Speech Command Recognition Using Deep Neural Network

Ahmad Ishtar Terra
terra@kth.se

Hassam Riaz
hassamr@kth.se

Abstract

Speech commands are useful in operating gadgets and devices without physical interaction. There are a number of Automatic Speech Recognition Systems which can predict speech commands with high accuracy but they are computationally very expensive. This project aims to build a speech recognition system which has a small memory footprint, is computationally cheap and has a decent accuracy, that can classify predefined speech commands. Using the Speech Commands Dataset provided by Google AI, we have implemented deep learning architectures using Deep Neural Network and Convolutional Neural Network. Convolutional Neural Network outperforms Deep Neural Network model and can achieve accuracy of 92.38% while predicting 31 speech commands.

1 Introduction

1.1 Background

Due to the rapid advancement and development in the technology, a lot of devices are available to assist people in their daily life. However, many of those tools and assistants require some form of physical interaction in order for us to utilize them. Moreover, it is nearly impossible for people with motor disabilities to use these kind of devices. In recent years, a handful of voice assistants were introduced to resolve this issue but they require specialized compatibility to operate other devices. Thus the motivation of this project is to build a speech command recognition system that is capable of detecting predefined commands and helps devices to interact differently based on the commands given to it. To make the command recognition system compatible and able to run on every device, it must have a very small memory footprint, and require only very small computation.

1.2 Previous Work

There are a number of Automatic Speech Recognition (ASR) methods that recognize the speech and can help in converting it into a transcription or for further processing. Template matching is one of the early ASR methods that compares an utterance to a collection of recorded utterances in order to find the best possible match [1]. This method is simple and easy to implement as it does not involve phonetic transcription in the process. However, this method is not feasible in recognizing large number of words as we should have a recording of every word in the database and it only works with isolated words. It is not easy to model the variations of the speech in these kind of template matching techniques because it depends on various factors. People may have different accents and vocal tracts that could affect their voice properties. Varying background noises is another factor which affects the speech characteristics. In order to overcome these factors for such systems, it was necessary to record training/sample speech data in the same environment where the system has to be used.

Machine learning has been proven to have a powerful ability for classification tasks [2]. Commonly used probabilistic modeling techniques like Hidden Markov Model (HMM) also impacted Automatic

Project Speech and Speaker Recognition (KTH-DT2119 2018), Stockholm, Sweden.

speech recognition (ASR) [3]. Hidden Markov Model (HMM) is a generative approach, an HMM model is trained on the keywords (word to recognize), and a filler HMM model is trained on the non-keyword segment of the speech signal. This approach works effectively on large vocabulary by splitting an utterance into a collection of phonemes and identifying each phoneme separately. However, it is very computationally expensive, since HMM requires Viterbi decoding. Other recent works like discriminative models based on large-margin formulation or recurrent neural network [4] show some improvements over HMM approach, but their major problem is having relatively long-latency as they either require to process over the whole speech to find the region of specific word or take inputs from a long period of time to predict the word. The current system at Google [5] uses frame-wise inputs to make posterior probability predictions that are then combined and fed to simple deep neural network, which outperforms the traditional systems. We believe that a Convolutional Neural Network (CNN) model can provide further improvements over Google's model in a variety of small and large vocabulary tasks [6].

1.3 Proposed work

When Hinton proposed layer-wise pre-training network [7] in 2006, neural networks gained popularity in Automatic Speech Recognition (ASR) tasks. However, recognizing speech using neural network requires a lot of resources in training the network. Our project aims to recognize a spoken word without phoneme model by using Speech Commands Dataset from Google AI [8]. Initially, we use fully connected feed-forward deep neural network and then develop a Convolutional Neural Network to compare the performance of the recognition system on the basis of these architectures. We use full utterances as an input to the network and make a single softmax layer computation rather than using combined frame-wise inputs. We found a similar experiment done by Li et al. [6] that also contributed to this idea but they focused on recognizing only 6 speech commands.

2 Method

This section explains the dataset, preprocessing techniques and the networks and their architectures used in the experiment.

