SPEECH RECOGNITION CHALLENGE

Harika Putti, Pavan Kumar Madineni, Priyadarshini Vijjigiri

Indiana University Bloomington

ABSTRACT

Artificial intelligence systems are prevalent everywhere and have become an integral part of our lives today. The main mode of communication with these intelligence systems has shifted from text to speech. It has become customary for users to rely on virtual assistants for mundane tasks. These systems have grown rapidly from just understanding human speech to being able to reciprocate. Though we may still be at least a few years away from having a fully autonomous intelligent artificial system that can genuinely communicate with us in a human-like way, deep inroads are being made in building robust automatic speech recognizing systems [9]. A lot of ASR systems have been integrated into home appliances to make lives easier, but these function only in the environment they are designed to. We propose to build an ASR system that can identify a set of words accurately even when recorded in a noisy environment and also be able to deal with speaker/accent variability. In this paper, we dealt with this problem using Deep Learning Methods like CNNs and RNNs using Gated Recurrent Units and applied novel data engineering techniques on the input time domain signals before we fed them to our models.

Keywords— Speech Recognition, Kaggle, CNN, GRU, Voice Activity Detection, MFCC

1. INTRODUCTION

With the advent of exploding technology, a lot of technological objects are invented each day. Enhanced versions of these objects can be seamlessly re-invented each day using abundantly available open source technological repositories. But the same is not the case with building a speech detector using free, open data and code. A simple speech recognizer would require a lot of preprocessing using varied voice recognition datasets before a neural network can be trained and built on it. In order to fill the vacuum in innovation in this space, TensorFlow/Google Brain has released a speech commands dataset that can be used to train a neural network and subsequently build a speech detector.

Speech recognition systems are not recent innovations but have been around since 1950s. The initial speech recognition systems were very confined in their scope such as their limitation to a single speaker and sparse vocabulary. The speech recognition systems today are very advanced mainly due to the progress made using Hidden Markov Models [4]. But these speech recognition systems have been commercialized and are able to create huge impact on personal lives primarily due to the advancements in Neural Networks and Deep Learning.

2. PRIOR WORKS

2.1. Google's Baseline Architecture

This is the most primitive architecture used for speech recognition. This is also the baseline architecture on top of which our approach is built. This architecture is inspired from the work of Tara.N.Sainath, Carolina Parada published under the name of Convolutional Neural Networks for small-footprint Keyword Spotting. This architecture was chosen primarily since it is simple, swift to train and easy to understand despite not being state of the art. In-general there are numerous approaches to building neural network models to work with audio signals such as using Recurrent neural networks or convolutional neural networks. This architecture is very much similar to any generic convolutional neural network that is used for image recognition. This can be immensely confounding as to how a convolutional neural network is applied to an audio signal [1] [3]. This perplexity arises due to the fact that an audio signal in its intrinsic nature is just a one-dimensional continuous signal across time and does not possess a two-dimensional spatial structure.

This confusion is addressed by converting the one-dimensional continuous signal into a two-dimensional array of numbers. This is achieved through a systematic process that involves defining a window of time and converting the audio signal in that window into an image. The input audio signals are split and grouped into short segments which are probably just a few milliseconds long. The strength of all the frequencies across these short time segments is computed. Each individual set of frequency strengths accumulated over all the set of bands is treated as a vector of numbers and these vectors are stacked and arranged in time order to form a two-dimensional array [6]. This double dimensional array which possess the spatial structure similar to that of an image can be treated like a single-channel image and is also known as the spectrogram.

2.2. Voice Activity Detection

Voice activity detection (VAD) is the research work done by Tom Backstrom at Aalto University in October 2015. The main objective of this work is to identify whether an audio signal contains speech or not. Voice activity detection is used as a pre-processing technique for most of the speech processing methods. VAD is primarily applied in two areas, they are speech recognition and speech enhancement [8]. Speech recognition is used to find out which parts of the signal should be fed to the recognition network since recognition is a computationally exhaustive operation, avoiding non-speech parts can save a lot of resources. In speech enhancement, the main objective is to minimize or eliminate noise in a speech signal. Non-speech parts of a signal can be extremely helpful in learning characteristics of noise and then using these traits to remove noise from the speech signal. Identifying non-speech parts is extremely resourceful, especially in keyword spotting where the VADs objective is to avoid running the computationally expensive keyword spotting algorithm always.

2.3. Takeaway from prior works:

Our approach is inspired slightly from these two prior works. Among the 12 labels our data has, one is silence i.e. identifying if an audio signal contains any speech or not which is in-line with the model implemented using VAD. We have taken basic traits from the architectures of both the above-mentioned works. We have based our architecture primarily on the baseline network released by Google and made changes on top it.

