Key Word Spotting Systems

Supervised Research Exposition

Archiki Prasad (160110085)

22 March 2020

1 Introduction

We look at the problem of key word spotting (KWS) and specifically its performance on noisy data. In the current literature, KWS is used synonymous with Wake-Word Detection. So we have explored the exhaustive literature already present in both the fields. We find that the field is dominated by industry based solutions and thus the models are often not clearly explained if not proprietary. The same holds for the datasets where we found only two publicly available datasets- Google Speech Commands [1] and Hey Snips! [2]. The later did not seem as prominent with only one keyword and the negative examples did not seem as 'relevant'as they could have been, along with the fact that this dataset is used mainly by the authors that provide this dataset. Google Speech Commands [1] is more widely used in the academic implementations and papers and we explored that next. We observe that this dataset is relatively simpler but has larger number of options for the keywords. However, the results of the existing state of the art models are already very promising. To shed some light on this we look at some of the implementations and evaluate the performance, which is described in the sections below.

2 Problem Setup

Keyword spotting usually referred to as the problem of identifying keywords in an utterance. Wake word detection refers to techniques/algorithms that searches a stream of audio for a special word called the wake-word and then triggers or activates another general purpose algorithm. The most common application is use of wake-words like "Ok Google", "Alexa", "Hey Siri" to activate a personal assistant.

In scientific literature, KWS is used synonymously with WWD and involves simple models that can detect the declared keyword(s). This can be thought of as a task of classifying audio snippets as positive or negative examples. There are not only different labels for keywords, but also for "unknown" (and sometimes "silence"). For such a setup a natural measure of performance is accuracy, which we have used in this work. One can also look at other measures such as F1 scores, false positives, false rejects, area under the curve (AUC) etc. The underlying models will be discussed in greater detail in sections 3 and 4 below.

3 Related Work

There is an abundance of literature and techniques for keyword spotting. In this line of work, often dataset and complexity of the problem go hand in hand. As mentioned previously there is a very large contribution from the industry with proprietary datasets and techniques, leaving very few publicly available datasets, the most prominent of them being Google Speech Commands [1]. Due to the presence of valuable commercial applications, there is a large push for 'small-footprint' models- which can be easily deployed on edge devices under constrained settings. This involves restricting the number of parameters and operations of the model to deal with the memory and compute constraints, described in detail in [3].

The first prominent work in this area was by Google in [4] using Convolutional Neural Networks (CNNs) based models which were compared against DNN based models. The performance was found to be promising and several methods like strides and pooling layers were described to control and reduce the number of parameters and multiply operations. These models are commonly found as the baseline in future work and settings. Another popular technique [5] uses LSTMs for the 'Alexa' dataset, which demonstrated that instead of using cross-entropy (CE) loss for training, one can use the max-pool loss. The intuition is that instead of firing at every frame, the network should be trained

based on the most confident firing, captured by the max-pool loss. The authors found best results by initialising with CE trained models and then using max-pooling loss. However the actual values of the performance measures are not declared in the paper but a comparative analysis is shown.

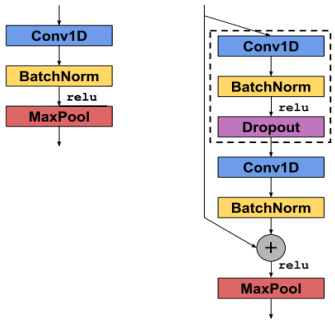
A natural extension of work in [4] is to look at deeper networks and draw inspiration from the success in the image classification specially by residual learning [6]. The use of ResNet architecture and the application to small-foot print KWS is given in [7] and subsequently using dilated networks similar to Wavenet [8] in [2]. Other lines of work include generating adversarial examples and adding these to the dataset to make models robust [9]. Data augmentation for robustness under real-life conditions which involve ambient noise from external sources like TV, home appliances etc and imperfect cancellation of the audio playback from the device, resulting in residual echo is explored in [10] using a proprietary dataset by the "Alexa" team. There is also some work on models trained on non-English keyword(s) such as [11], which uses Mandarin keywords.

