DeepHark: Real-time Speech Recognition Via MFCC Neural Networks

Ryan Gibson
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599

May 2019

1 Introduction and Problem Motivation

Over the past few years, voice interfaces have become increasingly popular as the number of common speech-based services has grown (e.g. Alexa, Google Assistant, Siri). Indeed, across household appliances, in-car speech recognition systems, and smartphones, the average individual comes into contact with multiple virtual assistants each day.

However, despite this explosion of products that depend on robust speech recognition, it remains difficult to create even a simple speech detector. In fact, the precise details of Apple and Google's speech-to-text products are closely guarded trade secrets.

Moreover, many available transcription services are expensive and rely on offloading computation to compute clusters. This is inconvenient and introduces unacceptable round-trip latency in applications involving strict real-time deadlines.

In this paper, we present the DeepHark machine learning architecture which is both easy to train and can quickly recognize speech commands on low-end, commodity hardware.

2 Related Work

Machine learning has been widely applied in the field of automatic human speech recognition. In this section, we briefly review the more prominent approaches.

Historically, virtually every machine learning strategy has been used in automatic speech recognition in some way, including early work with Hidden Markov Models and

Bayesian Learning [1]. More recently, however, the focus of the research community has shifted to neural networks.

Similarly, significant progress has been made with using Gaussian mixture models in speech recognition. Such models were competitive with the state-of-the-art until the recent advances in deep learning [2].

In 2016, a team associated with the Chinese tech company Baidu presented a "end-to-end deep learning method" that achieved state-of-the-art performance in speech recognition without the need for feature engineering [3].

3 Methods

Here, we focus on the TensorFlow Speech Commands Dataset [4]. This dataset consists of 65,000 one-second sound clips of 30 short words.[1] Importantly, these sound clips were originally obtained in a crowd-sourcing effort that "didn't stipulate any quality requirements" so as to capture the type of audio one would obtain in a typical consumer/robotics application.

In order to make comparisons to competing methods, we follow the guidance of the Kaggle competition [5] based on this dataset and attempt to recognize the following 10 words in speech: "yes", "no", "up", "down", "left", "right", "on", "off", "stop", and "go". All other words are considered to be "unknown".

Our architecture consists of two primary parts: computing a short-term power spectrum of the input sound and then using these as the input features to a neural network.

The first half deals with computing a feature vector. We use the "Mel-frequency cepstrum" (MFC) representation for this purpose.[2] In short, this is a representation that takes a short window of a sound clip and

1. Separates the sound into its spectrum (i.e. a sequence of its frequency components and their associated strengths) via a Fourier transform.

2. Bins the frequency strengths, using overlapping windows on a log-scaled version of the Mel pitch scale to correct for human sound perception being logarithmic in both pitch *and* loudness.

3. Computes the Discrete Cosine Transform of the binned frequency strengths (i.e. computing a "spectrum-of-spectrums") and uses the resulting coefficients as the final features. This decorrelates the strengths from the overlapping windows in step 2 and allows us to drop the high-frequency spectrum changes from our

[1] All sound clips are in a "16-bit little-endian PCM-encoded WAVE format at a 16000 Hz sample rate" and trimmed to a one-second length if needed.

[2] We use the Python implementation in the python_speech_features library and have adapted our explanation of MFC from their documentation [6].

features, which has been shown to improve performance in speech recognition tasks.

Using this, our feature engineering proceeds as follows for each sound clip.

1. Pad the end of the sound with silence if presented with fewer than 16000 data samples

2. Compute the Mel-frequency cepstrum coefficients (MFCC) from the input sound with a window length of 10 ms and 50 coefficients per window.

Once flattened, this gives us a feature vector of length 5000 that we present to our neural network. The architecture of this network is remarkably simple and only consists of four fully connected layers of size 5000, 50, 50, and 11, respectively. All layers used Leaky ReLU activation functions [7], 10% dropout, and were initialized with Xavier uniform weights [8].

