CS230

Extended Hotword Detection to Arbitrary Audio Triggerwords

Kevin Yeun
Department of Computer Science
Stanford University
kyeun@stanford.edu
Github: https://github.com/kyeun/cs230-final

Abstract

We explore an extension on a deep learned sequence model to detect whether a model trained on a supervised speech recognition task (recognize a subset of words) can be extended to measure similarity between arbitrary audio clips of spoken words that are not from the original subset. There exist very accurate and performant hotword detection models today which are modeled as supervised binary classification problems with respect to a single triggerword (e.g. "Alexa", "OK Google", ...). A generalized model could provide very large utility to smaller independent companies and applications to use their own hotwords or for users to specify hotwords which are more natural in their native language.

1 Introduction

With hotword detection today, models are primarily trained specifically as a binary prediction model on whether a sound clip contains the word, with the model outputting the probability. An interesting application would be to create a model which may be used across many different use cases which may not share a common triggerword, for example in a platform scenario where a thin API client may be established and provided on any number of different client devices, or in a scenario where a user may want to customize the triggerword of their own device. Thus, we frame the problem statement: can a generic model be trained to work for any arbitrary triggerword, taking 1-second input sound clips and outputting a similarity measurement (substituting as a probability prediction) which can be used to predict whether the hotword should be activated?

For our model we contain three parts: a preprocessing step to convert he 1-second audio clip into a 2 dimensional spectrogram, a deep learned sequence model to output a softmax prediction on a subset of words available during training, and a similarity function which takes the softmax activation and outputs a prediction on whether a test sound clip is similar to a base sound clip and should activate the hotword detection. The softmax activation is treated as a representation of the original audio clip in a new basis vector space, with each dimension measuring similarity to one of the original training words.

2 Related work

State-of-the-art applications of hotword detection can most commonly be seen in the emerging "Smart Device" market, led by large technology companies such as Google and Amazon. "Small-footprint keyword spotting using deep neural networks" [1] describes a deep neural network used at Google

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

as an improvement over earlier Hidden Markov Models with additionally low footprint benefits. "Convolutional Neural Networks for Small-footprint Keyword Spotting" [2] published 1 year later in 2015 introduces Convolutional Neural Networks as another improvement upon the earlier model. CNNs are also proposed in "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition" [3] as a means to improve time-invariance in the model. The CNNs are trained to recognize the spectrogram representation of the audio clip which represent the 1-dimensional signal as 2-dimensional time and frequency. Spectrogram decomposition in conjunction to CNNs continue to be a popular choice for audio recognition tasks due to their non-lossy nature compared to other representations such as MFCCs (Mel-frequency cepstral coefficients) [4].

3 Dataset and Features

The Google Speech Commands dataset [5] is used, consisting of 105,000 16kHz WAVE audio files of 34 different words. Each file is labeled with the true word and is approximately 1 second in length each. For our analysis we used a word-balanced training set size of 33861 and a test set size of 1292.

Word	Number of Utterances
Backward	1,664
Bed	2,014
Bird	2,064
Cat	2,031
Dog	2,128
Down	3,917
Eight	3,787
Five	4,052
Follow	1,579
Forward	1,557
Four	3,728
Go	3,880
Happy	2,054
House	2,113
Learn	1,575
Left	3,801
Marvin	2,100
Nine	3,934
No	3,941
Off	3,745
On	3,845
One	3,890
Right	3,778
Seven	3,998
Sheila	2,022
Six	3,860
Stop	3,872
Three	3,727
Tree	1,759
Two	3,880
Up	3,723
Visual	1,592
Wow	2,123
Yes	4,044
Zero	4,052

Figure 1: Dataset content breakdown from "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition"

The first step of the extended model involves the preprocessing step to convert the 1-second audio clip into a 2-dimensional spectrogram representation. This input serves as the input to the deep learned sequence model. The 16,000 (corresponding to 16 kHz) length input is converted into 119 timesteps and 134 frequency steps, which represent 15946 features and is nearly lossless.

4 Methods

For the sequence model, we used a model consisting of the following layers: Conv1D(196 filters, kernel size=15, stride=4), BatchNormalization, RELU activation, Dropout(0.8), LSTM(128), Dropout(0.5), LSTM(128), Dropout(0.5), Dense, Softmax activation, with a total of 696,918 trainable

parameters. The model is built in Keras on Tensorflow [6] and is heavily inspired by the models used in the "Sequence Models" deeplearning.ai Coursera course [7] in "Emojify" and "Triggerword Detection".