2.1 Dataset

We are using the Speech Commands Dataset provided by Google AI which contains 64,721 WAVE audio files. These audio files contain one-second long utterances of 30 simple voice commands ('stop', 'go', 'yes', 'no', etc) recorded by thousands of different individuals through AIY website [8]. The dataset is split into three parts, training data (80%), validation data (10%), and test data (10%). However, around 10% percent (8,203 files) of the data has less than one second duration. We added zero value (padding) around these data so we have the same dimension for the whole dataset.

Instead of having silence samples with one second duration, the dataset contains six different background noises with longer duration ($\geq$ 1 minute). We generated silence dataset by splitting the background noise files into one second and then shifting them by 0.2 second between each sample (having partly similar sound to its neighboring sample) to generate more examples. The reason to generate more examples is to have similar data distribution across all classes as every class in Speech Commands Dataset has around two thousand samples. If we only split the dataset into 1 second unique samples, we will only end up having 400 silence samples. In the end, the silence samples were divided into the same proportion (80% training data , 10% validation data, and 10% test data). The details of the dataset for this experiment is shown in Table 1.

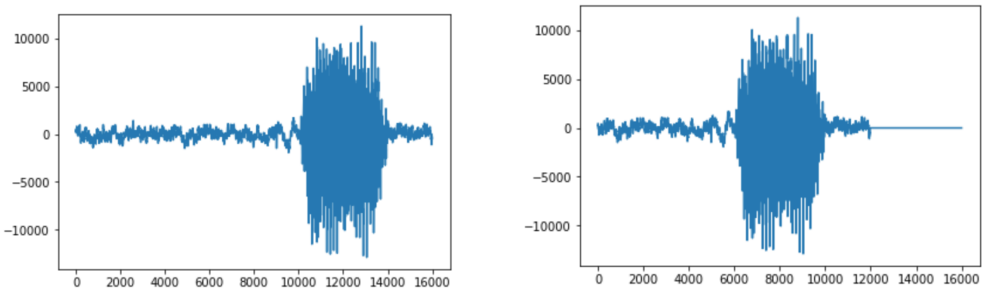
Table 1: Dataset Distribution

Data	Training	Validation	Test
1 second samples	44,517	5,829	6,172
$\leq$1 second samples	6,571	969	663
Silence samples	1,598	201	199
Total	52,686	6,999	7,034

2.2 Sound Alignment

We aligned the dataset to see if the sound alignment can help in improving the performance. This operation is inspired by Zhang et al's [13] work for face classification. They aligned the faces into the center of the image which helped in reducing the error rate for face classification. However, our method is much simpler as we take a WAVE file and align the utterance to the center of the audio file. Initially, we find the maximum amplitude (on Y-axis) of the utterance from the file. After that, we select the starting and ending points of the utterance by finding the earliest and latest samples having amplitude greater than 25% of the utterance's maximum amplitude respectively.

For example, the figure 1a shows the original sound file, to align this utterance we find the maximum amplitude which is around 10000 (on Y-axis) and then compare each sample's amplitude with 2500 (25% of 10000). So the starting point of this utterance is roughly at index 10000 (on X-axis) and the ending point is at 14000. According to these points, the center of the utterance is at index 12000 which is then shifted to the center of the sound file (at 8000). At the end, we pad the empty samples by zeros (from 12000 to 16000) as shown in figure 1b.



(a) Sound samples before alignment

(b) Sound samples after alignment

Figure 1: Plot of sound alignment to center the main utterance

2.3 Feature Extraction

Feature extraction is one of the most important tasks in ASR and neural network training pipeline. We apply liftered Mel-Frequency Cepstral Coefficients (LMFCC) on the raw speech signal (the WAVE audio files) to extract the features as shown in figure 2. An audio signal is constantly changing, so to simplify things we assume that on short time scales the audio signal does not change much, which allows us to enframe the speech sound by setting a window of 25 ms and shifted it by 12.5 ms. These frames of signals are processed by the pre-emphasis module, then hamming window is applied to get the smooth transition from each repetition of the signal (so that it looks more like an infinite signal and less like one finite signal repeated infinite times) which are then further processed by Fast Fourier Transformation (with length of 512 samples) and Mel Filterbank. Once we have the filterbank energies, we take logarithm which allows us to use cepstral mean subtraction, which is a channel normalisation technique. The output from the logarithm function is then transformed and liftered which gives us LMFCC features. Since the speech signals have 16 kHz sampling frequency, we have 78 time frames for each utterance. We calculate liftered mel-frequency cepstral coefficient (LMFCC) with 13 cepstrum coefficients to extract the spectral feature of the speech data. We use 13 coefficients because if we add more coefficients, the information gain is not significant [9]. The input data becomes two dimensional after all this processing with 13×78 feature size.

