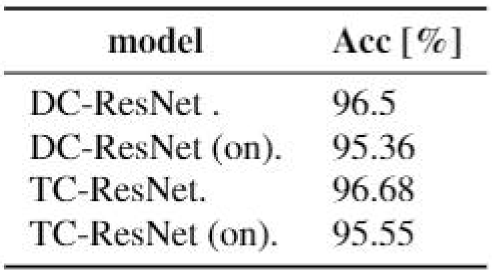
('T(/F)P(/N)' is the number of True(/False) Positive (/Negative) samples)

3.3.2 Third GSC

This third GSC was modified from the augmented GSC in 3.2.2. In the previous augmented GSC, there are only 6 background noises in the dataset, such as pink noise, white noise, do the dishes, etc... These are all silent noise. In the real world, there is much more audio speech noise. Therefore, we added 6 more real life background noise files, including the TED speech, CNN news, White house briefing, etc.... Most of them are conversation-s across different ages, genders, different accents, and tried our best to cover variant speech features.

Testing on augmented data challenging. As shown in Table 9, the accuracy of both models decreased by around 1.14%.

Table 9. The test accuracy (on)/(not on) third-augmented real life data. (“On” means trained and tested on third GSC. Otherwise, only trained on third GSC, tested on test set of augmented GSC in 3.2.2)



model	Acc [%]
DC-ResNet .	96.5
DC-ResNet (on).	95.36
TC-ResNet.	96.68
TC-ResNet (on).	95.55

3.3.3 Experiments and Results

In the first experiment, the model was trained on third - augmented real life dataset, but tested on normal augmented dataset in 3.2.2. We attained one tested TC-ResNet ROC, and one tested DC-ResNet ROC, as plotted in Fig 9, respectively. As shown in Fig 10 and Fig 11, we trained and tested both models on third - augmented real life dataset for 5 times. From the figures below, we can find that they are fairly similar in AUC, AUC of both the two models are around 0.998~0.999. Although more background noise was injected, the trend of the scatter distribution did not have a dramatic change, as shown in Fig11, which proves robustness. At last, we compared the best TC-ResNet ROC and the best DC-ResNet ROC, in Fig 12. Our model DC-ResNet is the one at lower space, which is slightly better than TC-ResNet.

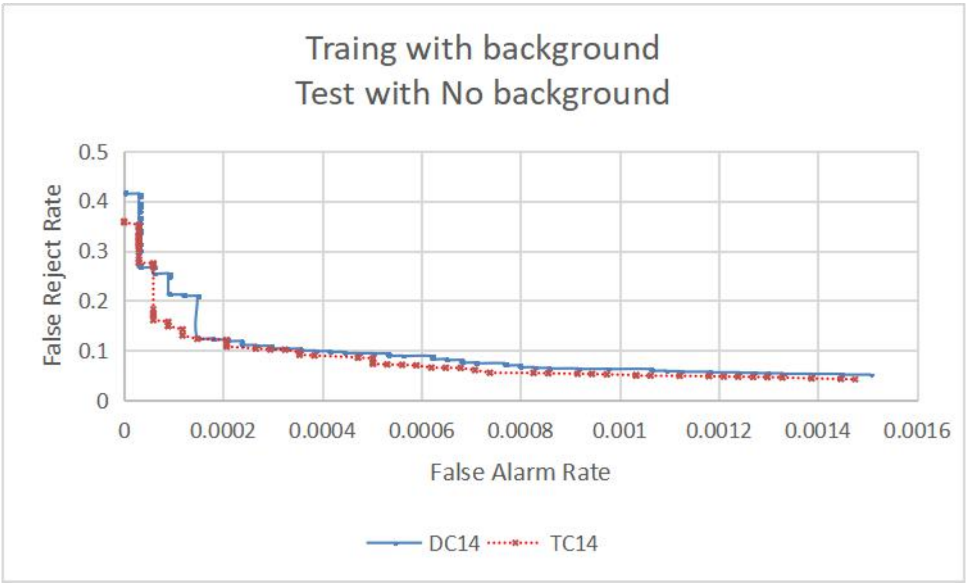


Fig 9. TC-ResNet ROC & DC-ResNet ROC tested on no augmented GSC

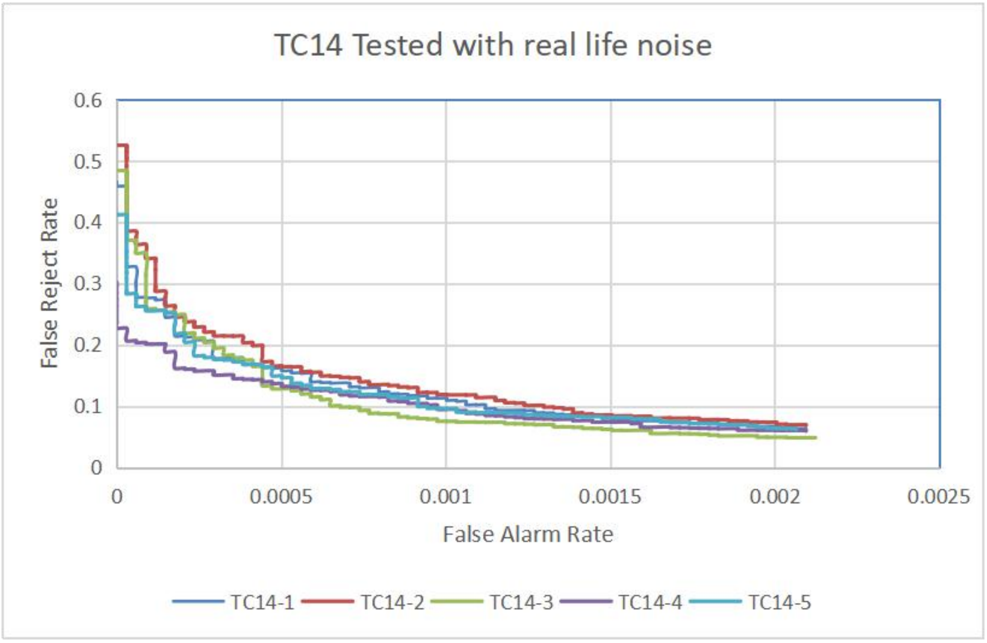


Fig 10. TC-ResNet ROC tested on augmented GSC 5 times

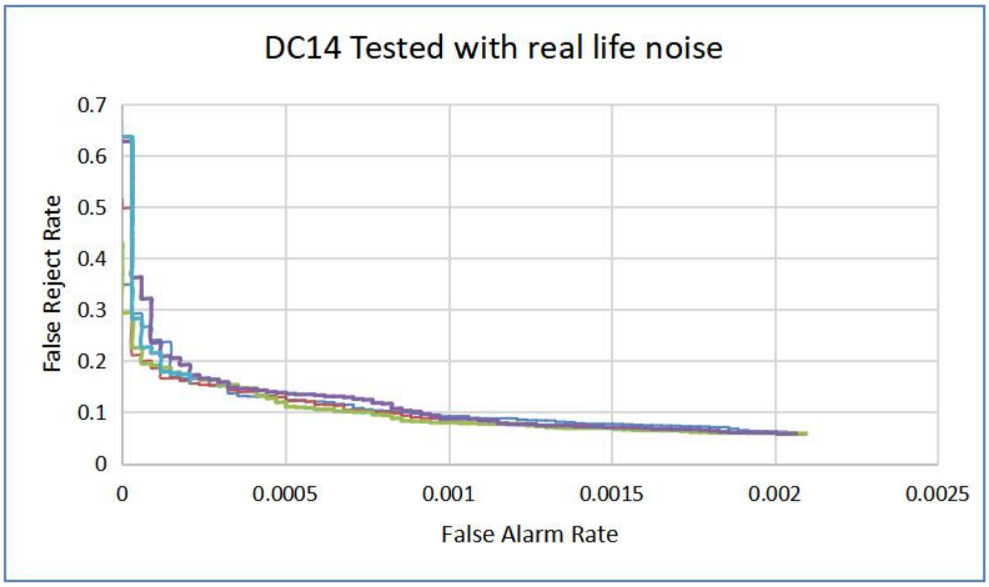


Fig 11.DC-ResNet ROC tested on augmented GSC 5 times

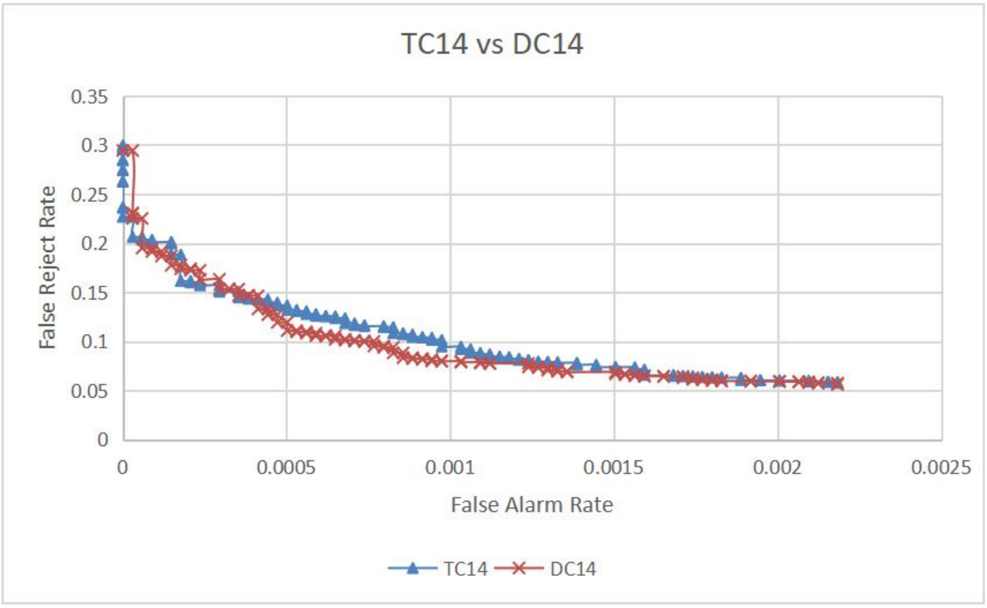


Fig 12. The comparison between the best TC-ResNet and the best DC-ResNet ROC tested on augmented GSC

Chapter 4

Conclusions

In this work, we compared various state-of-the-art KWS systems, including TC-ResNet, TDNN-SWSA, ResNet15, and DS-CNN. We also proposed a simple depthwise convolutional neural network, which greatly reduced the number of parameters, especially the computations, which is at most 300 times less than some of the existing models, and almost 3 to 5 times less than the state-of-the-art model.