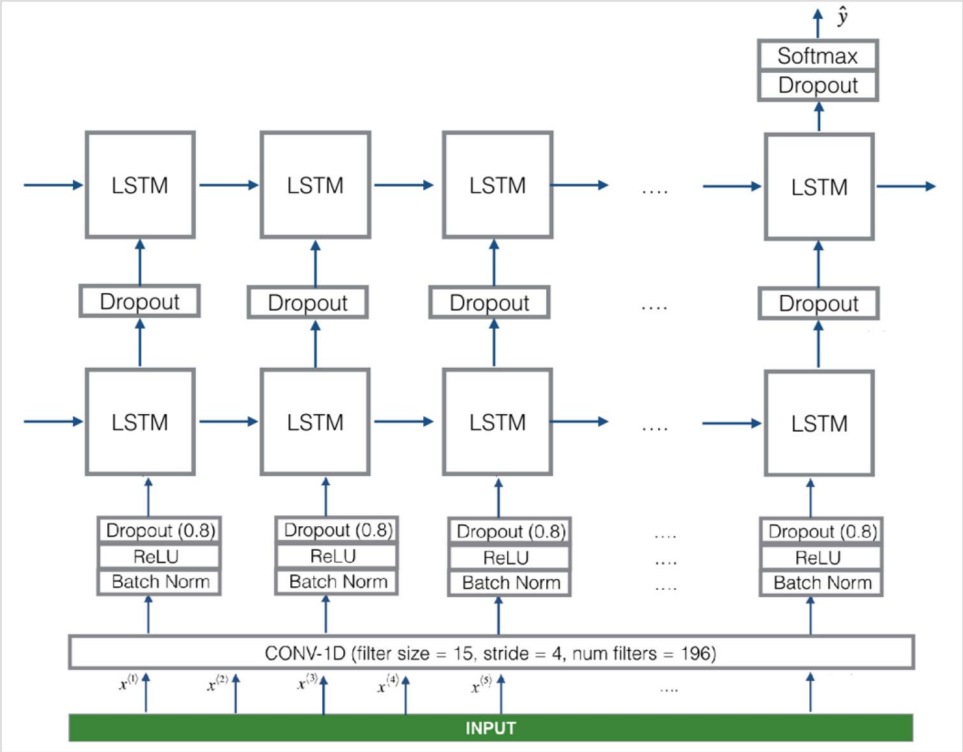

Figure 2: Deep learned sequence model with 696,918 parameters

The objective function used is the categorical crossentropy loss, which maximizes the true distribution likelihood. This is applicable to this use case as the softmax activation is treated as a likelihood estimation between the original set of training words.

$$-\sum_{b=1}^{M} y_{a,b} \log(p_{a,b}) \tag{1}$$

The softmax activation is fed to a final reduction function which outputs a similarity measurement between the base and test audio clips. We use the cosine similarity function

$$cos(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||} \tag{2}$$

with an additional hyperparameter α which determines a suitable similarity threshold for classifying the base and test audio clips as equivalent and activating the hotword detection.

5 Experiments/Results/Discussion

The sequence model is trained with Atom optimizatation (learning rate = 0.01, beta1 = 0.9, beta2 = 0.999, decay = 0.01) and a batch size of 1000. This learning rate is used since the model is trained without any basis model (from scratch) and has a medium sized category target, and it is able to complete 100 training epochs in under 2 hours on a 4-core VM. The sequence model itself performs quite reasonably achieving 0.6594 accuracy and 2.918 loss on the test set after 100 training epochs.

The extended model is then analyzed with some manual test inputs on words absent frmom the original training set. For 50 test clips (on new 25 words), and using α of 0.5, the extended model

Epochs	Train loss / accuracy (SM)	Test loss / accuracy (SM)	Mean / Variance Test Cosine similarity (same word, EM)	Mean / Variance Test Cosine similarity (other words, EM)
10	0.8811 / 0.7399	2.383 / 0.6076	0.8529 / 0.07874	0.04275 / 0.01168
50	0.5071 / 0.8407	2.667 / 0.6494	0.8975 / 0.06404	0.02779 / 0.008665
100	0.3681 / 0.8815	2.918 / 0.6594	0.9059 / 0.06291	0.02469 / 0.008303

Figure 3: Deep learned sequence model metrics

achieved about 78% accuracy and 42% recall. Figure 4 shows example waveforms for the word "Stanford", which achieved cosine similarity of 0.6696 and obtained leading activations for basis words "Sheila" and "One". This result supports the hypothesis that similar sounding words which are not from the training dataset could be supported by the extended model. It also suggests non-verbal utterances could be supported as long as the sound can be reconstructed in the input word vector space. Given the leading basis words lack large similarity with some of the syllables from the example word "Stanford", it is possible that the model would need additional curated basis words to perform well. This is further supported by the low recall on the manual test.

(a) Base

(b) Test

Figure 4: Waveforms for "Stanford" test audio

6 Conclusion/Future Work

The model shows promise on the ability for this type of model to exist. With the low number of training examples, naive basis words, and short training time, the model performs quite reasonably under those circumstances.

Future work includes exploring more specialized similarity functions for this task or finding more optimal word bases, either by dimensionality reduction (SVD) or involving acoustical characteristics of the words. Also, larger datasets should be used to evaluate the extended model. For the particular use case, it would be better to have a large dataset which includes many words spoken by a single person to avoid polluting recall metric due to different speaking patterns.

References

[1] Chen, Guoguo, Carolina Parada, and Georg Heigold. "Small-footprint keyword spotting using deep neural networks." ICASSP. Vol. 14. 2014.

[2] Sainath, Tara N., and Carolina Parada. "Convolutional neural networks for small-footprint keyword spotting." Sixteenth Annual Conference of the International Speech Communication Association. 2015.

[3] Abdel-Hamid, Ossama, et al. "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition." Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on. IEEE, 2012.

[4] Wyse, Lonce. "Audio spectrogram representations for processing with convolutional neural networks." arXiv preprint arXiv:1706.09559 (2017).

[5] Warden, Pete. "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition." arXiv preprint arXiv:1804.03209 (2018).

[6] Keras.io, 'Keras: The Python Deep Learning Library', 2018. [Online]. Available: https://keras.io/

[7] Coursera.org, 'Sequence Models', 2018. [Online]. Available: https://www.coursera.org/learn/nlp-sequence-models