Keyword Spotting with Google Speech Command Dataset

Jack Yu haoxiy2@uci.edu 31082718

Rong Mu rmu1@uci.edu 58682019

Yiru Tang yirut2@uci.edu 18958047

Abstract

Keyword spotting (KWS) technology is increasingly reliable and has seen wide commercial applications, such as smart voice assistants such as Siri, Alexa and Google Assistant. But, few proposals have been made to use KWS technology to censor explicit language from the vast pool of user-uploaded online videos and audios to provide a safer digital life for young kids. We propose a CNN-based recurrent neural network to exploit the sequential nature of speech data and provide high-frequency prediction output, enabling audio post-processing to filter out explicit language content.

1 Introduction

Keyword spotting (KWS) is an important technique of speech recognition and speech processing that has been widely applied to smart voice assistant devices such as Siri, Alexa and Google Assistant for the recognition of voice activation commands. However, another area that can benefit from the KWS technology is left behind - explicit language censoring, which is much needed when a vast pool of uncensored online videos is easily accessible to minors. In order to censor explicit language, the model needs not only to know that a particular word has been spoken, but also to recognize the start and end time of its utterance. This special use case inspired the creation of our deep learning model.

In order to detect the spoken keywords, we aim to build a convolutional neural network (CNN) based recurrent neural network (RNN) as our main KWS technique that takes audios as input and detects select keywords from the Speech Command dataset. This model contains several advantages for KWS tasks and language censoring in particular. CNN enables feature extraction by processing the spatial relationship of input, that is, the time and frequency domain from the input audio data. RNN ensures the model is able to retain information from previous input windows information, exploiting the temporal dependency of audio input. It is ideal for keyword censoring because the model is able to detect keywords within a short interval with high frequency, potentially providing information about the accurate start and end times of a keyword, enabling a variety of post-processing procedures to provide a safer online environment.

During the development of this model, we mainly contributed by building different models from simple components to more complex aggregates, observing the advantages and disadvantages of different base models or combined models. Also, we focus on improving the performance of our CRNN model to equip it with high accuracy of multiple-keyword detection.

2 Related Work

The earliest naming of keyword spotting for speech recognition appeared fifty years ago[1]. Nowadays, with the expansion of machine learning, speech recognition is a more general description, including tasks about natural language processing and language detection. Keyword spotting can be categorized as a detection problem. In this project, we focus on audio keyword spotting rather than texts. Previous pioneers have developed a lot of models for this problem. We will mainly discuss three different models below.

2.1 Feature Extraction and DNN Model

The model published by Google in 2014 shows an example of a general framework for KWS. Their project consists of three major components: (i) a feature extraction module, (ii) a neural network, and (iii) posterior handling that performs the classification task. This paper proposes a deep neural network model (DNN) with conventional logistic and rectified linear unit (ReLU) functions[1]. Motivated by this paper, our project started off with a similar feature extraction approach and also with a feed-forward neural network using ReLU as activation functions.

2.2 CNN Model

Convolution Neural Network is another popular neural network model. Google's later paper implemented CNN as an alternative of DNN for KSW. Researchers pointed out several advantages of using CNN models, including representing local correlations in time and frequency and modeling translational variances with far fewer parameters due to different speaking styles. Researchers build a basic CNN architecture and compare the performance of the models with different multiplies and parameters in clean and noisy conditions, resulting in over 27% relative improvement in performance with limiting multiplies, and over 41% with limiting parameters. Motivated by this paper, our project started by replicating a CNN model for reducing the model size and better performance.[2]

2.3 RNN, LSTM, and Attention Models

A more advanced approach for keyword detection tasks is the neural attention model[3], which has an advantage in tasks involving long sequences. This is because these models can be used to understand what parts of the input are essential to predict outputs[4]. The research was done by Douglas Coimbra, el., etc. in 2018 proposed a neural attention model for speech

[1] C. Teacher, H. Kellett and L. Focht, "Experimental, limited vocabulary, speech recognizer," in IEEE Transactions on Audio and Electroacoustics, vol. 15, no. 3, pp. 127-130, September 1967, doi: 10.1109/TAU.1967.1161911.
[2] Tara N. Sainath, Carolina Parada, "Convolutional Neural Networks for Small-footprint Keyword Spotting" in *Interspeech* 2015.
[3] Bahdanau, D., Cho, K. & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, .
[4] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: neural image caption generation with visual attention. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15). JMLR.org, 2048–2057.