1-D depthwise convolution utilized in this task is different from 2-D convolution and other types of 1-D convolution. The most preeminent advantage of it is that it drastically lowers FLOPs. Nevertheless, such small size with 12K parameters still cuts at least half of the FLOPs than other models. However, FLPOs are not directly proportional to the speed. When they are several, several hundreds or even several thousands times less, and at the same time other factors keep the same or even better, it means a lot. Without computational overhead, it meets the requirement of using on a real time. Finally, it achieved 17.5% increased speed compared to the fastest model, when it was tested on Samsung Galaxy S6 Edge.

For testing the robustness of the model, we adopted three different datasets: pure speech commands sets, silent noise augmented sets and the last one, training and tested on real life noise augmented sets.

In conclusion, this experiment can be a promising demonstration, implying a possibility of further implementations for the future.

Bibliography

[1] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. arXiv preprint arXiv:1707.01629, 2017.

[2] W. Xiong, L. Wu, F. Alleva, Jasha Droppo, X. Huang, and Andreas Stolcke. The microsoft 2017 conversational speech recognition system. CoRR, abs/1708.06073, 2017.

[3] Wang, Z., Li, X., & Zhou, J. (2017). Small-footprint keyword spotting using deep neural network and connectionist temporal classifier. arXiv preprintarXiv:1709.03665.

[4] Sainath, T. N., & Parada, C. (2015). Convolutional neural networks for small-footprint keyword spotting. In Sixteenth Annual Conference of the International Sp eech Communication Association.

[5] Zhang, Y., Suda, N., Lai, L., & Chandra, V. (2017). Hello edge: Keywordspotting on microcontrollers. arXiv preprint arXiv:1711.07128.

[6] Tang, R., & Lin, J. (2018, April). Deep residual learning for small-footprint key word spotting. In 2018 IEEE International Conference on Acoustics, Speech a n d Signal Processing (ICASSP) (pp. 5484-5488). IEEE.

[7] de Andrade, D. C., Leo, S., Viana, M. L. D. S., & Bernkopf, C. (2018). A neural attention model for speech command recognition. arXiv preprint arXiv:1808.0 8929.

[8] Arik, S. O., Kliegl, M., Child, R., Hestness, J., Gibiansky, A., Fougner, C.,... & Coates, A. (2017). Convolutional recurrent neural networks for small-footprint keyword spotting. arXiv preprint arXiv:1703.05390.

[9] Lee, L., Park, J., & Sung, W. (2019, December). Simple gated convnet for small footprint acoustic modeling. In 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU) (pp. 122-128).IEEE.

[10] Ian McGraw, Rohit Prabhavalkar, Raziel Alvarez, Montse Gonzalez Ar enas, Kanishka Rao, David Rybach, Ouais Alsharif, Ha¸sim Sak, Alexa nder Gruenstein, Françoise Beaufays, et al. Personalized speech reco gnition on mobile devices. In Acoustics, Speech and Signal Processi ng(ICASSP), 2016 IEEE International Conference on, pages 5955–595 9. IEEE, 2016.

[11] Jay G Wilpon, Lawrence R Rabiner, C-H Lee, and ER Goldman. Automatic recognition of keywords in unconstrained speech using hidden markovmodels. *IEEE Transactions on Acoustics, Speech, and Signal Processing*,38(11):1870–1878, 1990.

[12]Richard C Rose and Douglas B Paul. A hidden markov model based keyword recognition system. In Acoustics, Speech, and Signal Proce ssing, 1990. ICASSP-90., 1990 International Conference on, pages 12 9–132. IEEE, 1990.

[13] George Tucker, Minhua Wu, Ming Sun, Sankaran Panchapagesan, Ge ngshen Fu, and Shiv Vita ladevuni. Model compression applied to s mall-footprint keyword spotting. In *INTERSPEECH*, pages 1878–1882, 2016.

[14] R. Tang and J. Lin, “Deep residual learning for small-footprint keyword spotting,” in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 5484–5488.

[15] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha,“Temporal convolution for real-time keyword spotting on mobil e devices,” arXiv preprint arXiv:1904.03814, 2019.

[16] Y. Bai, J. Yi, J. Tao, Z. Wen, Z. Tian, C. Zhao, and C. Fan, “A time delay neural network with shared weight self-attention for smallfootprint keyword spotting,” Proc. Interspeech 2019, pp. 2190–2194, 2019.

[17] S. Myer and V. S. Tomar, “Efficient keyword spotting using time delay neural networks,” in Proc. Interspeech 2018, 2018, pp. 1264–1268. [Online].Availabl e: http://dx.doi.org/10.21437/Interspeech.2018-1979

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems 30, 2017, pp. 5998–6008.

[19] Y. Zhang, N. Suda, L. Lai, and V. Chandra, “Hello edge: Keyword spotting on microcontrollers,” arXiv preprint arXiv:1711.07128, 2017.

[20] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” arXiv preprint arXiv:1502.031 67, 2015.

[21] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. arXiv preprint arXiv:1707.01083, 2017.

[22] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Effificient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[23] P. Warden. (2017, August) Launching the speech commands dataset. [Online]. Available: https://ai.googleblog.com/2017/08/ launching-spe ech-commands-dataset.html

[24] S. Arik, H. Jun, and G. Diamos, "Fast spectrogram inversionusing multi-head convolutional neural networks," arXiv preprint arXiv:1808.06719, 2018.

Abstract

키워드 스팟팅(KWS)은 현재의 음성 기반 휴먼-컴퓨터 상호작용에서 중요한 역할을 하며 스마트 기기에서 널리 사용되고 있다. 신경망의 급속한 발달로 음성인식, 음성 합성, 화자인식 등 여러 음성 처리 분야에 걸친 어플리케이션에서 큰 성과를 거뒀다. 다양한 음성 처리 분야에서 강점을 보이고 있는 인공 신경망은 KWS를 위한 시스템에도 매력적인 선택이 되었다.

그러나 애플리케이션 환경은 스마트폰, 패드 및 일부 스마트 홈 기기를 포함한 소형 스마트 기기들이 대부분이기 때문에, 신경 네트워크 아키텍처들은 KWS 시스템을 설계할 때 이러한 스마트 기기의 제한된 메모리와 계산 용량을 고려해야 한다. 동시에 실시간, 사용자 친화적, 높은 정확도로 대응하려면 낮은 대기 시간을 유지할 수 있어야 한다. 또한 KWS는 다른 업무와 달라 상시 온라인 상태에서 이용자의 호출을 기다려야 하기 때문에 KWS 애플리케이션의 전력 예산도 크게 제한된다.
메인스트림 신경망 모델 중에는 과거 DNN, CNN, RNN, 그리고 서로의 조합이 주로 KWS에 사용되면서 최근에는 Attention 기반 모델도 점점 인기를 끌고 있다. 그 중에서도 CNN은 정확성과 견고성, 병렬처리가 뛰어나 KWS에서 널리 채택되고 있다.

본 연구에서는 효율적인 키워드 스팟팅을 지원하는 신경망 모델인 신플 콘볼루션 네트워크를 제시한다. 높은 정확도를 유지하기 위한 중간 과정으로 보다 컴팩트한 residual 네트워크와 노이즈 인식 훈련법을 주로 사용한다. ResNet은 좋은 성능을 얻기 위해 항상 수십만 개의 매개변수를 필요로 하기 때문에, 우리 모델에서는 한정된 자원을 가진 스마트 기기에 더 적합할 수 있도록 depthwise 콘볼루션 네트워크를 사용하여 파라미터 수를 줄이는 법을 제시한다. 마지막으로 실제 모바일 기기인 삼성 갤럭시 S6 엣지에서 제안된 모델의 실제 추론 시간(즉, 지연 시간)을 측정하였다. 온라인 상 공개된 Google 음성 명령 데이터 집합이

모델을 평가하는 데 사용되었다. 결과는 제시된 모델이 기존 모델보다 약 1/2 의 매개변수와 계산 횟수를 훨씬 적게 사용한다는 것을 보여주며거의 동일한 정확도로 속도가 17.5 % 빠르며 6.9ms에 도달했다. 훨씬 작은 메모리 소모로도 다른 최신 KWS 모델을 능가하는 96.59%의 높은 정확도를 유지하고 있다.

Keyword : Keyword Spotting (KWS), Convolutional Neural Network (CNN), Small Footprint

Student Number : 2018-24547

Master's Thesis

Simple Depthwise Convolutional Neural Network for Efficient Keyword Spotting

효율적인 키워드 인식을 위한 간략 콘볼루션 신경망

AUGUST 2020

BY
QIAN XUE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Simple Depthwise Convolutional Neural Network for Efficient Keyword Spotting

Professor Sung, Wonyong

Submitting a Master's Thesis of Engineering

2020.08

Graduate School of Engineering
Seoul National University
Department of Electrical and Computer Engineering

QIAN XUE

Confirming the master's thesis written by
QIAN XUE
2020.08

Chair	Namsoo Kim	(Seal) [illegible]
Vice Chair	Wonyong Sung	(Seal) [illegible]
Examiner	Kyomin Jung	(Seal) [illegible]

Abstract

Keyword spotting (KWS) plays an important role in the current speech-based human-computer interaction, and is widely used on smart devices. With the rapid development of neural networks, various applications in speech related fields such as speech recognition, speech synthesis and speaker recognition have achieved great performances. Neural networks have become attractive choices for KWS architectures because of their good performance in speech processing.