3. DATA PREPROCESSING

3.1. Data Set

The dataset is acquired from tensorflow speech recognition challenge hosted by Kaggle. The primary goal of this challenge is to build a model that can identify simple speech commands which are recorded under challenging circumstances such as in noisy environments and spoken by people with varied English accents and dialects. The training set contains around 65,000 one-second long utterances of 30 short words by different people.

The idea is to build a model using these short audio signals as inputs and classifying them into right labels by performing image recognition on MFCC spectrograms and logarithm of mel spectrograms. The dataset basically has utterances of 30 different words, but we are trying to build a model that identifies only 10 words which are yes, no, up, down, left, right, on, off, stop and go. The classification model we are trying to build has 12 classes. The other two classes are silence and unknown. Any word which is not among the above-mentioned 10 short commands will be categorized as unknown. For the silence labels, the dataset has 6 noise files each of which is 44 seconds long. Since only 6 signals would make the data highly imbalanced, we have split them into 2900 signals each of 1 second duration. This led to a huge improvement in the overall classification accuracy as well as recall for silence category.

To train under different environmental conditions we have added noise to the dataset provided by the Kaggle. Signal to noise ratio of the noisy signals ranged from 4dB to 16dB.

3.2. Preprocessing:

It is somewhat easy for the human eye to decipher the phonemes on a raw waveform of the audio signal.

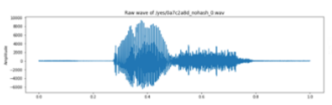

Figure : 1

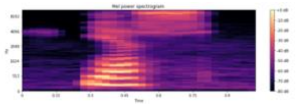

Figure : 2

The figure: 1 shows the waveform of a "Yes" audio sample. We can clearly delineate the different phonemes – *y e s* in it. While this is obvious to us, it's impossible for a neural network to process this information. The input that the models we are trying to implement require 2D signals or images. The standard practice for training any such kind of model on human audio data is to first extract spectrograms out of the raw audio signal. We have trained our models on these 2 different types of features

1. Firstly, we created normal spectrograms and we used MFCCs on our spectrograms as features and ran the same models on these features.
2. Secondly, we simply converted the signal to Mel spectrograms and took logarithm of this Mel spectrogram as features to input to our models one example for melspectogram is shown in figure 2.

We are comparing the models on these 2 types of features and are about to find the best type of features needed for this model. The important step is to extract spectrograms, this is absolutely imperative because the spectrogram gives an exact value of the intensity of each frequency at each instance in time by breaking the natural audio signal into its constituent waves at different frequencies. To convert audio signals to spectrograms we need to find the squared magnitude of the Short-time Fourier Transform (STFT) of the signal, for a customary window width. The result of converting the audio signals to spectrograms is a double dimensional array of numbers. These numbers are in a way a different representation of the originally imported audio signals. This process ensures conversion of audio files into images. These images can now be used to train any convolution-based model and implement different architectures. On top of this, another transformation called MFCC or Mel-Frequency Cepstral Coefficients is applied on these spectrograms. These transformations are employed to capture the unique characteristics of human speech. The MFCC captures the wrapper in which the voice band establishes itself. MFCCs features are generally the norm for speech recognition.

After training the models using the MFCC features we realized the approach is unsuitable. The major problem with MFCC is that it decorrelates the features of the input signal thus reducing the accuracies of our models. Neural networks generally perform better when mfcc's are not used because they can extract information even if they are correlated This is why we chose our next approach to just create a spectrogram using a Mel scale instead of the regular frequencies. Mel scale associates frequencies with those that are humanly perceivable. To convert regular frequencies on to Mel scale we use the formula [2]–

$$M(f) = 1125 \ln(1 + f/700)$$

After achieving the Mel spectrograms of the audio signals, we performed logarithm of the spectrograms and used that as the input features for our models. This proved to be a better approach since our accuracies for each model increased.

4. LITERATURE

4.1. Convolutional Neural Network

A convolutional Neural Network functions through interactions between three components, they are - the input image, the convolutional neural network and the output label or the class of the image. CNNs are predominantly used for image classification. In our project, the original audio signal is first transformed into an image called spectrogram, which is fed to the convolutional neural network. In general, any image is a collection of pixels where each pixel contains one byte of information which is generally a number between 0 and 255. Black and white images are two dimensional whereas colored images are three-dimensional. In the case of two-dimensional black and white images, each pixel is simply assigned a number between 0 and 255 but in the case of three-dimensional colored images, each pixel is represented on three levels since each pixel is a combination of red, green and blue at different levels of concentration.