command recognition[5]. They preprocessed the speech command dataset into a mel-spectrogram, which is a better representation of audio in audio compression that is able to approximate the human auditory system's response. They used two 2-D CNN layers for extracting local relations first. Then, they applied an attention layer to identify which part of the input contributes most to predicting the output and used a two bidirectional long short-term memory model (LSTM) to capture long-term dependencies of audio. Compared to other models like CNN and ResNet, the attention model has a way higher accuracy, more than 95%. This paper gives us a great insight into the power of LSTM and the attention layer and motivates us to build the LSTM layer on the original CNN model.

2.4 Others

There are still more models that are developed based on architectures that are mentioned above, like Depthwise Separable Convolutional Neural Network and Temporal Convolution ResNet (TC-ResNet)[6]. Instead of going further, we decided to first replicate the CNN model for our project, and later extend it with LSTM and attention model.

3 Dataset

3.1 Source

The dataset we train our model with is the Google Speech Command dataset, which contains "65,000 one-second long utterances of 30 short words by thousands of different people, contributed by the public population via AIY website" and stored audio files in wav format Given the concentrated vocabulary of utterance examples, the Speech Command dataset is a preferred data pool to train and evaluate keyword spotting systems.

3.2 Preprocessing

For efficient audio representation, we perform two preprocessing steps. The first is the transformation of 1-second wav files into mel-frequency cepstrum using a 16,000 sample rate. The mel-frequency cepstrum consists of mel-frequency cepstral coefficients, or MFCCs in short. Traditional audio representation methods such as Short-term Fourier Transform converts audio into time and frequency domain**[Figure 1: STFT],** where the frequency domain has a large. dimension, often in the order of thousands. The large frequency dimension creates computational burdens for both training and inference. Although mel-frequency cepstrum is also 2-dimensional data, one of its axes being time, the other dimension is MFCC, which is a compact and concise representation of audio. At a given time step, the dimensionality of MFCC is usually near 20, two orders of magnitude less than that of Short-term Fourier Transform, significantly reducing

[5] de Andrade, D. C., Leo, S., Viana, M. L. D. S. & Bernkopf, C. (2018). A neural attention model for speech command recognition. CoRR, abs/1808.08929.
[6] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, S. Ha, "Temporal Convolution for Real-time Keyword Spotting on Mobile Devices," CoRR, vol. abs/1704.04861, 2019. [Online]. Available: http://arxiv.org/abs/1904.03814

input size and increasing run time efficiency**[Figure 2: MFCC]**. As a result, each wav file corresponds to a 20 x 32, where 20 is the number of MFCCs for each time step, and 32 is the number of discrete time steps for a one-second audio.

The second step is to slice the resulting mel-frequency cepstrum into smaller consecutive overlapping windows. This slicing approach is necessary to train our RNN-based model, which is elaborated in the Approach section. We use a window size of 16 and a stride of 4 to slice the mel-frequency cepstrum along the time axis, resulting in 5 sequential 2D inputs for each second of audio. If organized in batches, the input data has shape BxLxHxW (B is the batch size, L is the length of sequence data, H and W are MFCC and time dimensions of the input window).

4 Approach

4.1 Convolutional Neural Network

We want to start off our projects from a simple neural network that does not require transforming data to spectrogram. So the first model we tried out is a multilayer convolutional neural network that is replicated from a ConvNet designed for the speech classification task.[7] The raw waveform audio is loaded, with the length of the sample rate R. With the model having a batch size B, and in_channel of C, the input size is BxCxR (B=100, C=1, R=8000 as default). The model has four 1D convolutional layers and three linear functions as posterior classification handling. After each layer, we use dropout and maxPool1D layer respectively for reducing computation efforts and for reinforcing robustness. We have experimented with the model with the best dropout rate of 0.3 (Table 1).

Dropout Rate	Validation Accuracy (%), cmd=5
0.3	88.21
0.4	87.45
0.5	84.04
0.6	76.94

Looking for further improvement of this ConvNet, we advance it by preprocessing the dataset. The model architecture is modified to match input dimensions. With the batch size and in_channel remaining the same, the input dimension is extended to BxCxMxN, where MxN is the shape of the MFCC array. Accordingly, the model's convolutional layer changes from 1D to 2D, and we reduce the number of layers and kernel size since the model now has a much smaller length of inputs. We expected this advanced ConvNet to be faster and more accurate.