4 Approaches and Implementation

We use 10 keywords from the speech command dataset, namely- *yes no up down left right on off stop go* which are frequently used in several previous work. We use three types of prominent models as baselines and each of them are described in detail below. All the implementation are done in PyTorch.

4.1 SampleCNN

In [12], the authors extend the architecture of Sample CNN and use it on Keyword Spotting as an audio classification task. This model is waveform based as opposed to spectrogram based approaches, similar to wavenet, with increasing receptive fields. We have used the Resnet based extension provided by the authors which is summarized in the Figure 1.



(a) Basic block (b) Res-n block

Figure 1: Block description of the blocks used in the model, taken from [12]

Residual Networks (ResNets) allow to train a very deep CNN using skip connections [6]. The shortcuts make backpropagation of gradients more fluent in the neural networks. The Res-n block shown in Figure 1 show 1 skip connection, with n standing for the number of conv layers used. The value of n is 1 without the dotted part and 2 otherwise. In our implementation we have used 8 Res-*2* blocks, as per the author provided Tensorflow code [13].

4.2 Conv Based networks

KWS system started off by using Hidden Markov Models and later progressed to Deep Neural Networks (DNNs), giving better performance and the ease to adjust the model size by limiting the number of parameters. Later, Convolutional Neural Networks (CNNs) showed improvements over DNNs in the task of acoustic modelling for small vocabulary tasks. This provided a logical motivation for using CNNs in KWS models, and was first done in [4]. The advantages of using CNNs over DNNs are two fold. First, unlike DNNs which ignore the input topology, CNNs can model the correlations along time and frequency axis in the spectral representations. Second, CNNs can handle transalation variations found

in speech efficiently. In [4], the authors provide convolutional layer based models for small footprint keyword spotting under the constraint of number of parameters and number of multiple operations. This was achieved by experimenting with depth, stride and pooling in the models. We use their *trad* model and the best small foot print *one-stride1* model with appropriate trade off with accuracy. Please refer to the paper and the implementation for further details.

4.3 Deep Residual Networks

Section 4.2 details the work involving use of CNNs for KWS. Deep residual networks had shown promising results in image-recognition [6] and were subsequently applied to speaker identification and automatic speech recognition and resulted in improvements. This motivates the use of ResNet like architectures for tasks like KWS. The intuition behind this is that deeper models lead to better performance. One way of achieving this and mitigating the vanishing gradient problem is by using skip connections. As detailed in section 4.1, they serve as shortcuts by adding the outputs of previous layers to the outputs of the stacked layers, resulting in an easier path for gradient flow and thereby facilitating backpropagation in deeper networks. In [7], the authors provide a ResNet like model with residual blocks for KWS and then provide methods to decrease the number of parameters to obtain 'narrow' small footprint versions for the same. They combine the residual blocks proposed by [6] with dilated convolutions (d_w, d_h) to increase the receptive field of the network which can capture long-term dependencies. The residual block comprises of bias-free convolutional layers, with height and width as *(m,r)* respectively, followed by ReLU activation and batchnorm as shown in figure 2. We use the the best performing *res-15* models for our baseline.

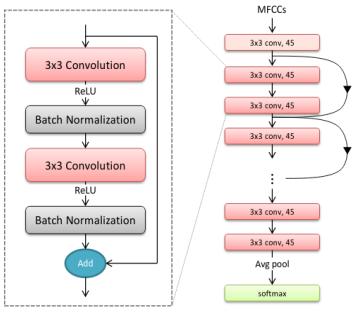


Figure 2: Expanded description of the residual block given in [7]