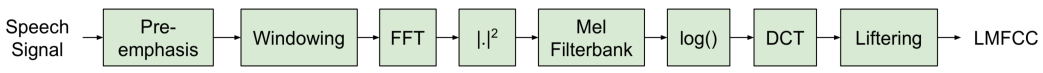


Figure 2: Extracting LMFCC feature

2.4 Deep Neural Network (DNN)

We firstly build a fully connected feed-forward network with a categorical cross entropy loss on output layer as shown in the figure 3. This simple neural network has 4 hidden layers (256 nodes each) with the input data representation flattened into $13 \times 78 = 1014$ one-dimensional features. For all the 4 hidden layers, Rectified linear unit (ReLU) is used as an activation function for computing reduction, the weighted sum of the output from previous layer and SGD as an optimizer.

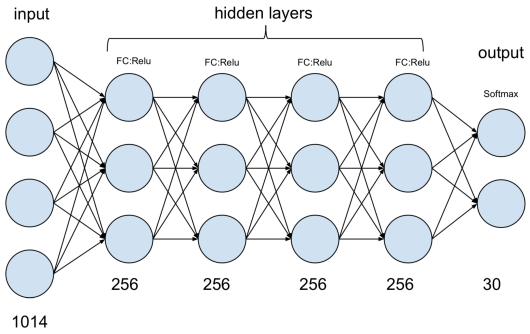


Figure 3: Feed-forward fully connected neural network

2.5 Convolutional Neural Network

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have been proven to be very effective in areas such as image recognition and classification[10]. We choose to implement CNN because convolutional layers in these networks are suitable to detect local features that may appear anywhere in the input. The spectrum representation of audio signals has very strong correlations in time and frequency so modeling these local correlations with CNNs would be beneficial. By implementing CNN, we expect to have much better performance than deep neural networks. One more advantage of implementing CNNs is due to the parameter sharing quality, they have fewer parameters compared to DNNs for the same task which could potentially reduce the memory footprint and computational requirement. The filters in CNN are able to extract the relation between each audio frame and its neighbour.

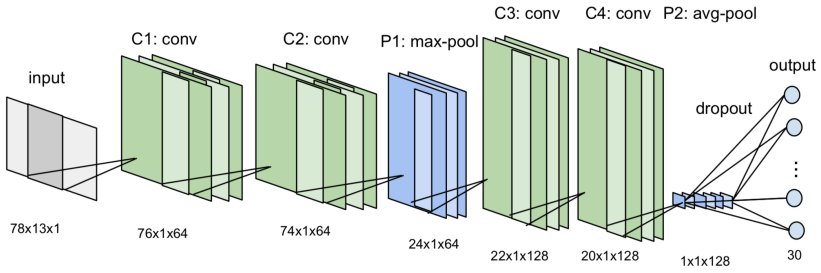


Figure 4: Convolutional neural network architecture

The implemented convolution neural network consists of 4 convolutional layers as shown in the figure 4. We used one dimensional convolutional window as LMFCC has low correlation among each feature [11]. In principle, one dimensional convolutional filter is equivalent to a column of a two dimensional convolutional filter. Max pooling layer is used after convolutional layer to reduce the number of parameter in the network, computation complexity, and to control overfitting. For similar reasons, average pooling layer and dropout is used after the last convolutional layer(C4) rather than fully connected layer which reduces the network parameters and avoid overfitting [12].