However, since the application environment is mostly in small smart devices including smart phones, tablets and smart home devices, neural network architectures must consider the limited memory and computation capacity of these smart devices when designing a KWS system . At the same time, the KWS system should be able to maintain low latency in order to respond in real time. In addition, KWS is different from other tasks, because it needs to be always online and waiting for the call from the users, therefore, the power budget of the KWS application is also greatly restricted.

Among the mainstream neural network models, FCDNN (fully connected deep neural network), CNN (convolutional neural network), RNN (recurrent neural network) and the combination of them are mainly used for KWS in the past. Recently, attention-based models have become more and more popular. Among them, CNN is widely adopted in KWS, because of its excellent accuracy, robustness, and parallel processing capacity. Parallel processing capacity is essential for low-power implementations.

In this work, we present a neural network model-Simple Depthwise Convolutional Network, which supports an efficient keyword spotting. We mainly focus on a more compact Residual Network, and apply noise injection as an intermediate process to maintain high accuracy. Typically, ResNet always requires several hundred thousands parameters to achieve good performance. In our model, we employ depthwise convolutional neural networks to decrease the number of parameters, so that it can be more suitable for smart devices with limited resources. Finally, our model is tested on a real mobile device Samsung Galaxy S6 Edge, reality in the real inference time (that is, latency) of about 6.9ms, which is 17.5% faster than the state-of-the-art model TC-ResNet. The publicly available Google Speech Commands dataset is used to evaluate the models. The results show that we only use about one half of the parameters and at most 300 times fewer number of computations than the original base model, meanwhile, much smaller memory footprint yet maintain the 96.59% comparable high accuracy which outperforms the other state-of-the-art KWS models.

Keyword : Keyword Spotting (KWS), Convolutional Neural Network (CNN), Small Footprint

Student Number : 2018-24547

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Keyword Spotting System (KWS)

The rapid development of neural networks has made artificial intelligence possible, and has achieved good results in processing speech and images [1,2]. Neural network(NN) based KWS has achieved great popularity in the recent years [3,4,5,6,7,8]. The accuracy of machine recognition basically exceeds that of human recognition. As the most basic and direct way to interact with machines, speech plays an extremely critical role in artificial intelligence systems. In recent years, it has been used by major technology companies in the world for daily interaction on smart devices or smart homes. Speech recognition is mostly performed in the servers of service providers after the user's voice is transmitted. However, this server-based speech recognition has drawn attention regarding security and privacy. This is because the user's voice has been transmitted to the server, making it vulnerable to external attacks, and possibly leaking personal information to the outside [9]. In order to alleviate these concerns, on-device speech applications are needed. Before performing speech recognition, the device needs to be woken up and detect several predefined keywords. These predefined short contents consist of several characters called keywords. This process of detecting keywords by the device is called keyword spotting.

Keyword spotting is the first step of human-computer interaction based on speech. Thus, it is very important to detect the keywords very accurately, so that subsequent speech recognition can be activated. Then it is possible to perform interactive task operations. The process of keyword spotting is actually divided into these following steps: First, the device needs to stay online at all times, waiting for the user to give a call. When the user speaks out the keyword, the online device can receive audio signals in real time to quickly detect if it is recognized as a keyword, the device will wake up from the standby state to enter the interactive preparation state [10].

As introduced above, simply speaking, in some terms, keyword spotting is actually a simplified version of speech recognition. However, there is no decoding part like a language model, and the final task is to complete a classification task. In a typical process of keyword spotting using a neural network model, the entire system is roughly divided into two processes, as shown in Fig 1. The first one is the acoustic feature extractions, and the other one is the classification process based on the neural network model.

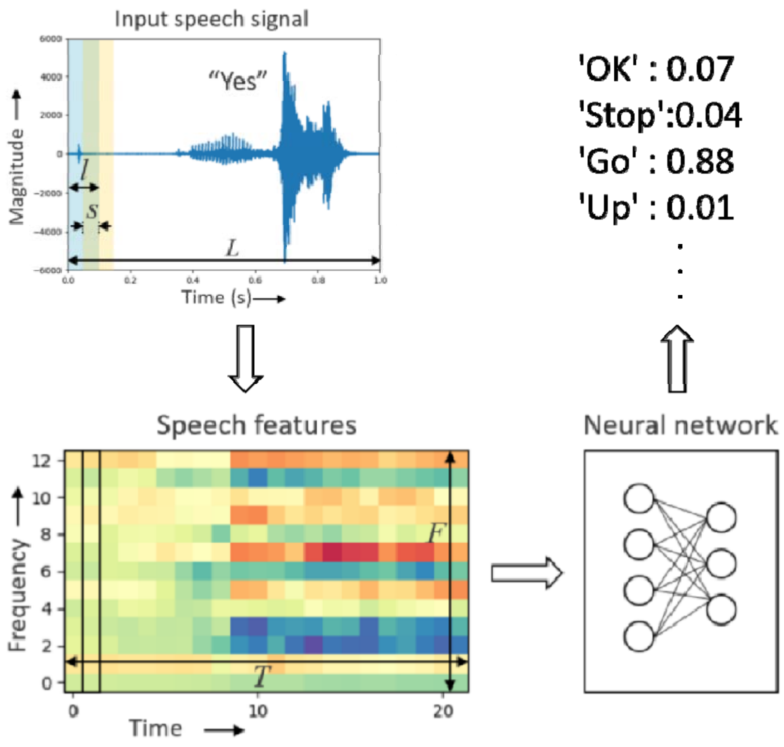


Fig 1. End to End keyword spotting system

The first step is the feature extractions, which is actually the same as that in speech recognition. The arriving speech signal is passed to the feature extraction module. If the speech signal length is L, a window function of length w is added, and s is the stride size. T frames are always obtained. Each frame extracts F-dimensional speech features through Mel-Frequency Cepstral Coefficients (MFCC) or Mel-Frequency Cepstral Banks (MFFB). Then, the entire input speech signal is converted into $T\text{x}F$ feature graph. In the second step, the two-dimensional feature matrix obtained above is transmitted to the classifier module. Finally the probability of the output category is obtained through the neural network model.

In addition to the end-to-end neural network-based KWS system described above, the traditional method also uses the keyword/filler hidden Markov model

(HMM) for recognition [11， 12], as shown in Fig 2. The key to this type of system is the decoding module on the lower side of Fig. 2. It is similar to the decoder in the HMM based speech recognizer. It also utilizes the Viterbi algorithm to obtain the optimal path, but it is similar to LVCSR (large-scale vocabulary continuous speech recognition). The difference from the speech recognition system is the specific construction of the decoding network. The decoding network in speech recognition contains all the words in the dictionary, while the wake-up decoding network contains the keyword and filler words on the upper side of Fig.2. The words excluding the keywords are all included in the filler path, and not every word will have a corresponding path. Such a network will be much smaller than a typical speech recognition network. When decoding keywords in a targeted manner, there are fewer optional paths, allowing the improvement of decoding speed. All of the other decoded candidates follow the same method to complete the overall framework. Although this method has achieved a reasonable performance in accuracy, it is still difficult to train, and it also requires a lot of computation process. Other technologies, such as RNN, are significantly better than HMM-based KWS in terms of accuracy [13]. Since RNNs have to wait for the previous steps, the structure demands a large delay, which is not ideal for KWS requiring a real-time response. Therefore, this article implements the variant network of CNN to perform KWS tasks.

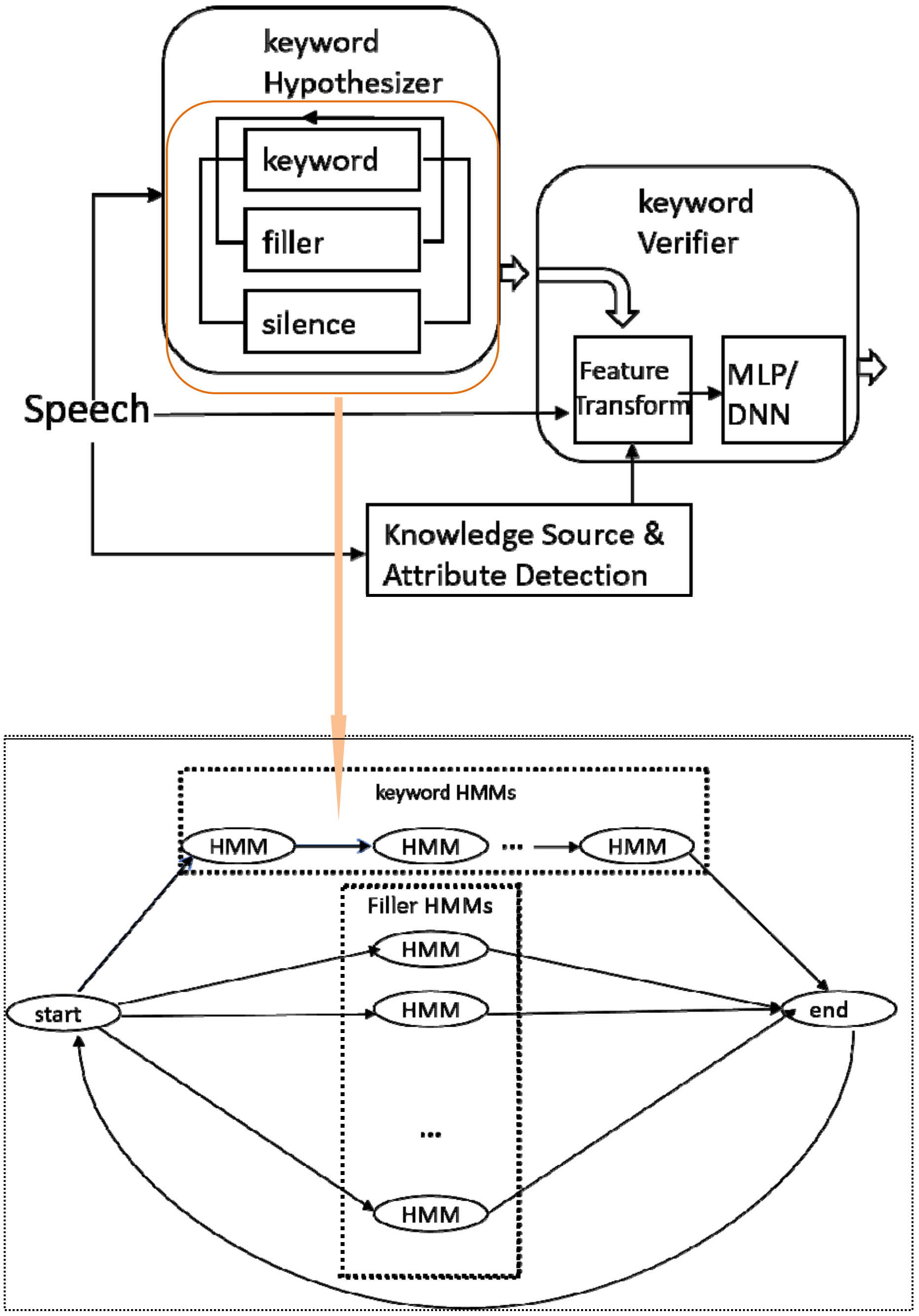


Fig 2. The Topology of HMM based keyword spotting system

1.2 Challenges in Keyword Spotting

As introduced in chapter 1.1, keyword spotting is usually considered as the first step of the human-machine interaction, mostly used on smart devices. There are basically four metrics for KWS.