The process of image detection using a CNN usually involves four stages, they are - convolution operation, Activation, Pooling and Flattening. The flattened output is then fed to fully connected artificial neural network to predict the probabilities of an image belonging to a particular class. Convolution operation when defined in pure mathematical terms is a function derived by the integration of two given functions which illustrate how the shape of function is modified by the other function. There are three entities that take part in convolution operation, they are - the input image in the form of pixel values between 0 and 255, feature detector otherwise called the filter or kernel and feature map also known as an activation map. Several feature maps are convolved over the input image with an optimal stride to generate corresponding feature maps. In general, larger the strides i.e. larger movement of the feature detector across the pixels, smaller is the feature map. The convolution operation is significant due to its ability to drastically reduce the size of the input image to the size of the desired activation map. The next obvious question is since there is a large reduction in size, is all the information in the input image preserved in the activation map. The feature map that is churned out of convolution operation definitely has fewer cells or pixels and therefore less information when compared to the original input image, but the whole purpose of a feature detector is to screen through the input image to capture only the essential features and discard the rest [7]. When the spectrogram images are fed to the CNN, the network determines features that are important to scan images and categorize them accordingly in terms of their priority. In most cases, the features identified by the network are inconspicuous to the human eye.

The next step is applying an activation function which is generally ReLU i.e. Rectified Linear Unit. This is not a separate component of the CNN's process but a supplementary step to the convolution operation which induces non-linearity into the network thereby enhancing the flexibility of the model further. The reason for implementing activation on the images is because images are inherently non-linear. The next step is pooling that enables the convolutional neural network to cope with issues associated with orientation of an image. Pooling is the process by which the CNN acquires the property of spatial variance which enables the CNN to detect all the intricacies of an image without being confused by different aspects of an image. In general, mean poling, max pooling and sum pooling are the pooling techniques used with max pooling being the most widely used technique. Pooling helps in minimizing the size of an image as well as the number of parameters which in turn prevent any chance of the model trying to overfit the data.

The last step in the CNN is flattening which, as the name suggests simply means to flatten the pooled feature map into a column which can then be directly fed to the artificial neural network. The flattened vector is simply the features hand-picked and distilled through the previous steps by the convolutional neural network. The role of Artificial neural network now is to combine these features into a wider variety of attributes to make the whole network capable of classifying images based on the probabilities that the network assigns to each class or label. The network simply votes among the neurons on which class among the 12 classes the input spectrogram will be attributed to. The next question that pops up is, since we have 12 classes, how do we coordinate their probabilities such that they add up to 1. This is taken care of by the SoftMax function or normalized exponential function that makes sure that the probabilities corresponding to each class add up to 1. Now, the last step in the entire network is to measure the performance of the network after each image is trained so that the network can propagate it backwards to make the learning process better. In general, the losses are very small values which seem insignificant and hence mean square error is not an apt loss function. Cross-entropy function through its logarithm allows the network to assess even these seemingly insignificant error values and steer the network in the desired direction.

4.2. Recurrent Neural Network

A Recurrent neural network are networks that retain the prior information. They contain loops between the layers for information to be passed amongst themselves. RNNs are particularly useful and applied on any data that contain temporal information or have time detail associated with it. Since audio signal is continuous along time domain, even after being transformed into a 2-Dimentional special image, the synergetic effect of using CNNs together with RNNs has produced great results on time related data. This is why we have used different implementations of CNNs along with RNNs. We have used Gated Recurrent units as our RNN neurons.

4.3. Signal to Noise Ratio

Signal to Noise ratio of any signal is basically logarithm of Signal Power over Noise Power. Best way to assess any signal is to calculate its Signal to Noise ratio. Any audio signal which has SNR of above 10dB can be considered as good signal.

5. NOVELTY

It is the norm for any speech recognition model to extract vocal characteristics using MFCC. But deviating from the normal approach we have tried using the log of Mel spectrograms as our features because the DCT used while calculating the MFCCs decorrelates the energies in the actual spectrogram. We have also added noise to the clean data provided by the Kaggle where we deviated from the aim of the provided Kaggle challenge and tried to build a simple speech recognition model which can work in various noise conditions

6. APPROACH

As we have already discussed we are using Recurrent Neural Network and Convolutional Neural Network for our models.

6.1.First Approach:

Our first approach is to build a simple model with one layer RNN and 1 fully connected layer. In this model we have used 118 Gated Recurrent Units as our RNN neurons. We have used a dropout of 0.2 which mean only 80 percent of the neurons which are trained actually contribute to the final layer by which model robustness increases to a great extent. We have added a fully connected layer after the RNN layer with softmax activation function. The model is trained on backpropagation using adam optimizer and model is gauged using categorical cross entropy.

6.2. Second Approach:

Our Second approach is to build a 4 layer model. First layer will be CNN with 48 filters, stride 2. Each of the next two layers have 60 Gated Recurrent Units and at the end two fully connected layer with 84 and 12 units respectively. Here we used ReLU activation in 4th layer and softmax at the output layer. We also used dropout of 0.2. Model is gauged using categorical cross entropy.