3 Experiments and Result

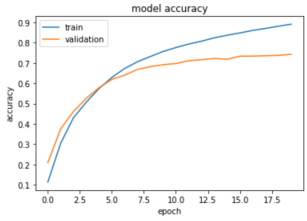
As we have implemented two different architectures, we divided our experiments into two sections. Section 3.1 describes the experiments of deep neural network architecture and section 3.2 describes the experiments of convolutional neural network. The experiments performed on these models are compared with the results reported by Li et al [6]. As Li et al's [6] experiments only classify six classes whereas we trained the network on 31 classes, we consider the commands other than 'silence', up', 'down', 'left', and 'right' into 'unknown' class. Another comparison was made on the basis of Kaggle's competition [14] which has two times larger test dataset as compared to Google's Speech Commands dataset. In Kaggle's competition, the model has to classify 12 commands ('unknown', 'silence', 'yes', 'no', 'up', 'down', 'left', 'right', 'on', 'off', 'stop', and 'go') rather than 31. We followed the same approach to classify the unknown class as we did for 6 class classification. The results on Kaggle's test set would be more reliable due to the larger size of test dataset.

Since the 'unknown' class has more examples than the other classes, we calculated six accuracy measures from the networks which are the following:

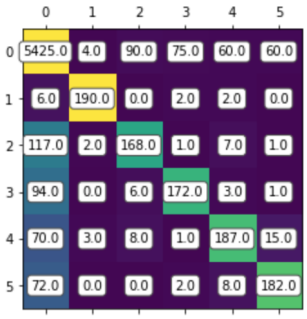
- Test accuracy on 31 classes
- Test accuracy on 6 classes including 'unknown'
- Test accuracy on 5 classes excluding 'unknown'
- Test accuracy on 12 classes including 'unknown'
- Test accuracy on 11 classes excluding 'unknown'
- Kaggle's test accuracy on 12 classes using Kaggle's test dataset

3.1 Deep Neural Network Experiments

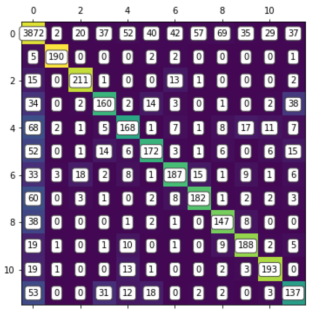
We performed initial experiments using fully connected feed-forward neural network architecture with four hidden layers. The first step is to standardize the data by normalizing it over training set. The network consists of 4 hidden layers with 256 nodes and was trained by using 256 batch size, adam optimizer and ReLU as the activation function. It was trained on stochastic gradient descent (SGD) with 0.002 learning rate and softmax classifier (cross entropy) as a loss function. The input size of the network is 1014 and the output consists of 31 speech command labels.



(a) Training Accuracy plot



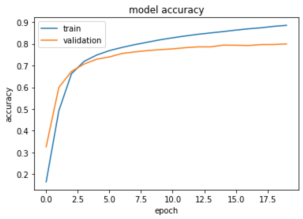
(b) Confusion matrix for 6 labels



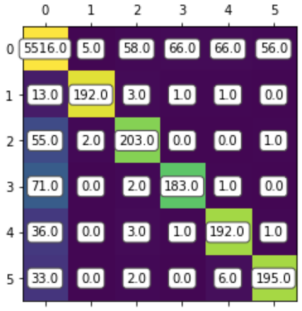
(c) Confusion matrix for 12 labels

Figure 5: Plot accuracy during training and confusion matrix of fully connected DNN without alignment

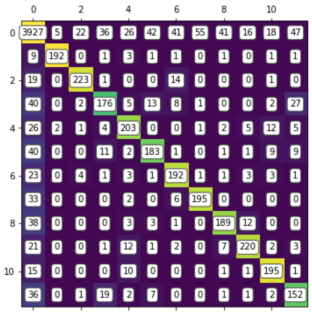
By using non-aligned dataset, the network achieved 72.04% test accuracy on 31 classes, 89.90% on 6 classes, and 83.40% on 12 classes. The accuracy graph of this experiment is shown in figures 5a. The confusion matrices for 6 labels and 12 labels classification are shown in figures 7b and 7c. In these confusion matrices, X-axis corresponds to the prediction and Y-axis corresponds to the actual label. 0:'unknown', 1:'silence', 2:'up', 3:'down', 4:'left', and 5:'right' are the labels representation with respect to the indexes in figure 7b. Whereas, the 12 labels in figure 7c are 0:'unknown', 1:'silence', 2:'yes', 3:'no', 4:'up', 5:'down', 6:'left', 7:'right', 8:'on', 9:'off', 10:'stop', 11:'go'. These index mappings are also applied in the other experiments.