The recall rate recalls to the number of times that it was correctly awakened as a percentage of the total number of times the keyword was detected. This value is better when it is larger.

The false alarm rate refers to the probability of keywords that should not be detected. A better the performance can be achieved with a lower value.

The real time factor is also one of the four metrics for KWS, which represents the response speed of the equipment.

Lastly, the power metrics is another metric for KWS. It is essential for portable devices.

Regarding the four metrics described above, there were some notable challenges. That is the trade-off between high accuracy and low power consumption, or high accuracy and low latency. In this thesis, we not only focus on the accuracy, but also pay attention on the latency. Usually power consumption is mainly affected by the capacity of hardware of devices and architecture of models we designed, which requiring us to deeply compress our model, so that less parameters and computations are demanded. However at the same time, this model should be able to maintain a high accuracy and faster speed comparable to that of the state-of-the-art models.

1.3 Neural Network Architecture for Small-Footprint KWS

There are some neural network architectures that are suitable for the on-device small footprint KWS. Among the mainstream neural network architectures, convolutional neural network (CNN) based models and ResNet based models show fairly great performances, especially the models outperform in accuracy [14,15] and showing low latency [15]. However, all of these ResNet models in previous studies consume quantities of parameters, doing lots of computations as the cost of pursuing high accuracy. As a result, the response speed is slowed by a considerable amount. The trade-off of these metrics is crucial for KWS, since KWS is commonly used on resource restrained devices. In this section, several latest researches will be explored including the architecture using self-attention which gained popularity in speech recognition, time delay neural network, temporal convolution combined with ResNet, and lastly with depthwise convolution.

1.3.1 TDNN-SWSA

This network is the time delay network with shared weight self-attention (TDNN-SWSA) [16]. TDNN is known as a classic network architecture and has achieved great success in recent speech recognition tasks [17]. In this study, they used TDNN here to capture local features, and shorten the length of the input before feeding it into the self-attention module. In addition, three matrices in the self-attention module [18] share the same matrix and are projected into the same single space. In this way, the number of parameters diminished sharply.

The schematic of the TDNN based subsampling is as shown in Fig 3. The length of the input is shortened to $(Tin - w + 1)/k$, where w is the length of the TDNN window Eq.(1) and Eq.(2) show the differences between the two different attention methods.

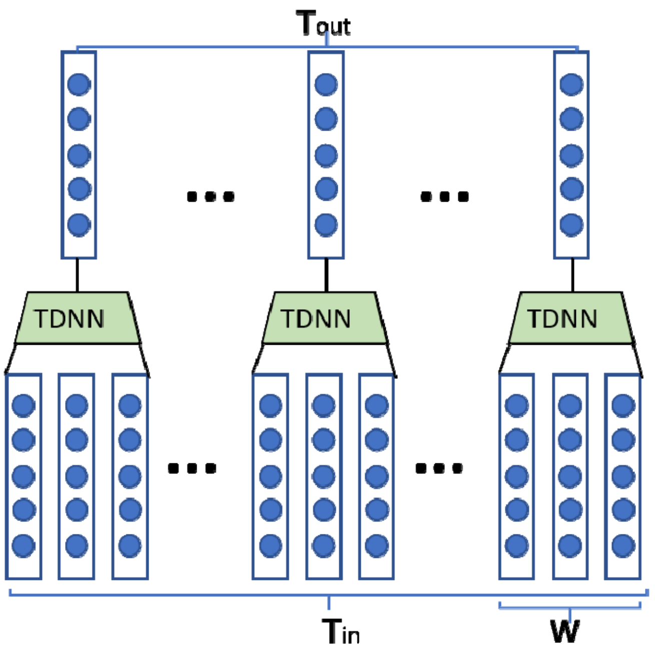


Fig.3 The schematic of TDNN-SWSA

The innovation of TDNN is the usage of the self-attention to share weights. In order to reduce the total number of parameters.

The traditional way of self attention is as follows:

$$\begin{aligned}&\mathrm{Attend}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_k}})\mathbf{V},\\ &\text{where } \mathbf{Q} = \mathbf{U}\mathbf{W}_q,\ \mathbf{K} = \mathbf{U}\mathbf{W}_k,\ \mathbf{V} = \mathbf{U}\mathbf{W}_v.\end{aligned} \tag{1}$$

While the SWSA is represented as:

$$\begin{aligned}&\mathrm{Attend}(\mathbf{V}) = \mathrm{softmax}(\frac{\mathbf{V}\mathbf{V}^T}{\sqrt{D}})\mathbf{V},\\ &\text{where } \mathbf{V} = \mathbf{U}\mathbf{W}.\end{aligned} \tag{2}$$

A shared weight matrix replaces three different matrices which correspond to queries, keys and values. In this way, the number of parameters are reduced sharply into 12k, only 1/20 of ResNet15, although there is some accuracy sacrifice.

1.3.2 TC-ResNet

TC-ResNet refers to Temporal Convolutional - Residual Networks [15]. Convolutional neural network (CNN) based KWS tasks have shown outstanding accuracy. This model applied temporal convolution, i.e. 1D convolution along the temporal dimension and took MFCC features as the input, as shown in Fig 4. Compared with 2D convolution, the output feature map size of temporal convolution is much smaller, which contributes to the drastic reduction of the computational burden in the next layers and its fast implementation. Another CNN architecture adopted is ResNet, specifically, ResNet8 and ResNet14. They changed the kernel size into 3x1 and 9x1, and expanded some channels in some of the model experiments.

By using temporal convolution, the burden of computation was lessened and the kernel looked at the whole range of frequency to improve the performance. TC-ResNet achieved the best accuracy in the model TC-ResNet14-1.5 with 96.6%. However the downsides of using large size of ResNet are consuming hundreds of thousands of parameters and requiring lots of computations, which leads to a large model size and increase in the inference time.

1.3.3 DS-CNN

DS-CNN refers to the depthwise separable convolutional neural network (DS-CNN) [19]. Depthwise separable CNN here is based on the implementation of stacking n many pure depthwise separableconvolutioins. A DS-CNN is composed of one depthwise convolution decomposes 3-D convolutions into 2-D convolutions, and one pointwise convolution which follows the depthwise convolution. Each convulotion was followed by a batch normalization [20] and the (Relu) activation function. N many DS-CNNs were stacked together to build up a DS-CNN model. The usage of depthwise and pointwise convolution made wider and deeper systems possible, even in the resource-constrained devices. Furthermore, 8bit quantization was performed here to compress the model size. We can take the advantage of efficiency in number of parameters, operations and model size.

1.4 Simple Depthwise Convolutional Neural Network for Efficient KWS

Through the previous study, we recommend a simple depthwise convolutional neural network for footprint KWS. Simple depthwise separable convolutional neural network is the simplest form of depthwise convolution, combined with residual neural network and we utilized this architecture in training our model.

Simple Depthwise Convolutional Network consists of 1-D depthwise convolution part and a ResNet part. ResNet-based KWS systems showed great performance. In order to keep a high accuracy, we used ResNet. The issue in utilizing ResNet that required solving was that it consumed too many parameters. To resolve this, we applied another way of implementing the 1-D depthwise separable convolutional neural network. Depthwise convolution is advantageous since it requires less parameters, making it different from the traditional convolutions. However, the lack of pointwise convolution makes it difficultto expand the feature map, resulting in an accuracy not as good as before. In order to make it learn neighboring channels, we chose simple depthwise convolution by looking through K channels (in our model, it means K features) and using T length as one input. In addition, we applied noise injection into weights, which improved accuracy at some level, since training with noise injection was helpful in finding a wide range of local minima in loss surface, and also avoid overfitting. Lastly, we trained on three different KWS datasets to make our system more robust.

Through this method, our simple depthwise convolutional neural network was able to maintain the best accuracy, while occupying much less parameters, smaller model size and faster speed .

1.5 Outline of the Thesis

More details about our recommended model will be introduced in the following pages. And the rest of this dissertation are organized as follows. In Chapter 2, simple depthwise convolution is illustrated, including the structure of simple depthwise convolution, its contribution and the results of some experiments when choosing the basic settings. Chapter 3 focuses on simple depthwise convolution with noise injection, the experiments on three different datasets, and the comparisons with the state-of-the-art models, especially on the accuracy, speed, the number of parameters and the computations. The last chapter gives a conclusion of the entire thesis.

Chapter 2

Simple Depthwise Convolutional Neural Network

This chapter includes four sections, and the first three are the introduction of basic models : the traditional depthwise separableconvolution, simple expanded depthwise convolution, and recommended network combined. The last part is composed of experiments and results of this model compared with other various networks .

