TENSORFLOW SPEECH RECOGNITION CHALLENGE

Goutham Arra, Divya Rajendran
{garra}, {divrajen}
Indiana University Bloomington

ABSTRACT

We are nearing a stage where there is an overwhelming increase in number of screens everywhere. New versions of objects for daily use are being reinvented with many more screens and people are addicted to these screens. Use of voice command systems, is the perfect remedy to help with breaking screen addiction. To enable this and help the speech command systems to become ubiquitous, we need to encourage independent makers and entrepreneurs to make strides in developing or updating speech recognition technologies. In order to make this possible, Tensorflow [1] has released an open source speech dataset and has invited the world to build a simple speech detector. We shall explore this speech recognition task using deep learning methods, specifically Convolutional Neural Networks. The results look good and hopefully would lead us to the start of a new technological revolution, Automatic Speech Recognition (ASR) systems [2].

Keywords— Speech Recognition, CNN, LSTM, MFCC

1. INTRODUCTION – SPEECH RECOGNITION

Speech Recognition [3] is a technique or method which allows the machine to recognize words out of speech signals. That is, if we have a speech signal as an input, the machine captures the peaks and troughs in the sound wave, identifies the word or words and outputs them as text. There are single words or continuous speech recognition, and user dependent or user independent speech recognition techniques. When a single word is uttered and needs to be identified and converted into text, it is called a single word speech recognition and if its multiple words in a sequence, it is called a continuous speech recognition. The same applies to the user dependent and independent, where the words are uttered by a single speaker or multiple speakers.

Tasks like Speech Recognition makes use of machine learning techniques such as Hidden Markov Models (HMMs) [4], Dynamic Time Warping (DTW) [5], Neural Networks [6], and Recurrent Neural Networks (RNNs) [7]. In this project we aim to approach user independent speech recognition using convolutional neural networks (CNNs) [8] and RNNs which are discussed in the later sections. The speech signal is fed into the CNNs after we transform the input audio signal into Mel-Frequency Cepstral Coefficients (MFCC) [9] with pre-existing Python libraries (Librosa [10]). Since the input is an image, it can be best trained using CNNs.

2. DATA

2.1 Dataset

We have obtained our data from "Warden P. Speech Commands: A public dataset for single-word speech recognition, 2017" [11]. This dataset was released by Google Brain Tensorflow [1] team and is freely available in the Tensorflow website. The dataset contains 67,727 training audio files and more than 1,50,000 test audio files. Each audio file consists of a single word uttered by different people who are chosen from diverse dialects and are approximately one second long. The training set contains 30 different words like up, down, left, right, bird, cat, house and so on.

2.2 Preprocessing

Our goal is to train a model to predict 12 labels on the test data namely, yes, no, up, down, left, right, on, off, stop, go, *silence* and *unknown*. This is a multiclass classification problem. If an audio signal contains a word other than above 10 words, we need to bucket it as *unknown*. We must predict *silence* if an audio signal has only noise and no word spoken in it. To enable silence label classification, we are given six background noise files which are 44s long. When compared with the number of audio files for the 10 words, 6 is a very small number to classify the label *silence*. So, we had sampled 1-second chunks of noise files from these 6 files to create around 2000 silence files in total.

In order to take advantage of the deep learning methods, discussed in the later sections, we have converted all the audio signals into 2D images using MFCC transformation. To enable convenient loading of these image files, we have converted them into Numpy [12] arrays and saved them to our local drive. This helped us in faster data load, when we used our models, saving us a huge amount of time.

3. NOVELTY – DATA AUGMENTATION

An interesting point to note in our dataset is that we have a lot of testing observations (150k) in contrast to the training observations (67k). It can be attributed to the presence of the *unknown* target label, which needs sufficient unseen words to be classified as unknown. And it is important for our classifier to bag these new words into the *unknown* category.

One way to introduce our classifier to more unseen examples is through Data Augmentation [13]. Data Augmentation is a simple process of increasing the size of our training dataset, making use of the observations in the training dataset itself via different techniques. A few of the techniques we have used when dealing with these signals are pitch shifting [14], adding white noise [16], stretching the audio file [14], and creating normalized background noise [18]. All of the techniques have implementations in librosa, the python library.