(a) Training Accuracy plot



(b) Confusion matrix for 6 labels



(c) Confusion matrix for 12 labels

Figure 6: Plot accuracy during training and confusion matrix of fully connected DNN with alignment

We implemented sound alignment for centering the utterance. In figure 6a, we can see that the alignment increased the performance of classification using fully connected deep neural network. It achieved 78.63% test accuracy for 31 classes, 93.06% on 6 classes, and 89.04% on 12 classes. In the later section, we will compare these results with the results of CNN.

3.2 Convolutional Neural Network Experiments

Convolutional neural network was implemented to maintain the time series properties of the dataset. Initially, CNN was trained on the best performing hyperparameters settings of deep neural network (data normalization, 256 batch size, input size 78×13 and softmax loss function). The filters size was kept to 3 because we wanted to extract the relationship of each sample with its immediate neighbor. The motivation of selecting a smaller filter size is that the transition of voice features may vary depending on how fast an individual can speak and this variation might not be successfully handled with a large filter size. As 1D convolutional filters were used, we kept the stride equals to 1. Table 2 shows the number of parameters, output, number of filters, filter size and stride settings of each layer in the architecture.

Table 2: Parameter of CNN network

Type	output size	filter width	no. of filter	stride	parameter
C1: Conv1	$76 \times 1 \times 64$	3	64	1	2,560
C2: Conv2	$74 \times 1 \times 64$	3	64	1	12,352
P1: Max-pool	$24 \times 1 \times 64$	3	0	3	0
C3: Conv3	$22 \times 1 \times 128$	3	64	1	24,704
C4: Conv4	$20 \times 1 \times 128$	3	64	1	49,280
P2: Global avg-pool	$1 \times 1 \times 128$	20	0	1	0
D1: Dropout	$1 \times 1 \times 128$	0	0	0	0
O: Softmax	$1 \times 1 \times 30$	0	0	0	3,870
Total					92,766

The network was trained on RMSProp optimizer (an adaptive learning rate method proposed by Hinton[15]) which stores an exponentially decaying average of the square of the gradient vector:

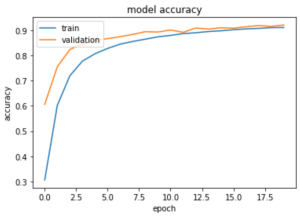
$$E[g_t^2] = \gamma E[g_{t-1}^2] + (1-\gamma) g_t^2 \tag{1}$$

with update rule as:

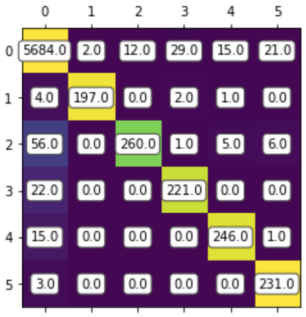
$$x^{(t)} = x^{(t-1)} - \frac{\eta}{\sqrt{E[g_t^2] + \epsilon}} g_t \tag{2}$$

We use the parameter as Hinton suggested: $\eta = 0.001$ and $\gamma = 0.9$ [16].

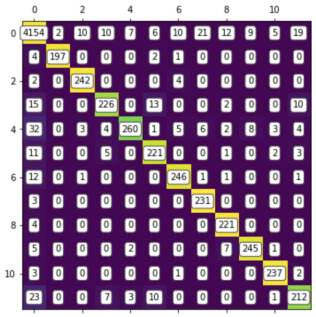
We started the CNN experiment by using the dataset without alignment. After exploration search of hyperparameters, the best performing CNN model scored 91.77% test accuracy on 31 classes, 97.23% on 6 classes and 95.14% on 12 classes. The network was trained on 30 epochs using softmax as a



(a) Training Accuracy plot



(b) Confusion matrix for 6 labels

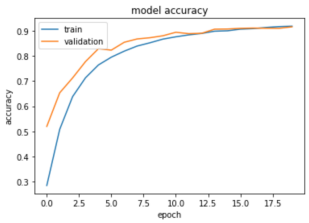


(c) Confusion matrix for 12 labels

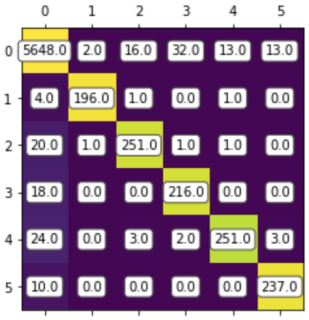
Figure 7: Plot accuracy during training and confusion matrix of CNN without alignment

loss function, ReLU as an activation, rmsprop as an optimizer and 50% dropout. The accuracy plot of this network is shown in figure 7a whereas the confusion matrices of 6 and 12 labels are shown in 7b and 7c respectively.