2.1 Depthwise ConvNet

Depthwise ConvNet is the variant of traditional convolutional network, which is popular and has been employed in various fields such as in machine translation, computer vision and speech recognition [21,22].

Depthwise convolution as shown in Fig.5 is a part of depthwise separable convolution which was inspired from MobileNet [22]. The other part is pointwise convolution, as shown in Fig.6. Unlike conventional convolution operations, one convolution kernel of Depthwise Convolution single channel kernel is responsible for only one input channel. If the number of input channels is N, there must be N many channels or multiple of N many channels. Besides, each kernel is a single-channel. Every time, the single-channel kernel convolutes with the corresponding single channel of the input. Therefore, a N-channel input is processed to generate N feature maps (if there is same padding, the size is the same as the input layer). While in the conventional convolution, each convolution kernel operates simultaneously with every channel of the input.

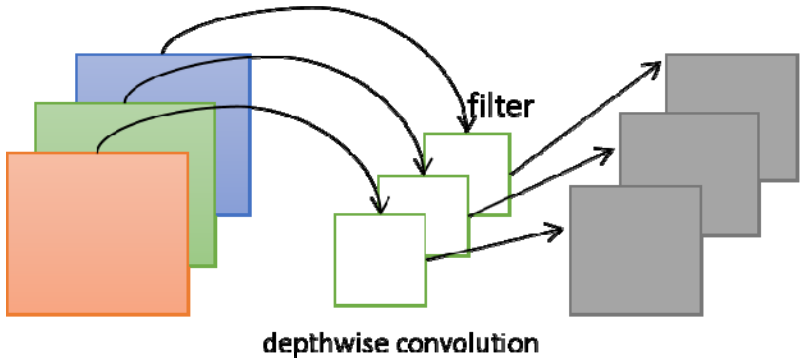


Fig.4 Depthwise convolution in depthwise separableconvolution

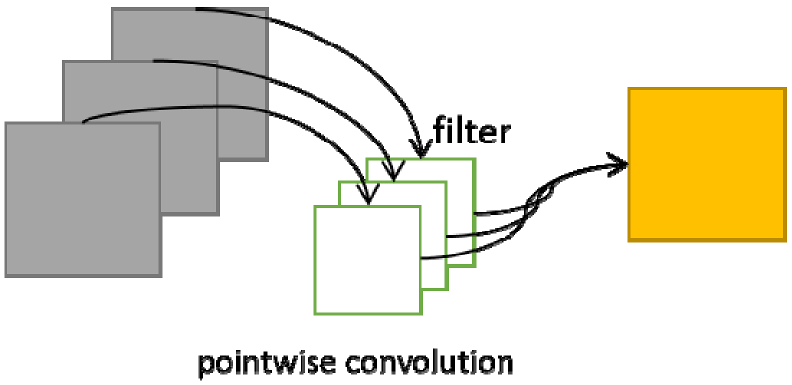


Fig.5 Pointwise convolution in depthwise separableconvolution

The output feature map of depthwise convolution cannot be extended. It is the input of pointwise convolution, and pointwise convolution operates the same convolution as depthwise convolution does, after which combining the output feature maps together as one new feature map. In this way, the model is able to effectively use feature information of different channels at the same spatial position.

2.2 Simple Depthwise ConvNet

First is the 1-D temporal convolutional neural network, rather than the 2-D convolutin in the left side of Fig.6. TC-ResNet chose to do 1-D convolution along the time axis in the right side of Fig.6 . If the kernel size is $3\text{x}3\text{x}1\text{x}C$ for 2-D convolution, $3\text{x}1\text{x}D\text{x}C'$ for 1-D temporal convolution, when they keep the same number of parameters, the MACs of 2-D convolution is is $3\text{x}C/\mathrm{f}$ (C is 160, f is 40, here) times of 1-D temporal convolution. Thereby, 1-D convolution needs much less parameters and computations than 2-D convolution.

Simple Depthwise Convolutional Network is illustrated as the 1-D depthwise convolution as shown in the right side of Fig.7, which is different from the conventional depthwise convolution part of depthwise separableconvolution, and is slightly different from the 1-D temporal convolution mentioned above. 1-D depthwise convolution also convolutes along the time axis. Although the height of the kernel is not 1 but k, it is still 1-D convolution, because we set the kernel size to $K\text{x}T\text{x}1$. The reason why we stacked K here is that pure depthwise convolution cannot obtain the information from other channels, which may result in bad performance. Therefore, after padding, there are D input channels (from the first channel to Dth channel) on the first layer, D channels (from the second channel to (D+1)th channel) on the second layer and so on until k layers. K neighboring channels were stacked together, then each channel of the new stacked input was made sure it contained k neighboring channels and was convoluted by one kernel at the same time. If the kernel size is $3\text{x}1\text{x}D\text{x}C$ for the 1-D temporal convolution, $K\text{x}T\text{x}1\text{x}D$ for the 1-D depthwise convolution, while keeping the same number of parameters, the MACs of 1-D temporal convolution is $C'/9$ (assume k=3 t=9

C’=12, but all of the kernel size in TC-ResNet are $9\mathrm{x}1$, therefore the value of this is about 4) times of 1-D depthwise convolution.

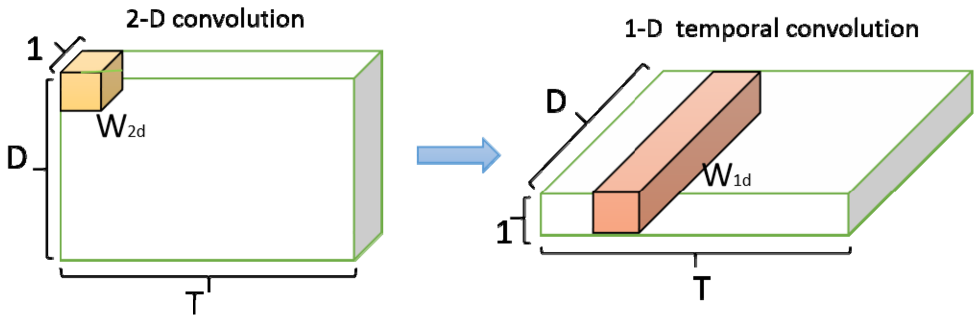


Fig.6 Compare traditional 2-D convolution with
1-D temporal convolution.

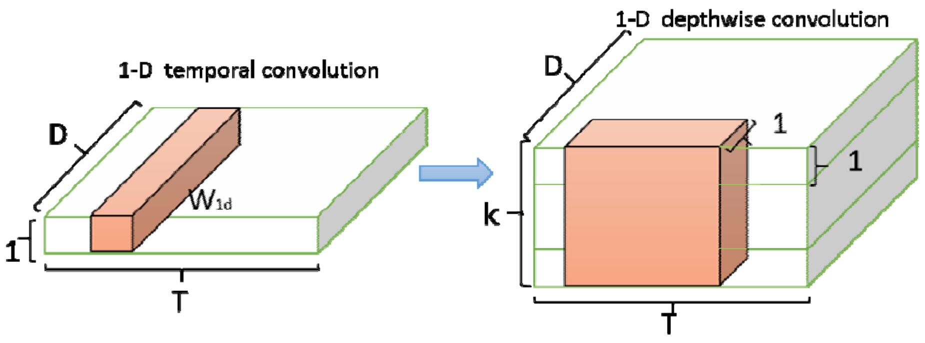


Fig.7 Compare 1-D temporal convolution with
1-D depthwise convolution

2.3 Residual Simple Depthwise ConvNet

As mentioned-above in section 1.3.3, the model merely used depthwise convolution neural network (DS-CNN), and performed better than the mainstream recurrent neural networks, including LSTM and GRU. However, the accuracy was not that high enough compared to state-of-the-art model. The newly proposed model, TC-ResNet in section 1.3.2, achieved the highest accuracy with 96.6% using ResNet14 with multiplier 1.5 (multiplier is used to expend channels in each block). It proved that ResNet architecture improves the accuracy, the only problem was that ResNet consumes a lot of parameters and memory. Therefore, we adopted the advantages of depth-wise convolution and residual network. The whole architecture of our model is shown in Fig 8. The audio signal after the preprocessing produced a speech feature representation, which is the input of the neural network. The first layer is a 1-D depthwise convolution followed by 3(for DC-ResNet8) or 6(for DC-ResNet14) blocks in which there are two 1-D depthwise convolutions connected with one residual connection. If the channels between the previous layer and the following layer are different, the stride of the first layer inside a block would be set to 2 and stride-2 1x1 kernel sized normal convolution module would be added into residual connection.

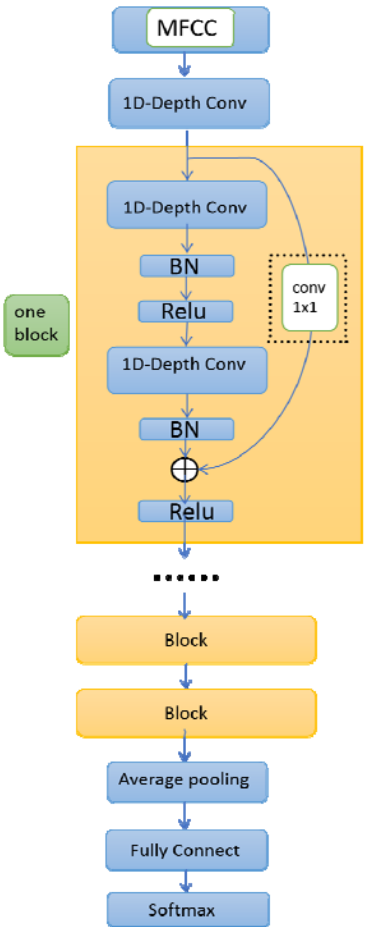


Fig.8 The Whole Architecture of DC-ResNet

2.4 Experiments and Results

Dataset we used in this task is Google Speech Command Dataset(GSC) [23], which is specially designed for device controlling tasks. There are about 65000 recordings of 30 words, each of them being a second long. Among them, 10 of them are keywords and the rest are fillers, and all are labeled as '_unknown_'. In this experiment, there is one more class, which is the background, necessitating a total of 11 to 12 classifications. This dataset is used for training, validation and test, and the proportion of these three parts are 8:1:1 respectively.