6.3. Third Approach:

Our Third approach is to build a 2 layer CNN model. In this model we have used 2 convolutional layers with 28 filters in first layer and 30 filters in second layer. We used strides of 1 and 2 respectively. We have used ReLU activation function at the end of each convolutional layer. We have used drop out too after the activation function with a drop out probability of 0.2 which mean only 80 percent of the neurons trained in each layer are being used in next layer. After the 2 CNN layers we have flattened the data into 1-dimensional and added a fully connected layer in the end which has a linear activation function, ReLU activation function, dropout of 0.2 and a softmax layer in the end with 12 classes. This model is trained by backpropagating using adam optimizer with learning rate 0.0005. We have used a metric called categorical accuracy to gauge our model.

7. EXPERIMENTS

We have used the dataset given by Kaggle to train all the models. Kaggle doesn't provide labels for test data, so we have split the train data into train and validation. We used validation data to test our model.

Silence files created from background noise of dataset are used to add noise in the files. These noisy signals have SNR values ranging from 4dB to 20dB. Each model of each approach is tested on both clean signals and noisy signals. Each model is trained and gauged on both mfcc's and log of melspectogram features. Validation accuracies with mfcc features are shown in table 1 and validation accuracies with melspectogram features are shown in table 2.

Model	Clean Signal Results	Noisy Signal Results
1-RNN	92.34%	76%
1-CNN 2-RNN	95.94%	79.12%
2 CNN	96.02%	87.45%

Table:1 Accuracies of all the models using MFCC features

Accuracies of these models for log of melspectogram features increased from first approach which is 83% to 96% in second approach and 98% in third approach. From which we can say that these CNN layers are perfect for this problem and the 2

Model	Clean Signal Results	Noisy Signal Results
1-RNN	83.48%	83.5%
1-CNN 2-RNN	96.95%	89.45%
2 CNN	98.46%	93.24%

Table : 2 Accuracies using Melspectograms as features.

layer model is sufficient to predict with very good accuracy on clean dataset. The same models when tried on noised dataset with different hyper parameters we have seen that simple RNN has able to reach just 83 percent accuracy and 1 layer of CNN combined with RNNs gave and accuracy of 89 percent. Even with noised dataset 2 layer CNN model outperformed both the approaches and gave an accuracy of 93 percent.

8. CONCLUSION AND FUTUREWORK

The Challenge created by Google and Kaggle helped us to build a Simple Speech Recognition model. As the Kaggle provides labels only for train data we have gauged our models on validation data by splitting train data into train and validation. We have observed melspectograms work better to extract the content in case of Speech recognition, Convolution Neural Networks outperform Recurrent Neural Networks. As per the novelty requirement of this course we have added noise to the files and built models in different environmental noise conditions. Adding Noise to the signals helped in upgrading our model to Robust model. It might seem from the accuracies that the model might be overfitting the data. So our future work will be to increase the robustness of the model and check the accuracies by submitting the results to Kaggle.

9. WORK BREAKDOWN

Pavan Kumar and Harika Putti had researched about the project and read papers about speech recognition. All of them collectively decided about using MFCC's and melspectograms as features for the models. Priyadarshini and Harika worked on preprocessing data (Generating silence files, calculating features from the raw signals). Pavan Kumar and Priyadarshini collectively built models for all clean signals and Harika putti worked on models for noise files.

10. REFERENCES

[1] X. Huang, A. Acero, and H. Hon. *Spoken Language Processing: A guide to theory, algorithm, and system development*. Prentice Hall, 2001.

[2] Mel Frequency Ceptral Coefficients
http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[3] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional Neural Networks for small-footprint Keyword Spotting," in INTERPEECH, 2015

[4] G. Chen, C. Parada, and G. Heigold, "Small-footprint Keyword Spotting using Deep Neural Networks," in Proc. ICASSP, 2014.

[5] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, Dong Yu, "Convolutional Neural Networks for Speech Recognition", IEEE/ACM Transactions on Audio, Speech, and Language Processing (Volume: 22 , Issue: 10, Oct. 2014)

[6] Speech Recognition
https://towardsdatascience.com/tensorflow-speech-recognition-challenge-solution-outline-9c42dbd219c9

[7] Convolutional Neural Networks
https://www.superdatascience.com/the-ultimate-guide-to-convolutional-neural-networks-cnn/

[8] Voice Activity Detection -
https://mycourses.aalto.fi/pluginfile.php/146209/mod_resource/content/1/slides_07_vad.pdf

[9] Automatic Speech Recognition -
https://usabilitygeek.com/automatic-speech-recognition-asr-software-an-introduction/