Building a Low Latency Keyword Spotting System

Ritika Gupta, Shreyans Jasoriya, Mohini Limbodia
{gupta.ritika, jasoriya.s, limbodia.m} @husky.neu.edu

1 Introduction

Recent advancements in technology has opened doors for an immense number of applications where human-machine interaction is required. This idea of interaction has advanced significantly in the past decade in Automatic Speech Recognition (ASR). Speech Recognition serves as the primary mode of communication among mammals. Speech Recognition makes use of the process and technology for converting speech signals into a sequence of words. ASR is a complex process that involves audio sampling, feature extraction and speech recognition to recognise individual sounds and convert them to text. ASR uses keyword spotting (KWS) which is a critical component for enabling speech based user interactions on smart devices. It requires real-time response and high accuracy for good user experience.

We will use the Tensorflow Speech Commands dataset [1] to build an algorithm that understands simple spoken commands to identify the keywords. This dataset was developed to detect when a single word is spoken, from a set of ten or fewer targets. This has to be achieved by limiting the false positive rate from background noise or unrelated speech to as low as possible. This task is known as Keyword Spotting. By improving the recognition accuracy of voice interface tools, we can improve product effectiveness and their accessibility in all speech recognition applications.

Input- One second audio clips that contain one out of the 30 keywords including silence and background noise, spoken by thousands of different people.

Output- The models will be trained to classify the incoming audio into one of the 10 keywords- Yes, No, Up, Down, Left, Right, On, Off, Stop, Go, along with silence(i.e. no word spoken) and unknown word, which is the remaining 20 keywords from the dataset.

Example: if there is an audio clip that has someone saying ON, then the 1-second audio clip is the input and the output is the word ON.

2 Related Work

Research in the field of Keyword Spotting with Deep Learning began with the usage of Deep Neural Networks [9] wherein standard feed-forward fully connected neural networks with rectified linear unit (ReLU) activation functions are trained using transfer learning by initializing the acoustic model network with a network trained on general speech data. The research shifted to Convolutional Neural Networks (CNN) [3][2] to preserve the local temporal and spectral correlation in the input speech features. Using these recent advances, Convolutional Recurrent Neural Networks (CRNN) was built [10] to utilize the context for the entire processed frame of the speech data. Experiments were done with Long Short-Term Memory (LSTM) with max-pooling loss [11] that yielded better accuracy in comparison to DNN.

As noted in Hello Edge [2], the above models were trained on different datasets resulting in different input speech features and speech duration. The work done in [2] is an effort in that direction to consolidate the results of these models [9][3][10][11] by testing them on the Tensorflow Speech Commands dataset [1]. Additionally, the authors have taken inspiration from the MobileNet architecture [12] to develop their own Depthwise Separable Convolutional Neural Network (DS-CNN). They were able to state of the art 95.4% multiclass accuracy. All the models in the Hello Edge [2] are trained on the features extracted using Mel-frequency cepstral coefficients (MFCC).

An innovative approach to feature engineering for the task of keyword spotting is done in [4] where spectrogram images are created using the audio clips of the Tensorflow Speech Commands dataset

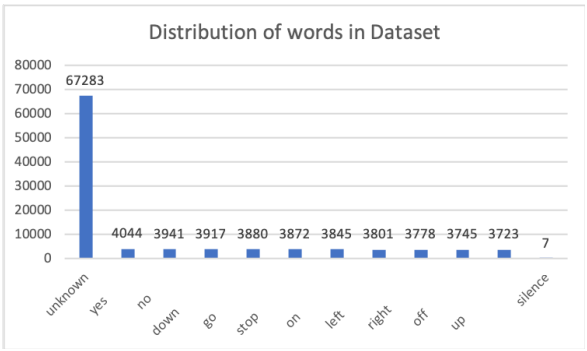

Figure 1: Distribution of words

[1]. This converts the problem into an image recognition task rather than speech recognition. Their CNN implementation was able to achieve a 92.6% accuracy on the test set.

3 Methodology

3.1 Data and features

Dataset: The dataset we have taken is Open Source data by TensorFlow. It consists of 1-second audio clips each containing of only one out of 30 keywords and is recorded by thousands of different people. There are 30 keywords in both the training and the test set, out of which our task is to recognize the 10 wake words. The 10 Wake words we are identifying are - 'Yes', 'No', 'Up', 'Down', 'Left', 'Right', 'On', 'Off', 'Stop', 'Go'. The other 20 keywords in the dataset are 'Bed', 'Bird', 'Cat', 'Dog', 'Eight', "Five", "Four", 'Happy', 'House, 'Marvin', 'Nine', 'One', 'Seven', Sheila', 'Six', 'Three', 'Tree', 'Two', 'Wow', 'Zero'. These 20 keywords are labelled as 'Unknown'. There are also some audio clips containing background noise labelled as 'Silence'. Thus, there are 12 possible labels for the model. The distribution of the words in the dataset is shown in Figure 1. As shown, the dataset has a very high proportion of 'Unknown' words, which represents the real world circumstances.