The preprocessing of the data followed the previous study. Each one-second raw audio is decomposed into a sequence of frames by a window with length of 30 ms and shifted by 10ms stride at each time for feature extraction. We utilized 40 dimension MFCC feature representations for each frame, and stack them over time-axis.

First of all, we found the best base channels, without multiplier, according to the settings in the previous study. We set k=5 t=9 to find the best base model. N_channels = [40,50,80,110] was chosen as the base model as shown in Table1.

Table 1 Finding the base channel list.

Channel	Acc [%]	Params
40,36,36,48,48,72,72.	94.83	90468
40,48,48,72,72,108,108.	95.93	138372
40,50,50,80,80,110,110.	96.05	148420
40,60,60,80,80,120,120.	95.92	162620

Secondly, we used the multiplier to extend the number of channels. The experiments indicated that setting k as 1.5 improves accuracy the most, with relatively fewer parameters.

Table 2 Various multiplier K attempts.

Multiplier K	Acc [%]	Params
expendnewk= 1.4.	96.12	228220
expendnewk= 1.5.	96.33	276750
expendnewk= 1.6.	96.27	302768
expendnewk= 2.0.	96.17	375880

Thirdly, the accuracy and the number of parameters were computed with different values of K and T. Table 3 displays the results with different values of K and T. Nearly 10 cases with k = 3,5,7 and t = 7,9,11,13 were tested. These results display the best ones chosen in each K, which used the values of K=3 and T=9. It had an accuracy of 96.49%, while TC-ResNet had 96.6%(only 0.11% dropout). However, merely half the parameters were required.

Table 3 Results of different 1-D depthwise convolution kernel size.

Model	Acc [%]	Params
(k=3, t=9)	96.49	155450
(k=3, t=11)	95.91	196094
(k=5, t=9)	96.44	225744
(k=5, t=11)	96.33	276750
(k=7, t=11)	96.35	304906

Chapter 3

Robustness of Efficient Keyword Spotting

3.1 Weight Noise Injection

KWS is considered as a neural network based classification task, and we used the cross entropy for Keyword Spotting training. As proved in [16, 10], appropriately injecting noise into weights in training stage helps toavoid over fitting and being easily stuck in narrow local minima in the losssurface. Thereby, it helps to achieve a higher accuracy more or less. We were able to achieve the improvement of performance in our experiment. There are several different ways to operate weight injections such as Gaussian noise, uniform noise and smooth out. In this experiment we chose the uniform noise injection.

Traditional stochastic gradient descent method to update parameters:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta/B \sum_{i=1}^{B} \nabla \mathbf{L}_i(\mathbf{w}_t)$$

After the noise was injected :

$$\hat{\mathbf{w}}_{t+} = \mathbf{w}_t + \alpha \mathbf{n}_t$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta/B \sum_{i=1}^{B} \nabla \mathbf{L}_i(\hat{\mathbf{w}}_t +)$$

Where B is the batch size, and ‘η’ is the learning rate. In this noise injection, ‘α’ is the scale factor of noise injected into weights, and ‘n’ is exactly the noise vector which is randomly produced via uniformly distribution. The value of this noise was determined by the standard deviation of all the parameter weights. Through these efforts, our simple depthwise convolutional neural network was able to maintains a comparable accuracy, much less parameters, smaller model size, and faster speed.

In this part, I utilized the same dataset as the GSC mentioned above. Noise injection did improve the accuracy by 0.1% in our task, as shown in Table 4. Through a lot of experiments, we decided to use 0.03 as the scale factor ‘α’ in Table 5.

Table 4. Results of the models trained with the single weight injection. SW denotes the single weight injection, Acc denotes the accuracy.

Model	**Acc [%]**	
	dev	test
DC-ResNet14. (*k*=3, *t*=9)	-	96.49
DC-ResNet14. (*k*=3, *t*=9) + SW.	-	96.59
DC-ResNet14. (*k*=5, *t*=9)	-	96.44
DC-ResNet14. (*k*=5, *t*=9) + SW.	-	96.48

Table 5. Results of different noise scales. n denotes the noise scale, k,t represent the height and the weight of kernel size.

Model (k=3,t=9)	**Acc [%]**	
	dev	test
DC-ResNet14. (n=0.0)	-	96.49
DC-ResNet14. (n=0.03)	-	96.59
DC-ResNet14. (n=0.05)	-	96.33
DC-ResNet14. (n=0.3)	-	96.34

3.2 Experiments on Two Different GSCs

We trained and evaluated our model on an English dataset Google Speech Commands (GSC) [17], which is specially designed for controlling tasks on device. Three are two versions of this, and we experimented on each group. There are two versions of this dataset, and we performed two groups of experiments on these.

10 keywords of these two versions are the same: ‘yes’, ‘no’, ‘left’, ‘right’, ‘on’, ‘off’, ‘go’, ‘stop’, ‘up’, ‘down’.

3.2.1. Standard GSC

This version of GSC dataset is composed of 64752 recordings from various people for total 30 words, including 10 keywords, and the rest 20 words noted as fillers. Each recording is a one second long speech signal, and composed of only one word.Therefore, the classification classes in this experiment are 11 or10 keywords, all the other 20 fillers are collectively referred to as ”unknown”. According to the .txt files of validation and testing, the whole dataset is divided into three parts: 51088 recordings for training, 6798 recordings are used as validation set, and the rest 6833 recordings for evaluation.

3.2.2. Augmented GSC

This version of GSC dataset is composed of 64727 recording, and each of them is one second long and composed of 30 words. Experiments performed in section 2.4 and section 3.1 used this dataset. Furthermore, following Google's implementation [17], data was augmented with background noise. In addition to the 11 classes above, the 12th class, 'silence' was also produced. According to the .txt files of SHA-1 hashed name, the dataset is split into training, validation, and test datasets with 22246, 3093 and 3081 files respectively.

3.2.3 Experiments and Results

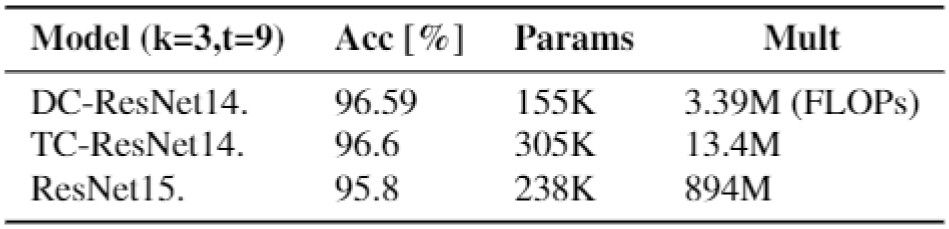
Experimental settings followed the setup in the previous work [15]. The window size and the stride was set up as 30ms and 10ms, respectively. Finally, the 40-dimension MFCC features were extracted. When training models, we set the dropout probability was 0.5, weight decay was 0.001, and applied 0.03 scaled weigh injection to optimize the loss. Learning rate started from 0.1 and dropped by 1/10 every 10k iterations, each models trained for 30k iterations in total.

According to the different datasets, the experimental results were separated into two groups. First group was the comparisons among DC-ResNet, TC-ResNet, ResNet15and DS-CNN, using the augmented GSC dataset. Second group was the comparisons among DC-ResNet, TDNN-SWSA, ResNet15 experimented on standard GSC dataset.

Table 6 illustrates several different architectures. We found that our model (DC-ResNet14) keeps a comparable accuracy with the state-of-the-art model, consuming the least parameters, and the merely 3.39M FLOPs [24] instead of Multipliers displayed by the other two models. Table 7 focuses on the latency between two models. We tested the real inference time using Tensorflow Lite Android benchmark tool. The value 8.4 is different from 5.7 in paper because we tested the inference time on other mobile device, the Samsung Galaxy S6 Edge. In Table 7, DC-ResNet responds 17.5% faster than the latest model, with the best performances so far, and maintains a high accuracy, and at the same time, our model only employed 1/2 of the parameters.

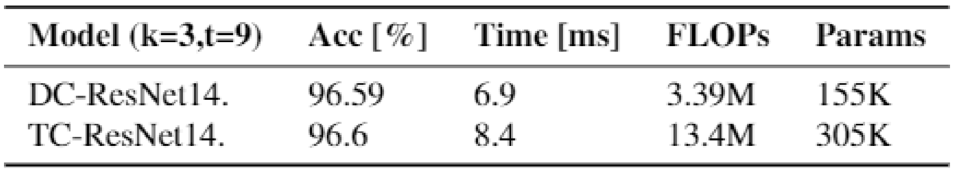
Table 8 displays diverse architectures tested on Standard GSC. Compared to the ResNet15, the accuracy of DC-ResNet dropped slightly. However when we traded off these three metrics illustrated on the table together, it was still fairly acceptable. Particularly, the FLOPs. FLOPs of DC-ResNet is half of TC-ResNet, 1/6000 of ResNet15.

Table 6. Results of various models on augmented GSC



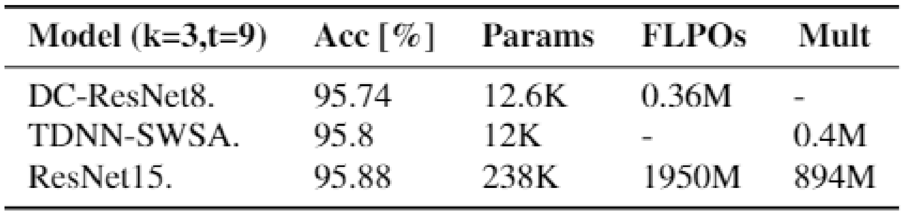
Model (k=3,t=9)	Acc [%]	Params	Mult
DC-ResNet14.	96.59	155K	3.39M (FLOPs)
TC-ResNet14.	96.6	305K	13.4M
ResNet15.	95.8	238K	894M

Table 7. The performance results for DC-ResNet and TC-ResNet



Model (k=3,t=9)	Acc [%]	Time [ms]	FLOPs	Params
DC-ResNet14.	96.59	6.9	3.39M	155K
TC-ResNet14.	96.6	8.4	13.4M	305K

Table 8. Results of various models on a standard GSC



Model (k=3,t=9)	Acc [%]	Params	FLPOs	Mult
DC-ResNet8.	95.74	12.6K	0.36M	-
TDNN-SWSA.	95.8	12K	-	0.4M
ResNet15.	95.88	238K	1950M	894M

3.3 FRR and FAR in a Third Dataset

3.3.1 FRR and FAR

In KWS system, besides the four main metrics introduced above, receiver operating characteristic (ROC) curve, and area under the curve (AUC) are also crucial. In KWS, x-axis of ROC is the false alarm rate (far), y-axis is the false reject rate (frr), AUC is the area under the ROC curve.