[7] A. Huq, "Speech Command classification using PYTORCH and torchaudio," *Medium*, 29-Sep-2020. [Online]. Available: https://medium.com/@aminul.huq11/speech-command-classification-using-pytorch-and-torchaudio-c844153fce3b. [Accessed: 09-Jun-2022].

Compared with more complex models, ConvNet has a prominent advantage of shorter running time and easier computation. However, with one chunk of raw data audio as inputs, the model lacks information about the speech contexts. Referring to the initial idea of designing this model, it is constrained to audio classification tasks only, and probably not suitable for keyword spotting tasks as the final target. Therefore, in the following, we will discuss combining this model with recurrent neural networks.

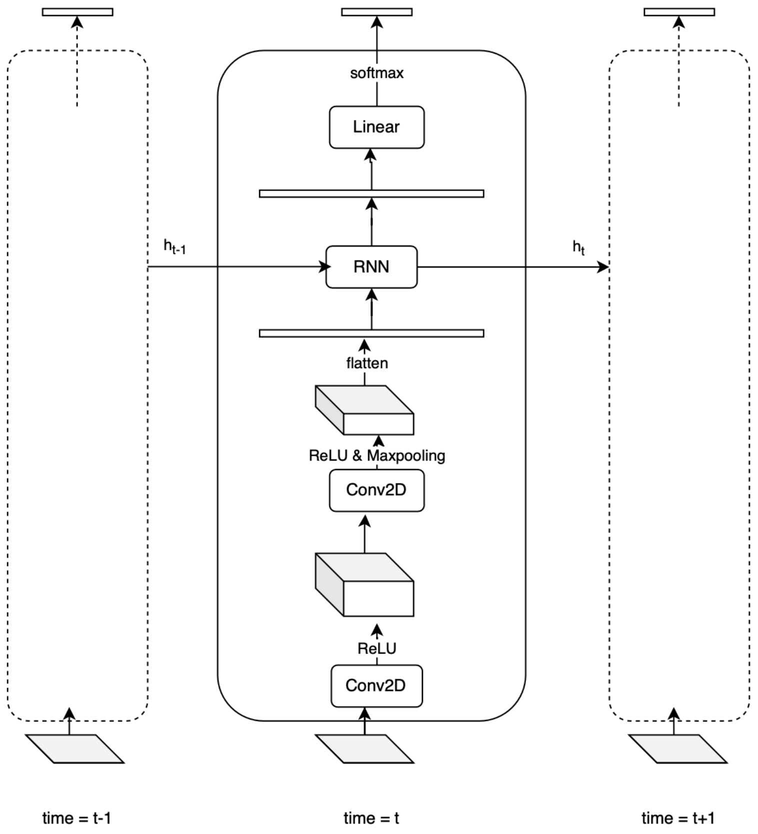
4.2 CNN-based Recurrent Neural Network (CRNN)

We use a sliding window approach. The window size should be a fraction of a second, roughly equal to or slightly shorter than the duration of a typical keyword utterance. The model takes pre-processed mel-frequency cepstrum as 2D inputs and produces an output vector for each input frame. Each output value indicates the probability of spotting the given keyword, or the lack thereof.

We design a CNN-based recurrent neural network architecture. 2D CNN layers are applied to extract features from the input mel-frequency cepstrum and further extract features from hidden feature maps. We chose CNN for feature extraction because audio contains temporal information - the arrangement of phonemes in a particular order can indicate certain pronunciations that the model tries to catch. We further exploit the temporally correlated nature of audio input by introducing recurrent architecture, feeding the hidden state h_t produced by the recurrent cell at time t as the input of the same cell at the next discrete time step t+1. The detailed architecture design is described below.

Each sliced window input at time step t is $I_t \ni \mathbb{R}^{f \times s}$ where f is the resolution of Mel-frequency Cepstral Coefficients (MFCCs) of the input mel-frequency cepstrum and s is the resolution in time. Then, we apply a 2D convolution layer that strides 1 step in both frequency and time with n kernels with size $p \times q$, in frequency and time respectively, to obtain feature map $U_t \ni \mathbb{R}^{n \times f \times s}$ activated by ReLU function. Then, stack another layer of convolution and ReLU activation on top. Next, we apply a maxPool2D layer to reduce the dimension of the convolution layer's output to a smaller feature map V_t. We then flatten V_t into a 1D vector v_t and feed it into the RNN cell and apply a softmax layer to the output of the RNN cell for classification.