The dataset is divided into train, validation and test set as follows:

Total dataset size	105k audio clips
Train Set	76k audio clips
Validation Set	19K audio clips
Test Set	11k audio clips

Table 1: Dataset sizes

3.2 Data Preprocessing

The preprocessing steps for every methodology implemented are different. The features we have used are MFCC features, Log-mel filterbank energies and spectrogram images of audio clips:

1. Log-mel filter bank energy (LFBE) features: A filter bank is an array of band-pass filters that separates the input signal into multiple components, each one carrying a single frequency sub-band of the original signal. A signal goes through a pre-emphasis filter; then gets sliced into (overlapping) frames and a window function is applied to each frame; afterwards, we do a Fourier transform on each frame (or more specifically a Short-Time Fourier Transform) and calculate the power spectrum; and subsequently compute the filter banks. Then we take the log of the filter banks.

   The audio clip is read in mono channel and then the downsampled by a ratio of 3. Then the LFBE feature is calculated. The feature is padded or clipped if required, so that we get a consistent feature of shape (32, 26).

2. MFCC features: It turns out that filter bank coefficients are highly correlated, which could be problematic in some machine learning algorithms like Hidden Markov Models (HMM). Therefore, we apply Discrete Cosine Transform (DCT) to decorrelate the filter bank coefficients and yield a compressed representation of the filter banks. Typically, for Automatic Speech Recognition (ASR), the resulting cepstral coefficients 2-13 are retained and the rest are discarded. The reasons for discarding the other coefficients is that they represent fast changes in the filter bank coefficients and these fine details dont contribute to Automatic Speech Recognition (ASR).

   The audio clip is read in mono channel and then the downsampled by a ratio of 3. Then the MFCC feature is calculated. The feature is padded or clipped if required, so that we get a consistent feature of shape (20, 44).

3. Spectrogram: A spectrogram is a visual way of representing the signal strength, or loudness, of a signal over time at various frequencies present in a particular waveform. Not

only can one see whether there is more or less energy at, for example, 2 Hz vs 10 Hz, but one can also see how energy levels vary over time.

Spectrograms are basically two-dimensional graphs, with a third dimension represented by colors. Time runs from left (oldest) to right (youngest) along the horizontal axis. The vertical axis represents frequency, which can also be thought of as pitch or tone, with the lowest frequencies at the bottom and the highest frequencies at the top. The amplitude (or energy or loudness) of a particular frequency at a particular time is represented by the third dimension, color, with dark blues corresponding to low amplitudes and brighter colors up through red corresponding to progressively stronger (or louder) amplitudes.

The log spectrogram image of the file is computed on reading the images. And then the log spectrogram image is normalized by dividing the pixels by 255 and finally we zero center the images by mean pixel of the channel.

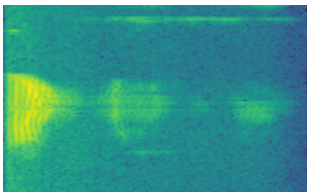

Figure 2: Spectogram of word 'Right'

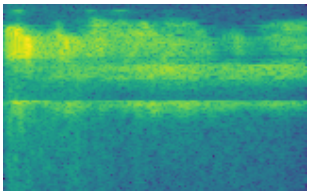

Figure 3: Spectogram of word 'Left'

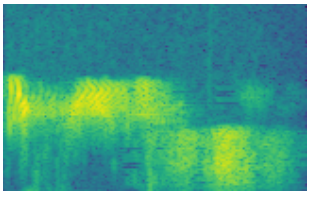

Figure 4: Spectogram of word 'Yes'

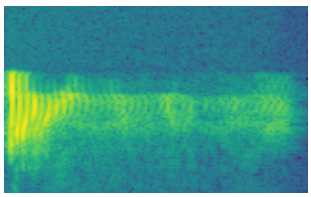

Figure 5: Spectogram of word 'No'

3.3 Implementation

We tried the following experiments for our implementation of a Keyword Spotting System:

3.3.1 CNN

All the work done [2][3] on this dataset [1] have utilized MFCC features. MFCC features became popular when Gaussian Mixture Models - Hidden Markov Models (GMMs-HMMs) were very popular. GMM-HMMs could not perform well on LFBE features since the filterbank coefficients were correlated. Since, deep learning algorithms are not constrained by this, and to avoid the information loss during MFCC computation we will also use LFBE features.
The architecture of our CNN is shown in Figure 6. We have trained our CNN model on both MFCC and LFBE features. We would like to compare the accuracy obtained using MFCC features with the accuracy when the features are extracted using Log-mel filterbank energies (LFBE).