False alarm rate is the same as the false positive rate (fpr). False reject rate is also same as the false negative rate (fnr). Summing the false negative rate up with true positive rate is 1, where true positive rate is the recall rate. i.e.:

False alarm rate = False positive rate = FP / N = FP / (FP + TN)

False reject rate = False negative rate = FN / P = FN / (FN + TP)

= 1- True positive rate = 1- recall rate

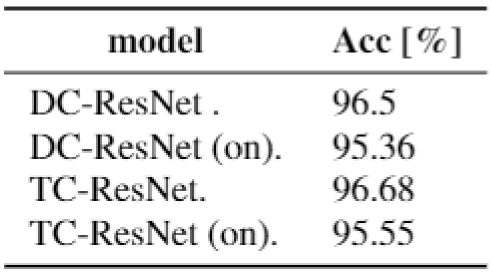
(‘T(/F)P(/N)’ is the number of True(/False) Positive (/Negative) samples)

3.3.2 Third GSC

This third GSC was modified from the augmented GSC in 3.2.2. In the previous augmented GSC, there are only 6 background noises in the dataset, such as pink noise, white noise, do the dishes, etc... These are all silent noise. In the real world, there is much more audio speech noise. Therefore, we added 6 more real life background noise files, including the TED speech, CNN news, White house briefing, etc.... Most of them are conversation-s across different ages, genders, different accents, and tried our best to cover variant speech features.

Testing on augmented data challenging. As shown in Table 9, the accuracy of both models decreased by around 1.14%.

Table 9. The test accuracy (on)/(not on) third-augmented real life data. (“On” means trained and tested on third GSC. Otherwise, only trained on third GSC, tested on test set of augmented GSC in 3.2.2)



model	Acc [%]
DC-ResNet .	96.5
DC-ResNet (on).	95.36
TC-ResNet.	96.68
TC-ResNet (on).	95.55

3.3.3 Experiments and Results

In the first experiment, the model was trained on third - augmented real life dataset, but tested on normal augmented dataset in 3.2.2. We attained one tested TC-ResNet ROC, and one tested DC-ResNet ROC, as plotted in Fig 9, respectively. As shown in Fig 10 and Fig 11, we trained and tested both models on third - augmented real life dataset for 5 times. From the figures below, we can find that they are fairly similar in AUC, AUC of both the two models are around 0.998~0.999. Although more background noise was injected, the trend of the scatter distribution did not have a dramatic change, as shown in Fig11, which proves robustness. At last, we compared the best TC-ResNet ROC and the best DC-ResNet ROC, in Fig 12. Our model DC-ResNet is the one at lower space, which is slightly better than TC-ResNet.