To make the architecture recurrent, we experimented with different flavors of recurrent networks models, notably employing RNN and LSTM cells. The intuition is 1) if the window isn't long enough to contain the entire keyword, the pronunciation from the previous window contains valuable information to determine if the next window also contains part of the keyword. 2) if the previous window contains a word that often precedes a keyword, we want to use a recurrent connection to preserve such information. Both models were trained using the same hyperparameters.



5 Results and Evaluation

5.1 Convolutional Neural Network

For the basic ConvNet model, it has the highest validation accuracy of 93% on 5 speech command words, as the advanced ConvNet model using MFCC has the highest validation accuracy of 94.8%. With 20 speech command words, they have the highest accuracy of 87% (Figure 2) and 73.02% (Figure 3) respectively. The advanced ConvNet model runs significantly faster than the basic model, in respect to the smaller input size and fewer layers, but it has a much poorer performance as the number of labels increases. Referring to the report from this model's author, our replication shows a similar result, with the accuracy of the MFCC model dropping about 10% compared to the raw wav method. One thing to note is that we indeed achieve a performance boost with MFCC with few keyword commands. I think it can be attributed to the complexity of this model. MFCC refined audio features to the model, and we also simplified the model to adapt to the new input format. As a result, this model can be too simple to classify over ten command words. In addition, the best drop out rate also needs to be tested again.

Another phenomenon to explain is the overall higher testing accuracy than training accuracy. A normal accuracy plot should have the training accuracy higher than the testing accuracy. As epochs increase, the testing accuracy might exceed training accuracy, while sometimes not. As in our graph, the testing accuracy is always higher, which can be caused by unbalanced label distribution in training data and testing data. The Google Speech Command dataset we use has different amounts of audio for each keyword, and with simple train-test splitting based on 8:2 ratio cannot shuffle them with a balanced distribution. This is in the to-do list for us in the future model improvement.