3.1. Pitch Shifting

Pitch shifting [14] is the opposite of changing the pitch without affecting the speed or duration of the audio signal [14]. ("Pitch is a perceptual property of sounds that allows their ordering on a frequency-related scale [15]")

3.2 Stretching

Stretching [14] is a reverse of Pitch shifting. It can be defined as changing the speed or duration of the time signal without making changes to the pitch., keeping the pitch a constant. By performing stretching operation our model will be able to recognize words spoken in different dialects/many people.

3.3 Adding White Noise

"White noise [16] is defined as a random signal having equal intensity at different frequencies giving it a constant power spectral density" [16]. By adding white noise, we would make our model more robust in identifying speech signals in a noisier ambience. Also, we would improve our chances in identifying *silence* category as speech signals under this category are nothing but background noise.

3.4 Normalize all Background noise

Before we talk about normalizing sound signals, let us define a volume of an audio signal. The volume of an audio signal can be defined as a measure of intensity in Sound Pressure Level (SPL) [17]. It is different from 'loudness' of a signal, in that, Volume is not tangible to human perception.

Normalizing [18] an audio file would mean changing the volume of the audio to the desired level. In this project, we have normalized the given background noise files, and have created chunks of *silence* files from it, adding more observations to the existing *silence* audio files.

4. DEEP LEARNING MODELS

4.1 Convolutional Neural Networks

CNNs stands for "Convolutional Neural Networks" [8], and as its name conveys it is a special branch of neural networks which is popularly used to classify images. Before we get into CNN, we need to understand what a neural network is and how it works. Neural Nets [6] are a series of neurons interconnected with each other to imitate the structure of a human brain [6]. Each neuron is an entity or variable which receives an input and performs a set of operations on this input before sending it to the next layer of neurons in the neural nets [19]. Each of these layers has a weight and bias attached to them, using which a computation is performed. This process is repeated several times until the error rate is reduced [19].

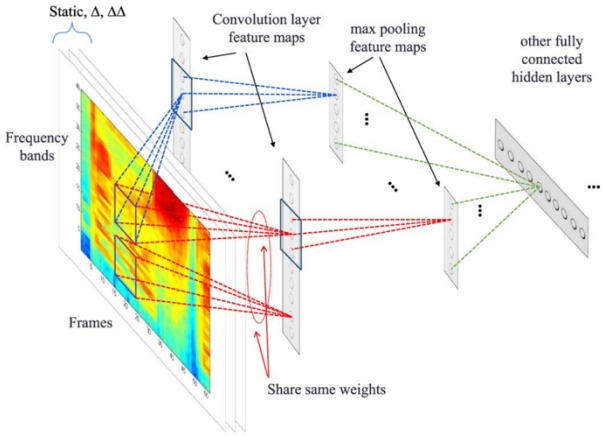

Fig: Illustration of CNN applied on an image [26]

The Neural Networks perform well on several classification problems involving text, however, they do not perform when image input is considered [20]. Here, the convolutional neural networks take over and are highly successful in classifying images and videos making use of convolutions of image or video filters. A convolution is a mathematical operation performed on image pixel matrices of similar dimensions and provides a single value output [21]. This calculated value is multiplied with weights and bias is added to it before sending to another layer for a similar process to be repeated. Since the process gives importance to a previous pixel input while performing calculations on the next pixel, convolutions do not miss out important details. This makes Convolutional Neural Networks work successfully than a regular neural network for image classification [20].

The architecture of a basic CNN consists of three layers, a convolutions layer, a dimensionality reduction layer called pooling layer, and a non-linear classification layer called a fully connected layer [20]. The convolutions layer performs convolutions using an activation function and image filters which is generally an MxN matrix block which runs horizontally or vertically or both based on the type of CNN. This layer picks up important details from each section of the image, during each pass, retains this information and sends it to the next layer. This calculation spans across several hidden layers and on the output from this layer a dimensionality reduction calculation is performed through the pooling layer, to reduce the dimensions without losing important information, and helps speed up the calculations. The output

from the pooling layer is used to make a non-linear classification to predict labels for our images [20].

