Speech Recognition with SincNet Convolutional Filters

Instructor: 張添烜 Tian-Sheuan, Chang

0610144成耘瑄 Yun-Hsuan, Cheng
0610157李依庭 Yi-Ting, Lee

Introduction

With the rise of the pandemic COVID-19, we designed a machine learning model of speech recognition elevator control system in Mandarin, that could prevent contact infection through buttons. With the goal to apply machine learning on edge devices like elevators, it should have low power consumption. Hence, we used SincNet convolutional filter as the first layer of our model, an efficient way to reduce parameters and simplify computation. With the method of transfer learning, we combined big English Google speech dataset with our small self-collected data in Mandarin to achieve high accuracy. Finally, we simplified the model by implementing quantization and pruning to achieve our goal of low computation.

Structure

SincNet

SincNet is to combine Sinc-convolution with CNN. The first step is to parameterize raw audio by sinc functions, which acts as a band pass filter. Next, it performs convolution to extract signals that lie within a certain frequency range. The difference between SincNet convolution and CNN is that for each filter, SincNet convolution training only requires lower and upper cut-off frequency, that is, two parameters for each filter. Additionally, the number of parameters for one filter is from its kernel width, take this project as an example, the kernel width k = 51. As a result, SincNet is an efficient method on speech recognition tasks with low memory consumption.

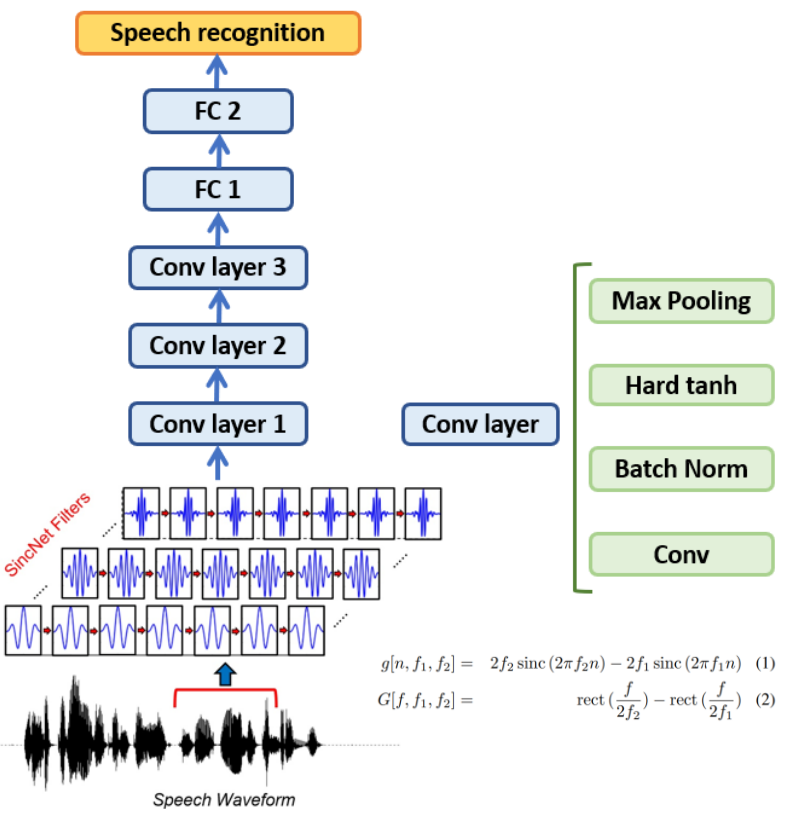

Fig 1 The model architecture as described in the Stucture part

Quantization

The weights of the trained model are originally presented as 32-bit floating point. However, the high precision is redundant, as we could have lower precision with the similar performance. In order to reduce memory consumption and accelerate the computation, we reduced the weights' precision to 8-bit (75% reduced), and it only had a small effect on the accuracy of the model (3-4% lower).

Pruning

Pruning is implemented by setting weights that are too small to zero. Since they are too small to influence the result, they could be reduced to simplify computation. According to table1, 20% weight pruned has the least effect on the accuracy. We speculate that the reason why only few weights can be pruned is because we had 8-bit quantization, which the precision has already been reduced.