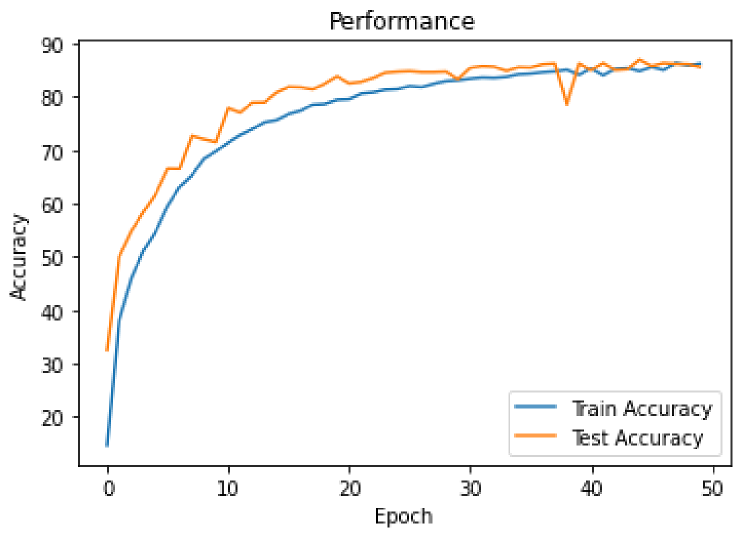


Figure 2 ConvNet with Raw WAV

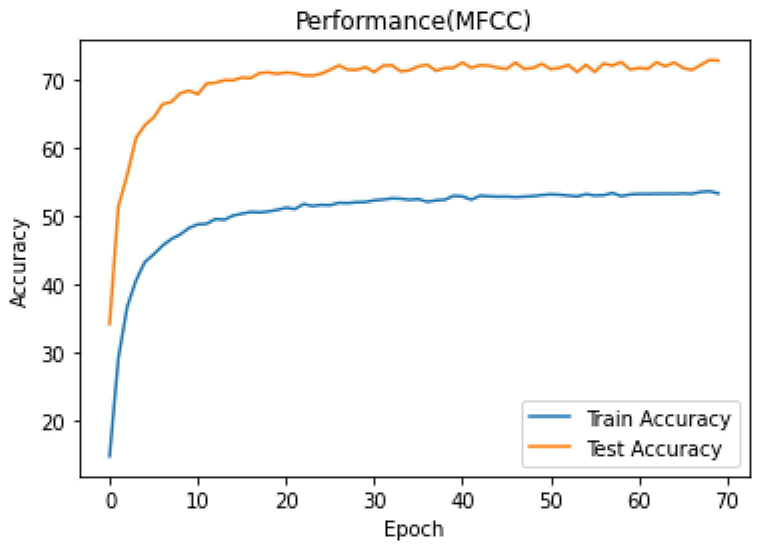


Figure 3 ConvNet with MFCC

5.2 CNN-based Recurrent Neural Network

Both the model with RNN cell and the one with LSTM cell achieved near 90% after 25 epochs. The training pace and results between the two versions of the model appear to be quite similar despite LSTM having longer-term memory. We reason that may be because LSTM cells aren't able to achieve their full potential under the training setup because all audios are only one second long, which means the sequence length is at most 5, potentially capping LSTM's ability to retain longer-term information.

Compared to the baseline CNN model, both versions of our recurrent network model underperform by about 3% margin. We believe this is partially attributed to the fact that the dataset (even after custom tailoring to generate more labels for each time step) doesn't directly cater to the need of our model's training process. Our model tries to detect keywords with high frequency by checking shorter time intervals more frequently. Because of this, some intervals, although labeled as a keyword according to the dataset, may just be empty audio since the actual length of different keywords can vary. Imprecise labeling of the data for our model could be one of the factors leading to its underperformance. In contrast, the traditional CNN model takes the entire 1-second audio (after transformation) as input, after which the top fully connected layer produces one single prediction vector, which is more suitable to the chosen dataset.

	Validation Accuracy(%)	
Model	**5-cmd**	**20-cmd**
ConvNet on raw WAV	93	87
ConvNet on MFCC	94.8	73.02
CRNN (RNN)	91.3	N/A
CRNN (LSTM)	91.5	N/A

6 Future Work

Due to time and computing resource constraints, we weren't able to fully explore the roles of some important hyperparameters. But further investigations remain our interest. For example, it is relevant to vary the number of layers within RNN and LSTM modules and observe its impact on model performance. Adjusting the stride and kernel size of the convolutional layers could also impact run time efficiency as well as model accuracy. Most importantly, stress testing our model by assigning it more keywords to detect can provide important insight into the model's adaptiveness to different application contexts.

7 Conclusion

In summary, our contribution is the proposal of CNN-based recurrent neural network architecture (CRNN) for KWS tasks, especially for explicit language censoring by taking advantage of the model's high-frequency detection ability. The model uses 2D CNN to extract features from the time and frequency domain, and utilizes RNN architecture to retain sequential information inherent in audio and human speech. Future work is still needed to obtain more suitable, better-labeled training data and to adjust various hyperparameters to explore the model’s true envelope of run time efficiency and detection accuracy.

REFERENCE

C. Teacher, H. Kellett and L. Focht, "Experimental, limited vocabulary, speech recognizer," in IEEE Transactions on Audio and Electroacoustics, vol. 15, no. 3, pp. 127-130, September 1967, doi: 10.1109/TAU.1967.1161911.

Tara N. Sainath, Carolina Parada, "Convolutional Neural Networks for Small-footprint Keyword Spotting" in Interspeech 2015.

Bahdanau, D., Cho, K. & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, .

Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: neural image caption generation with visual attention. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15). JMLR.org, 2048–2057.

de Andrade, D. C., Leo, S., Viana, M. L. D. S. & Bernkopf, C. (2018). A neural attention model for speech command recognition. CoRR, abs/1808.08929.

S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, S. Ha, “Temporal Convolution for Real-time Keyword Spotting on Mobile Devices,” CoRR, vol. abs/1704.04861, 2019. [Online]. Available: http://arxiv.org/abs/1904.03814

A. Huq, “Speech Command classification using PYTORCH and torchaudio,” Medium, 29-Sep-2020. [Online]. Available: https://medium.com/@aminul.huq11/speech-command-classification-using-pytorch-and-torchaudio-c844153fce3b. [Accessed: 09-Jun-2022].