The reduction in footprint, calculated by reduction in number of parameters and multiplication operations is achieved by reducing the number of feature maps (n) from 45 to 15. Alternatively, the authors explored reduction depth (given in *res8* architecture) but that did not achieve significant dent in number of parameters as achieved by their 'narrow' approach. The narrowest model proposed is the narrower version of *res8* model known as *res8-narrow* As per the nomenclature used in the paper to describe the *res-15 architecture* the parameters are mentioned below:

type	m	r	n	d_w	d_h	Par.	Mult.
conv	3	3	45	-	-	405	1.52M
res × 6	3	3	45	$2^{\lfloor \frac{i}{3} \rfloor}$	$2^{\lfloor \frac{i}{3} \rfloor}$	219K	824M
conv	3	3	45	16	16	18.2K	68.6M
bn	-	-	45	-	-	-	169K
avg-pool	-	-	45	-	-	-	45
softmax	-	-	12	-	-	540	540
Total	-	-	-	-	-	238K	894M

Figure 3: Parameters of *res-15* taken from [7]

5 Results

For the spectrogram based models, we used audio samples with sampling rate of 16 kHz, 40 mel coefficients with hop length of 20ms. For SampleCNN, given in 4.1, we resample the speech files and the noise files to a sample rate of 22.05 kHz and the samples are fed directly into the model or else after adding noise depending on the experiments. We use Adam optimizer [14] with learning rate as 0.001 and batch size of 32. For the remaining models, described in sections 4.2 and 4.3, we use SGD optimizer with learning rate of 0.001 (the *res* models use a decaying learning rate schedule). Further details are available in the implementation and the log files[1]. The spectogram based models are adopted from

[1]The private code repository can be found at: https://github.com/archiki/KeyWordSpotting-Models

the implementation given in [15].

In Google Speech Commands, the speech files corresponding to the commands are about 1s long. So for the noise part, the background noise is truncated to this length before adding from a random starting point. We retrain the models by adding this noise selected at random uniformly to the clean speech samples with probability of 0.8. We use the additive background noise available in [16] along with the default background noises available in the speech commands dataset for different SNR values. This results in 5 experiments, 'Regular' stands for using clean/minimal noise in the train, test and validation sets(the splits are 80,10,10 for all experiments). The remaining experiments are labelled with the SNR obtained after the noise addition, ie, in the experiments labelled +5dB the noise is added such that the resulting SNR is 5dB and so on.

Model	**Accuracy (%)**					**#Parameters**
	Regular	Noisy (SNR)				
		+10dB	+5dB	-5dB	-10dB	
SampleCNN	90.94	92.81				11,627,028
Conv *trad*	90.22	91.44	90.52	89.5	88.34	623,500
Conv *one-stride1*	72.81	71.9	74.77*	67.81	60.94	994,502
res15	96.28	95.84	95.74	95.11	94.01	237,882
res-15-narrow	94.17	94.27	93.72	91.08	89.32	42,648
res8	94.44	94.58	93.61	91.02	90.30	110,307
res-8-narrow	90.93	90.54	89.18	87.17	84.22	19,905

Table 1: Performance comparison of various models under different noise conditions specified by the SNR values.

* Trained for considerably larger number of epochs, saturates here. It is clearly noticeable that Conv *trad* has smaller number of parameters than Conv *one-stride1* although the later is a narrower formulation. This has happened because the later has been set up as per [7] but for the former I was able to change the model parameters with reduction in number of parameters and not much drop in the accuracy. For the narrower version of the same one would notice smaller number of parameters.

6 Discussion and Conclusions

We notice form the Table 1 that the SampleCNN model has the highest number of parameters, at least 2 orders of magnitude greater than the other models, accompanied with slower training. This leads us to believe that it might not be a good fit specially for small-footprint tasks. The accuracy is also not as good as the spectrogram based models, this is also indicated in the paper and a possible work around can be increasing the number of layers. However,this would further increase the number of parameters making it unsuitable.