3.3.2 Depthwise Separable CNN

The architecture of our DS-CNN is shown in Figure 7. We have trained our DS-CNN model on both MFCC and LFBE features. We would like to compare the accuracy obtained using MFCC features with the accuracy when the features are extracted using Log-mel filterbank energies (LFBE).

3.3.3 CNN with Transfer Learning

Feature engineering of converting the speech data into spectrogram images as done in [4] is an intuitive way of visualizing the input data. We have chosen the Inception-ResnetV2 [7] pretrained on ImageNet datset [11] for transfer learning. ImageNet is a large

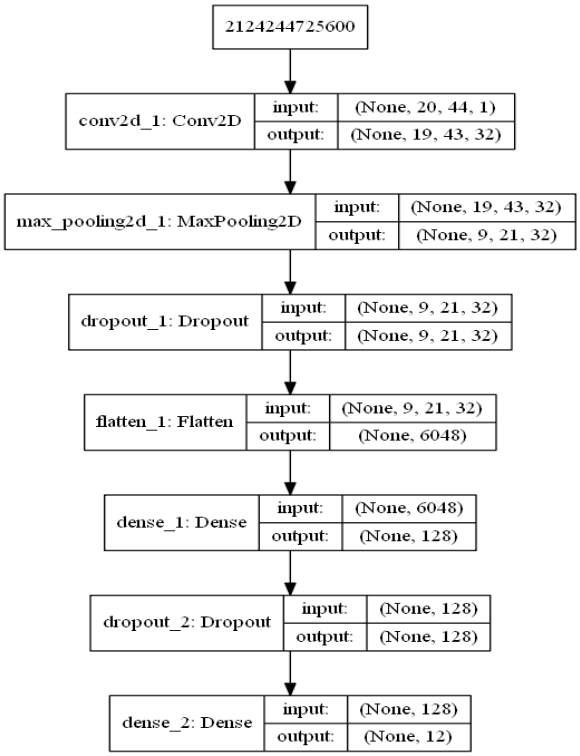

Figure 6: CNN

visual dataset containing more than 14 million images of 20,000 different categories. We fix the weights of the first 32 hidden layers in this pretrained Inception-ResnetV2 [7] model. These fixed weights represent an internal representation that works well on any image. Then, we combine the Inception-ResnetV2 model in combination with some other layers as shown in Figure 8. This custom CNN architecture is then trained using the spectrogram images. We tried this model by reshaping the images to a size of both (139, 139) and (299, 299) and observed that we got higher accuracy on the image size (299, 299) as their was less information loss. The architecture of this custom CNN architecture is shown in Figure 8.

4 Experiments and Results

Multiclass accuracy is used as an evaluation metric. The DS-CNN model is the baseline model.

1. CNN
   Number of layers: 7
   Activation function: ReLU & Softmax
   Optimizer: Adadelta
   Loss: Categorical Cross Entropy