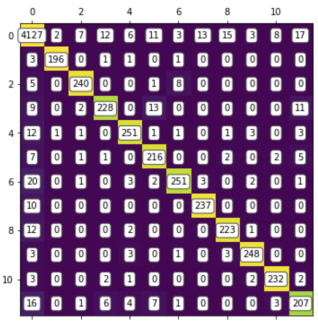
While using sound-aligned dataset, the network performed almost similarly to the non-aligned dataset's experiment. It achieved 92.38% test set accuracy on 31 classes, 97.63% on 6 classes and 95.58% on 12 classes.



(a) Training Accuracy plot



(b) Confusion matrix for 6 labels



(c) Confusion matrix for 12 labels

Figure 8: Plot accuracy during training and confusion matrix of CNN with alignment

The CNN showed in figure 4 outperformed previously implemented feed forward network. This network also outperformed the results reported in Li et al. [6] in 6 labels classification. Their network achieved 95.1% on test accuracy for 6 classes. Moreover, the network used by Li et al. [6] has 165% more parameters as compared to our model (244,400 vs 92,766 parameters).

3.3 Evaluation

All results from both architectures are shown in table 3. The results on Kaggle's dataset are comparatively lower than the Google's dataset which is reasonable because Kaggle's dataset has two times larger test set. The sound alignment improved the performance of DNN whereas CNN showed marginal improvement as convolutional filters in CNN can detect important features regardless of their positions. Additionally, we used the whole dataset to train the network and achieved 78.55% on Kaggle's dataset.

4 Discussion and Conclusion

In the initial experiments using feed forward fully connected network architecture, we compared the performance of the network on classification of non-aligned and aligned speech commands. Classification using aligned speech commands performs better as compared to the non-aligned dataset because the alignment reduces the variability of the sound so each speech command starts and ends

Table 3: Overall Result

	DNN		CNN	
Test accuracy on	not aligned	aligned	not aligned	aligned
31 classes	72.06%	78.63%	91.77%	92.38%
6 classes including 'unknown'	89.91%	93.06%	97.28%	97.63%
5 classes excluding 'unknown'	93.91%	97.54%	98.81%	98.96%
12 classes including 'unknown'	82.56%	86.83%	95.14%	95.58%
11 classes excluding 'unknown'	83.40%	89.04%	95.95%	95.88%
12 classes via Kaggle	60.07%	65.63%	75.50%	76.90%

with silence. Whereas, the speech recognition using fully connected network did not perform better than CNN. The main reason behind the lower performance of DNN is that it loses the time series properties of the data when we flatten the input feature of the speech commands.

While comparing the performance of convolutional neural network (CNN) and fully connected feed-forward neural network, CNN performs better than fully connected network. CNN maintains the time series properties of speech signals as the input size of the network is 73×13. We also found out that one-dimensional convolutional filters perform as well as two dimensional convolutional filters in Automatic Speech Recognition. It is more efficient to use one-dimensional convolutional filters for speech recognition tasks while using LMFCC features because we get a decent performance by reducing large number of parameters as compared to the two-dimensional convolutional filters.

Since this experiment only works with the speech samples with the duration of one second, the application in the real world is limited. One useful application of this model could be to recognize human commands (less than or equal to one second) to control devices or gadgets. We can also use this model by sampling a continuous speech of one second duration with 250 ms shift in each sample. The prediction of these samples can be merged if they are similar to their neighbors.