Pruned weight	Accuracy
20%	**82.1%**
30%	73.2%
40%	59.0%
50%	21.4%

Table 1 Comparisons of pruning percentage

Results and Discussion

In our project, we collected 700 data in Mandarin, that is a relatively small dataset to train a model. Therefore, we used transfer learning to train the model with English Google speech dataset first. Next, we used our self-collected data in Mandarin to train the model. Finally, the model is able to recognize Mandarin with a better accuracy. As shown in Table 2, SincConv can result a better performance than only using 1D convolution, which is an almost 10% increase. The accuracy before transfer learning is 80.4%, and the accuracy after transfer learning is 83.9%, that is a 3.5% increase. The accuracy before quantization and pruning is about 89.2%, after quantization and pruning is about 83.9%, though the drop is quite large, but the calculation has decreased 80% (20% pruned and quantize 32-bit to 8-bit).

Model	Accuracy	Parameters
No SincNet Conv	75.0 %	62k
Before Channels Reduced	83.9 %	120k
No Transfer Learning	80.4 %	40k
No Quant (32-bit) & No Pruning	89.2 %	40k
Quant (8-bit) & Pruning (20%)	**83.9%**	**40k**

Table 2 Comparisons of results of different models

Fig 2 shows the difference of convergence between the use of transfer learning, SincConv, quantization, and pruning. We can see that the ones using transfer learning will converge faster and result in a higher accuracy. Since we use a small dataset, the fluctuation of the error rate will be more intense.

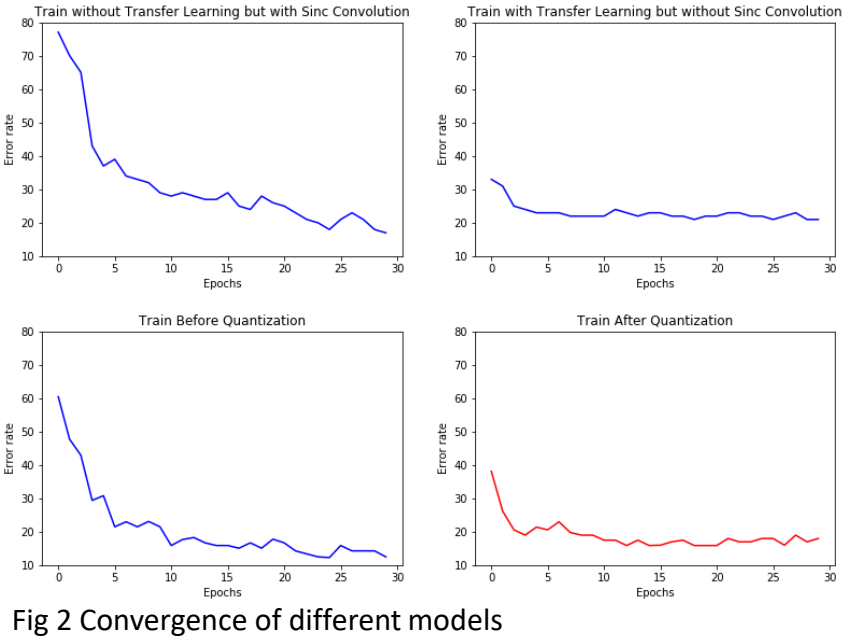

Fig 2 Convergence of different models

Our model does not require data preprocessing, such as MFCC, but is able to extract features from raw audio with SincConv layer. This makes the model easier to implement on hardware without adding additional hardware to transfer preprocessed features(MFCC), all we need is floating point calculation.

In conclusion, we used small dataset to train the speech recognition model, and in order to achieve low power consumption and low latency, we had the calculations that one audio file requires from originally 162M reduced to 34M. That is, for a 100M clock frequency FPGA board, the calculation time is reduced from 1.6 second to 0.4 second.

Reference

M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," in 2018 IEEE Spoken Language Technology Workshop (SLT), Dec 2018, pp. 1021–1028.