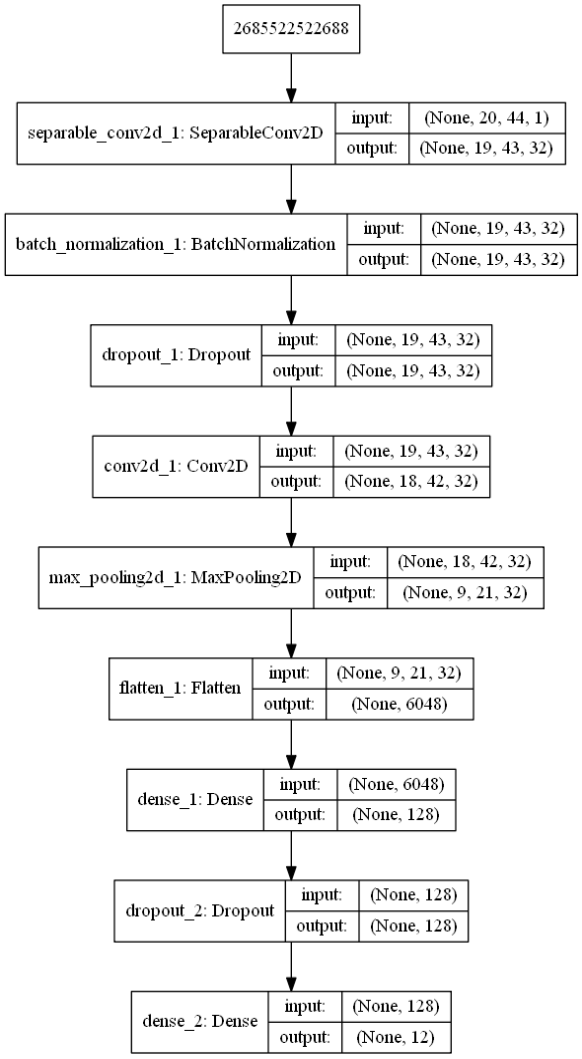

Figure 7: DSCNN model architecture

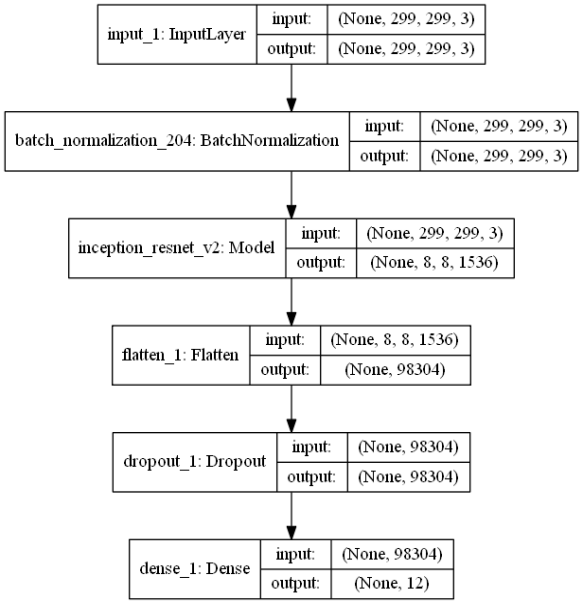

Figure 8: CNN with Transfer learning

Epoch = 200 (Saved model with best val_acc)

2. DS-CNN
   Number of layers: 9
   Activation function: ReLU & Softmax
   Optimizer: Adadelta
   Loss: Categorical Cross Entropy
   Epoch = 500 (Saved model with best val_acc)

3. CNN with Transfer Learning
   Number of layers: 471
   Activation function: ReLU & Softmax
   Optimizer: Adam
   Loss: Categorical Cross Entropy
   Epoch = 5 (Saved model with best val_acc)

Models	Train Acc	Test Acc
CNN (MFCC)	83.82%	82.74%
CNN (LFBE)	90.79%	87.29%
DSCNN (MFCC)	87.54%	81.44%
DSCNN (LFBE)	94.55%	84.56%
CNN Transfer Learning (Spectogram)	99.54%	93.77%

Table 2: Results

5 Conclusion

1. The CNN and DS-CNN results clearly show that on the same model architecture, LFBE features are better suited for deep learning algorithms then MFCC.

2. Transfer Learning techniques are very helpful in boosting the accuracy of a simple model architecture. Using the same we can see a major improvement in accuracy as compared to 92.6% obtained in [4].

6 Future Work

Wavenet [12] is a state of the art model for raw audio. We would like to use Wavenet to explore transfer learning with MFCC and LFBE features.

7 References

1. Warden, Pete. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. arXiv:1804.03209 (2018).

2. Zhang, Yundong, et al. Hello edge: Keyword spotting on microcontrollers. arXiv preprint arXiv:1711.07128 (2018).

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

4. Gouda, Sanjay Krishna, et al. Speech Recognition: Keyword Spotting Through Image Recognition. arXiv preprint arXiv:1803.03759 (2018).

5. Domingos, Pedro. A few useful things to know about machine learning. Communications of the ACM 55.10 (2012): 78-87.

6. Bauer, Eric, and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine learning 36.1-2 (1999): 105-139.

7. Szegedy, Christian, et al. Inception-v4, inception-resnet and the impact of residual connections on learning. AAAI. Vol. 4. 2017.

8. Sandler, Mark, et al. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, 2018.

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

10. Arik, Sercan, et al. Convolutional recurrent neural networks for small-footprint keyword spotting. U.S. Patent Application No. 15/688,221.

11. Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009.

12. Van Den Oord, Aron, et al. "WaveNet: A generative model for raw audio." SSW 125 (2016).