References

[1] Mishra, N., Shrawankar, U., & Thakare, V. M. (2013) Automatic Speech Recognition Using Template Model for Man-Machine Interface. *arXiv:1305.2959 [cs.SD]*, [Online]. Available: (`https://arxiv.org/pdf/1305.2959.pdf`)

[2] Kotsiantis, S. B. (2007) Supervised Machine Learning: A Review of Classification Techniques *Informatica, vol. 31*, pp. 249–268, [Online]. Available: (`https://datajobs.com/data-science-repo/Supervised-Learning-[SB-Kotsiantis].pdf`)

[3] Wilpon, J. G., Rabiner, L. R., Lee, C. H., & Goldman, E. R. (1990) Automatic Recognition of Keywords in Unconstrained Speech Using Hidden Markov Models. *IEEE Transactions on Acoustics, Speech, and Signal Processing, 38(11):1870–1878, 1990*, [Online]. Available: (`https://www.researchgate.net/profile/Chin-Hui_Lee/publication/3175916_Automatic_recognition_of_keywords_in_unconstrained_speech_using_hidden_Markov_models/links/549405180cf240d1cb4d235a/Automatic-recognition-of-keywords-in-unconstrained-speech-using-hidden-Markov-models.pdf`)

[4] Fernández, S., Graves, A., & Schmidhuber, J. (2007) An application of recurrent neural networks to discriminative keyword spotting, *Artificial Neural Networks–ICANN 2007*, pp. 220–229. Springer, [Online]. Available: (`https://www.cs.toronto.edu/~graves/icann_santi_2007.pdf`)

[5] Chen, Guoguo, Parada, C., & Heigold, G. (2014) Small-footprint keyword spotting using deep neural networks.*IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014*, [Online]. Available: (`https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42537.pdf`)

[6] Li, X. & Zhou, Z.. (2017) Speech Command Recognition with Convolutional Neural Network. *CS229: Machine Learning, Autumn 2017*, [Online]. Available: (`http://cs229.stanford.edu/proj2017/final-reports/5244201.pdf`)

[7] Hinton, G.E., Osindero, S. & Teh Y.W. (2006) A fast learning algorithm for deep belief nets.*Neural computation*, 18(7):1527–1554, [Online]. Available: (http://www.cs.toronto.edu/~fritz/absps/ncfast.pdf)

[8] "Launching the Speech Commands Dataset." *AI Blog*, 24 Aug. 2017, [Online]. Available: (ai.googleblog.com/2017/08/launching-speech-commands-dataset.html).

[9] Gupta1, S., Jaafar, J., Ahmad, W. F., & Bansal, A. (2013) Feature Extraction Using MFCC *Signal & Image Processing : An International Journal (SIPIJ) Vol.4, No.4, August 2013*, [Online]. Available: (http://aircconline.com/sipij/V4N4/4413sipij08.pdf)

[10] Krizhevsky, A., Sutskever, I., & Hinton, G. E. () ImageNet Classification with Deep Convolutional Neural Networks, *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems* vol.1:1097-1105 ,[Online]. Available: (https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf)

[11] Narayana, L., & Kopparapu, S. K.(2014) Choice of Mel Filter Bank in Computing MFCC of a Resampled Speech. *arXiv:1410.6903 [cs.SD]*, [Online]. Available: (https://arxiv.org/pdf/1410.6903.pdf)

[12] Lin, M., Chen, Q & Yan, S. (2014) Network In Network, *CoRR, vol. abs/1402.1128, 2014.*, [Online]. Available: (https://arxiv.org/pdf/1312.4400.pdf)

[13] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016) Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks *arXiv:1604.02878 [cs.CV] 11 Apr 2016*, [Online]. Available: (https://arxiv.org/pdf/1604.02878.pdf)

[14] "TensorFlow Speech Recognition Challenge | Kaggle." (2017) *Countries of the World | Kaggle*, [Online]. Available: (www.kaggle.com/c/tensorflow-speech-recognition-challenge).

[15] G. Hinton, Srivastava, N., & Swersky, K. (2012) Neural Networks for Machine Learning *Lecture 6a rmsprop: Divide the gradient by a running average of its recent magnitude*, [Online]. Available: (https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf)

[16] Ruder, S. (2017) An overview of gradient descent optimization algorithms *CoRR, vol. abs/1609.04747v2, 2017*, [Online]. Available: (https://arxiv.org/pdf/1609.04747.pdf)