We see that the Conv models have high number of parameters which can be attributed to the linear layers in the network. The narrower model (Conv *one-stride1*) however suffers a significant drop in performance and this gap appears to increase with diminishing SNR, and does not seem very promising in high noise settings.

The ResNet based models look very promising even in very high noise settings, specially *res15*. The narrower version (including *res8*) also do well but we see an expected trade-off between performance and number of parameters. One may also notice that the narrower models see higher drop with decreasing SNR but even the smallest model- *res-8-narrow* does better than the narrow Conv model and comparable to the Conv *trad* model.

In this work, our dataset is restricted to one word keywords, so it will be interesting to see what happens as the number of words in the keywords increases. In [4] we notice that the dataset used comprises of two word keyword phrases. The frame work that they have used comprises of feature extraction, neural network model and posterior handling as shown in fig 4. In the posterior handling module, individual frame-level posterior scores from the models are combined into a single score corresponding to the keyword.

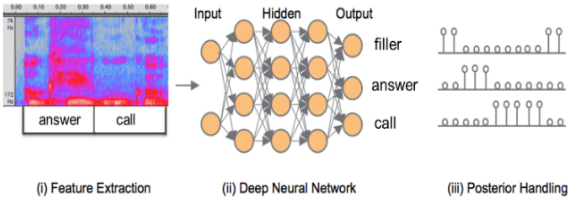


Figure 4: The schematic of the framework given in and taken from [4]

We believe that if we have well performing model for single word detection then this can be extended using this approach for multi work keyword phrases. The challenge here is that there is no multi-word keyword publicly available dataset known to us, so we are demonstrating that in the one-word setting the emphresnet models perform better than the *conv* models and assume that with other things remaining the same, when put in the framework mentioned above, the *resnet* models will outperform *conv* models in multi-word setting by extension. More extensive techniques for calculating keyword scores from posteriors like beam search decoding and smoothening may be needed as we move towards higher number of words in keywords. If there is a grammar specific to the dataset/task then one can think of a small language model (LM) for combining the posteriors. Such techniques are very commonly used in the field of Automatic Speech Recognition (ASR).

References

[1] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[2] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril, "Efficient keyword spotting using dilated convolutions and gating," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6351–6355, IEEE, 2019.

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

[4] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[5] M. Sun, A. Raju, G. Tucker, S. Panchapagesan, G. Fu, A. Mandal, S. Matsoukas, N. Strom, and S. Vitaladevuni, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting," in *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 474–480, IEEE, 2016.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[7] 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)*, pp. 5484–5488, IEEE, 2018.

[8] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[9] X. Wang, S. Sun, C. Shan, J. Hou, L. Xie, S. Li, and X. Lei, "Adversarial examples for improving end-to-end attention-based small-footprint keyword spotting," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6366–6370, IEEE, 2019.

[10] A. Raju, S. Panchapagesan, X. Liu, A. Mandal, and N. Strom, "Data augmentation for robust keyword spotting under playback interference," *arXiv preprint arXiv:1808.00563*, 2018.

[11] Z. Wang, X. Li, and J. Zhou, "Small-footprint keyword spotting using deep neural network and connectionist temporal classifier," *arXiv preprint arXiv:1709.03665*, 2017.

[12] T. Kim, J. Lee, and J. Nam, "Comparison and Analysis of SampleCNN Architectures for Audio Classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 285–297, 2019.

[13] T. Kim, J. Lee, and J. Nam, "SampleCNN for audio classification." https://github.com/tae-jun/sampleaudio, 2019.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] R. Tang and J. Lin, "Honk: A pytorch reimplementation of convolutional neural networks for keyword spotting," *arXiv preprint arXiv:1710.06554*, 2017.

[16] C. K. Reddy, E. Beyrami, J. Pool, R. Cutler, S. Srinivasan, and J. Gehrke, "A scalable noisy speech dataset and online subjective test framework," *Proc. Interspeech 2019*, pp. 1816–1820, 2019.