The network was trained with cross-entropy loss (corrected for the size of the training classes since the "unknown" class is significantly larger than the rest in this dataset) and the Adagrad optimization algorithm [9] with early stopping.

Our implementation using PyTorch [10] can be found at `https://github.com/deephark/DeepHark`.

4 Results

We implemented and tested our architecture on a machine with an i7 9700K CPU and GTX 1070 GPU. As suggested by the dataset creators [4], we split the sound files into three sets: 80% training, 10% validation, and 10% testing.

After training for approximately 10 minutes, we obtained a classification accuracy of ∼81% on the testing set and we show the normalized confusion matrix in Table 1.

Notably, our model tends not to misclassify sound clips (with the exception of distinguishing between "no" and "go"), but simply predicts "unknown" with problematic inputs. Moreover, our model's performance here is relatively close to state-of-the-art. Indeed, the best known classification accuracy on this dataset is just over 90% [5] (recall that the audio was obtained via crowdsourcing with no quality requirements).

Importantly, the simplicity of our model gives us *extremely quick* inference times. When running on the CPU, we found that the MFCC computation from [6] takes ∼10–15 us per 10 ms window and the forward inference time of the network is ∼1–2 us, on average.

For reference, the sound clips of interest here have 62.5 us between samples (since the sample rate is 16 KHz). Hence, we expect that this model would be feasible on very low-end embedded systems, especially if hardware support for the Fourier and Discrete Cosine transforms is available.

		Actual Label										
		yes	no	up	down	left	right	on	off	stop	go	unknown
Predicted Label	yes	82%				4%	1%					1%
	no		57%		6%	1%				1%	14%	1%
	up	1%	2%	67%		1%		1%	4%	4%	4%	1%
	down	1%	2%		58%		1%				2%	2%
	left	5%	1%	1%		64%	2%		1%		2%	1%
	right	1%				2%	69%					2%
	on			1%				69%	2%		1%	2%
	off	1%	1%	1%		3%		1%	77%		1%	1%
	stop			4%		1%				82%		1%
	go		12%	3%	4%	1%	2%	2%	1%	1%	53%	2%
	unknown	9%	24%	22%	30%	22%	25%	26%	13%	12%	23%	89%

Table 1: Column-normalized confusion matrix of our performance on the test set (i.e. each column represents the per-class classification percentages of our model). Percentages that round to zero are omitted.

5 Conclusion and Future Work

In this paper, we have presented a machine learning architecture that achieves "near state-of-the-art" classification performance in speech command recognition tasks while being quick to train and having extremely short inference times.

Future work should focus on extending this approach to situations in which the set of words is not known in advance.

Future work could also analyze tweaking our architecture for increased performance, though we have found that DeepHark performs significantly better than small RNNs when the allotted training time is limited.

Lastly, we believe that our method's speed lends itself towards integration into larger ensemble-learning strategies or fast feature engineering. As such, more work should focus on the applications of DeepHark as part of other machine learning frameworks.

References

[1] L. Deng and X. Li. Machine Learning Paradigms for Speech Recognition: An Overview. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1060–1089, May 2013.

[2] L. Deng, G. Hinton, and B. Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603, May 2013.

[3] Dario Amodei et al. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 173–182, New York, New York, USA, June 2016. PMLR.

[4] Pete Warden. Speech Commands: A public dataset for single-word speech recognition. *Dataset available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz*, 2017.

[5] Kaggle.com. TensorFlow Speech Recognition Challenge. *https://kaggle.com/c/tensorflow-speech-recognition-challenge*.

[6] James Lyons. python_speech_features: A library providing common speech features for ASR including MFCCs and filterbank energies. *https://github.com/jameslyons/python_speech_features*.

[7] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv:1505.00853 [cs, stat]*, May 2015. arXiv: 1505.00853.

[8] Xavier Glorot and Y Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, 2010.

[9] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[10] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017.