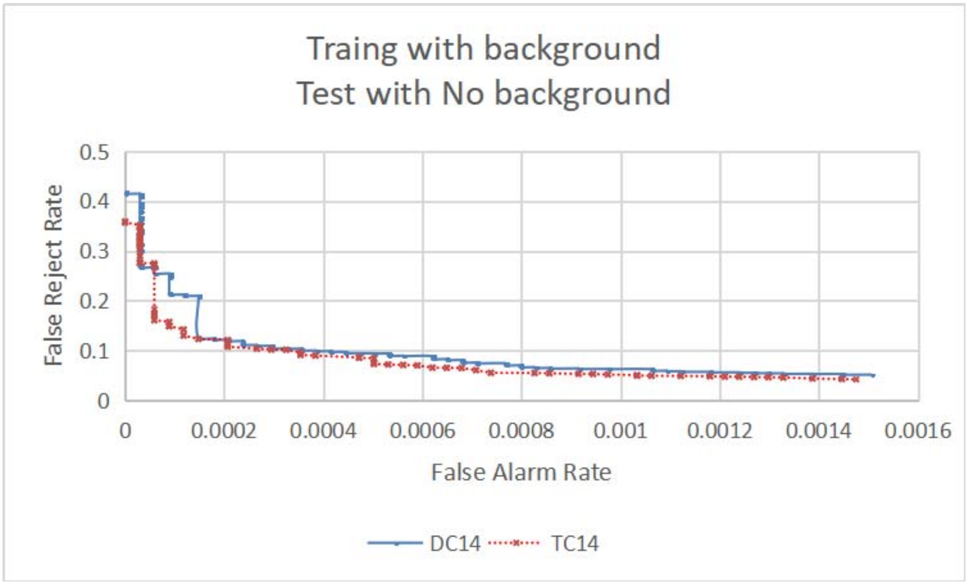


Fig 9. TC-ResNet ROC & DC-ResNet ROC tested on no augmented GSC

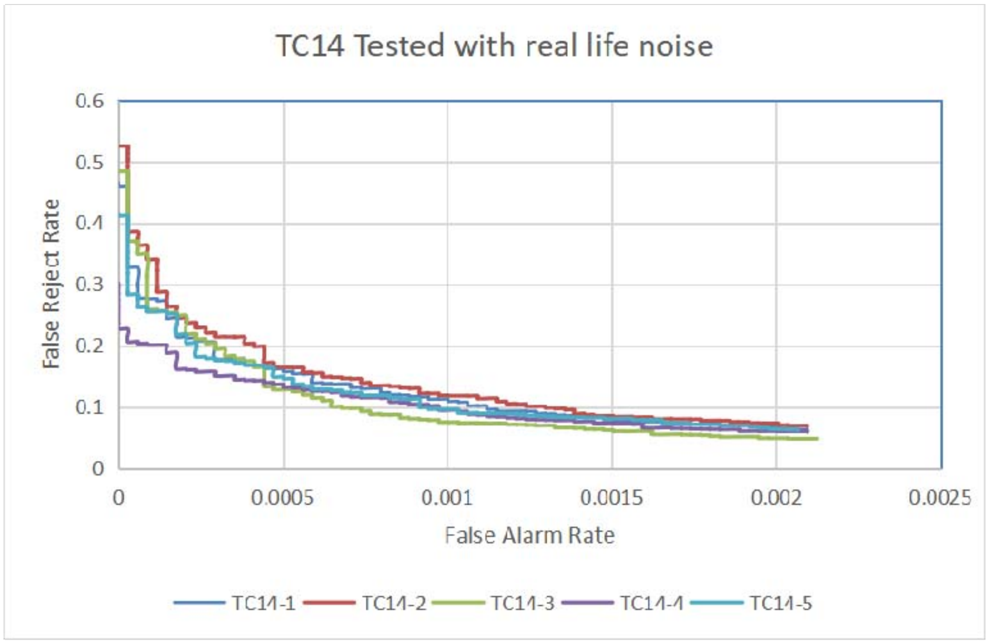


Fig 10. TC-ResNet ROC tested on augmented GSC 5 times

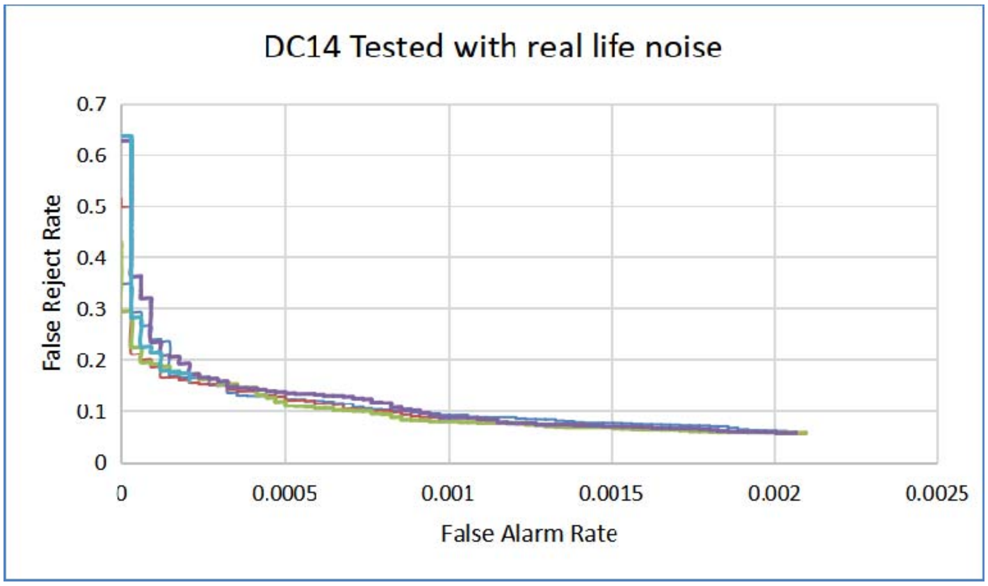


Fig 11.DC-ResNet ROC tested on augmented GSC 5 times

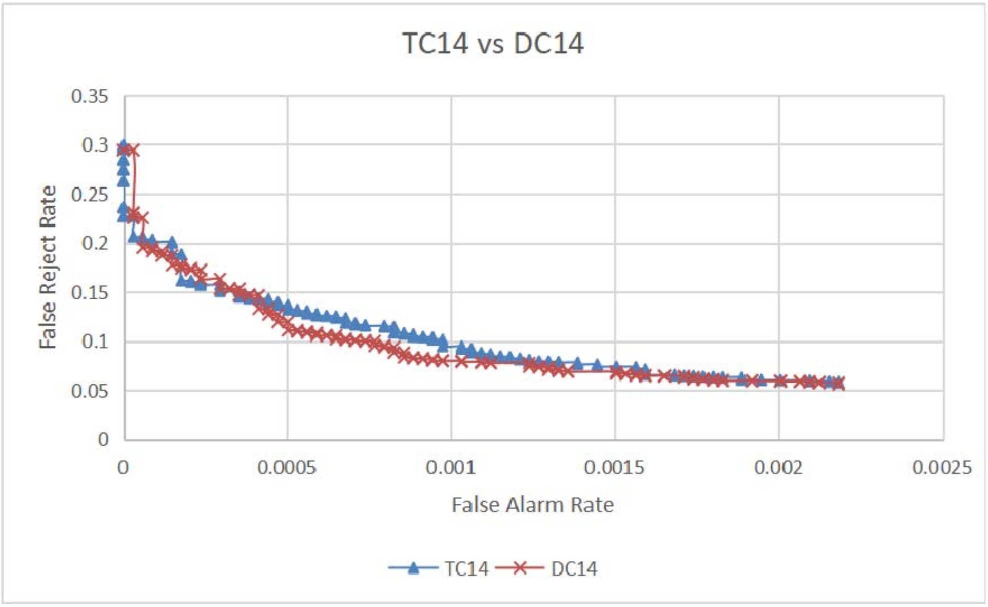


Fig 12. The comparison between the best TC-ResNet and the best DC-ResNet ROC tested on augmented GSC

Chapter 4

Conclusions

In this work, we compared various state-of-the-art KWS systems, including TC-ResNet, TDNN-SWSA, ResNet15, and DS-CNN. We also proposed a simple depthwise convolutional neural network, which greatly reduced the number of parameters, especially the computations, which is at most 300 times less than some of the existing models, and almost 3 to 5 times less than the state-of-the-art model.

1-D depthwise convolution utilized in this task is different from 2-D convolution and other types of 1-D convolution. The most preeminent advantage of it is that it drastically lowers FLOPs. Nevertheless, such small size with 12K parameters still cuts at least half of the FLOPs than other models. However, FLPOs are not directly proportional to the speed. When they are several, several hundreds or even several thousands times less, and at the same time other factors keep the same or even better, it means a lot. Without computational overhead, it meets the requirement of using on a real time. Finally, it achieved 17.5% increased speed compared to the fastest model, when it was tested on Samsung Galaxy S6 Edge.

For testing the robustness of the model, we adopted three different datasets: pure speech commands sets, silent noise augmented sets and the last one, training and tested on real life noise augmented sets.

In conclusion, this experiment can be a promising demonstration, implying a possibility of further implementations for the future.

Bibliography

[1] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. arXiv preprint arXiv:1707.01629, 2017.

[2] W. Xiong, L. Wu, F. Alleva, Jasha Droppo, X. Huang, and Andreas Stolcke. The microsoft 2017 conversational speech recognition system. CoRR, abs/1708.06073, 2017.

[3] Wang, Z., Li, X., & Zhou, J. (2017). Small-footprint keyword spotting using deep neural network and connectionist temporal classifier. arXiv preprintarXiv:1709.03665.

[4] Sainath, T. N., & Parada, C. (2015). Convolutional neural networks for small-footprint keyword spotting. In Sixteenth Annual Conference of the International Sp eech Communication Association.

[5] Zhang, Y., Suda, N., Lai, L., & Chandra, V. (2017). Hello edge: Keywordspotting on microcontrollers. arXiv preprint arXiv:1711.07128.

[6] Tang, R., & Lin, J. (2018, April). Deep residual learning for small-footprint key word spotting. In 2018 IEEE International Conference on Acoustics, Speech a n d Signal Processing (ICASSP) (pp. 5484-5488). IEEE.

[7] de Andrade, D. C., Leo, S., Viana, M. L. D. S., & Bernkopf, C. (2018). A neural attention model for speech command recognition. arXiv preprint arXiv:1808.0 8929.

[8] Arik, S. O., Kliegl, M., Child, R., Hestness, J., Gibiansky, A., Fougner, C.,... & Coates, A. (2017). Convolutional recurrent neural networks for small-footprint keyword spotting. arXiv preprint arXiv:1703.05390.

[9] Lee, L., Park, J., & Sung, W. (2019, December). Simple gated convnet for small footprint acoustic modeling. In 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU) (pp. 122-128).IEEE.

[10] Ian McGraw, Rohit Prabhavalkar, Raziel Alvarez, Montse Gonzalez Ar enas, Kanishka Rao, David Rybach, Ouais Alsharif, Ha¸sim Sak, Alexa nder Gruenstein, Françoise Beaufays, et al. Personalized speech reco gnition on mobile devices. In Acoustics, Speech and Signal Processi ng(ICASSP), 2016 IEEE International Conference on, pages 5955–595 9. IEEE, 2016.

[11] Jay G Wilpon, Lawrence R Rabiner, C-H Lee, and ER Goldman. Automatic recognition of keywords in unconstrained speech using hidden markovmodels. *IEEE Transactions on Acoustics, Speech, and Signal Processing*,38(11):1870–1878, 1990.

[12]Richard C Rose and Douglas B Paul. A hidden markov model based keyword recognition system. In Acoustics, Speech, and Signal Proce ssing, 1990. ICASSP-90., 1990 International Conference on, pages 12 9–132. IEEE, 1990.

[13] George Tucker, Minhua Wu, Ming Sun, Sankaran Panchapagesan, Ge ngshen Fu, and Shiv Vita ladevuni. Model compression applied to s mall-footprint keyword spotting. In *INTERSPEECH*, pages 1878–1882, 2016.

[14] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 5484–5488.

[15] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha,"Temporal convolution for real-time keyword spotting on mobil e devices," arXiv preprint arXiv:1904.03814, 2019.

[16] Y. Bai, J. Yi, J. Tao, Z. Wen, Z. Tian, C. Zhao, and C. Fan, "A time delay neural network with shared weight self-attention for smallfootprint keyword spotting," Proc. Interspeech 2019, pp. 2190–2194, 2019.

[17] S. Myer and V. S. Tomar, "Efficient keyword spotting using time delay neural networks," in Proc. Interspeech 2018, 2018, pp. 1264–1268. [Online].Availabl e: http://dx.doi.org/10.21437/Interspeech.2018-1979

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems 30, 2017, pp. 5998–6008.

[19] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," arXiv preprint arXiv:1711.07128, 2017.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.031 67, 2015.

[21] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. arXiv preprint arXiv:1707.01083, 2017.

[22] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Effificient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[23] P. Warden. (2017, August) Launching the speech commands dataset. [Online]. Available: https://ai.googleblog.com/2017/08/ launching-spe ech-commands-dataset.html

[24] S. Arik, H. Jun, and G. Diamos, "Fast spectrogram inversionusing multi-head convolutional neural networks," arXiv preprint arXiv:1808.06719, 2018.

Abstract

키워드 스팟팅(KWS)은 현재의 음성 기반 휴먼-컴퓨터 상호작용에서 중요한 역할을 하며 스마트 기기에서 널리 사용되고 있다. 신경망의 급속한 발달로 음성인식, 음성 합성, 화자인식 등 여러 음성 처리 분야에 걸친 어플리케이션에서 큰 성과를 거뒀다. 다양한 음성 처리 분야에서 강점을 보이고 있는 인공 신경망은 KWS를 위한 시스템에도 매력적인 선택이 되었다.

그러나 애플리케이션 환경은 스마트폰, 패드 및 일부 스마트 홈 기기를 포함한 소형 스마트 기기들이 대부분이기 때문에, 신경 네트워크 아키텍처들은 KWS 시스템을 설계할 때 이러한 스마트 기기의 제한된 메모리와 계산 용량을 고려해야 한다. 동시에 실시간, 사용자 친화적, 높은 정확도로 대응하려면 낮은 대기 시간을 유지할 수 있어야 한다. 또한 KWS는 다른 업무와 달라 상시 온라인 상태에서 이용자의 호출을 기다려야 하기 때문에 KWS 애플리케이션의 전력 예산도 크게 제한된다. 메인스트림 신경망 모델 중에는 과거 DNN, CNN, RNN, 그리고 서로의 조합이 주로 KWS에 사용되면서 최근에는 Attention 기반 모델도 점점 인기를 끌고 있다. 그 중에서도 CNN은 정확성과 견고성, 병렬처리가 뛰어나 KWS에서 널리 채택되고 있다.

본 연구에서는 효율적인 키워드 스팟팅을 지원하는 신경망 모델인 신플 콘볼루션 네트워크를 제시한다. 높은 정확도를 유지하기 위한 중간 과정으로 보다 컴팩트한 residual 네트워크와 노이즈 인식 훈련법을 주로 사용한다. ResNet은 좋은 성능을 얻기 위해 항상 수십만 개의 매개 변수를 필요로 하기 때문에, 우리 모델에서는 한정된 자원을 가진 스마트 기기에 더 적합할 수 있도록 depthwise 콘볼루션 네트워크를 사용하여 파라미터 수를 줄이는 법을 제시한다. 마지막으로 실제 모바일 기기인 삼성 갤럭시 S6 엣지에서 제안된 모델의 실제 추론 시간(즉, 지연 시간)을 측정하였다. 온라인 상 공개된 Google 음성 명령

데이터 집합이 모델을 평가하는 데 사용되었다. 결과는 제시된 모델이 기존 모델보다 약 1/2 의 매개변수와 계산 횟수를 훨씬 적게 사용한다는 것을 보여주며거의 동일한 정확도로 속도가 17.5 % 빠르며 6.9ms에 도달했다. 훨씬 작은 메모리 소모로도 다른 최신 KWS 모델을 능가하는 96.59%의 높은 정확도를 유지하고 있다.

Keyword : Keyword Spotting (KWS), Convolutional Neural Network (CNN), Small Footprint

Student Number : 2018-24547