4.2. RNNs and LSTMs

RNNs [7] are a branch of neural networks which works well with sequences of data be it text or image or time series data. Long Short-Term Memory Cells [22] abbreviated as LSTM is a kind of recurrent neural networks (RNNs) which overcomes the shortcomings of a simple vanilla RNN. RNNs give importance to the time at which an input observation is passed into the network, which allows it to have persistence in the sequence of data flow.

In a normal feedforward neural network, the network takes in an input in a layer, performs some operation and then sends the output to the next layer. Here, the input state or information of a variable is not stored and in every loop of calculation, the sequence information is lost [23]. With the RNNs there exists a special unit to store the sequential information an input provides, called a hidden state [23]. Thus, in every loop of calculation, the RNN has two inputs to a layer, one from the observation itself and the other from the state of the observation calculated in the previous set of calculations. This helps in by preserving time or sequence information which makes the classification very close to the actual labels [23].

Even though this process performs well, there is a drawback in terms of memory [23]. If an input data is huge and we are using an RNN to classify it, the layers in the RNN run out of memory in a few cycles of calculations and we cannot make any meaningful observations out of our network. Other drawbacks of vanilla RNNs are exploding and vanishing gradients. Sometimes, during training/back propagation of RNNs, the multiple derivatives of output error that reach the initial layers, are either very close to zero(vanishing) or very large (exploding). This either makes very negligible updates to weights of neurons or miss the global minimum by making large strides. LSTMs have a memory cell, that has an uninterrupted flow of information throughout the time axis. The presence of several gates also facilitates the flow of information very efficiently, thereby solving exploding and vanishing gradients problem

5. MODELS AND EXPERIMENTS

Our primary approach to this speech recognition problem is rooted in Convolutional Neural Networks and LSTMs. Here are the types of model architectures built by us for this problem.

5.1. Type of Model Architectures:

5.1.1. 1D CNNs:

This is one of the simplest and most efficient models that but gave us decent accuracies. The striking trait of this technique is the shape of the feature detector or kernel which is a row vector that is as long as the number of frequencies and is used to convolve over the input two-dimensional spectrogram. As the name suggests, the kernel is confined to slide only downwards along the time axis (or frequency axis) with an appropriate stride.

5.1.2. 2D CNNs:

This technique enables one to literally treat the spectrogram as a one channel image and apply any image recognition model architectures. The feature detector or the filter used in this technique is the regular two-dimension grid which slides both along the frequency and time axes. This process is just like any other conventional CNN architecture used for image recognition.

5.1.3 CNN – RNN Combination:
The main idea in this technique is to extract features using a convnet (CNN + FCN), flatten non-time dimensions, and on top it uses a Recurring Neural Network to exploit the temporal sequence associated with an audio signal. This method needs to be implemented with utmost care ensuring that not a lot of information is disposed of in the pooling stage while applying max pooling since the time domain signal is already very short i.e. one second long.

Our approach to building models can be divided into two phases:

5.2. Phase I:

In phase I, we built 5 models: two each with 1D CNN and 2D CNN – single and three layers. The final model is 3 layered 2D CNN combined with RNN. All the models are connected to a Dense layer at the end to do SoftMax classification into 12 classes.

For all our experiments, we have split the training data into train and validation sets with 80:20 ratio, leaving around 53k sound samples for training the CNNs.

Here are the results of phase I:

S. No	Model	Train Accuracy (%)	Validation Accuracy (%)
1	1D CNN (1 layer)	64.6	61.8
2	2D CNN (1 layer)	69.3	66.1
3	1D CNN (3 layers)	83.6	78.5
4	2D CNN (3 layers)	96.2	93.8
5	2D CNN (3 layers) + LSTM (2 layers)	98.8	93.18

So far 2D CNN with LSTM has given us the best accuracy. An important point to note here is that, MFCC input vectors to our model have time and frequency axes. We know that the

LSTMs are experts when it comes to working with time series data (time-related data), as they have a memory state apart from other cell states, which runs through the length of all the LSTM cells and captures the temporal dependencies in the training data.

All the above models ran for 50 epochs, with categorical cross-entropy as loss function and Adam as an optimizer. Each convolution block has batch normalization and dropout layers included in them.

5.3 Phase II:

We got our best model as CNN + LSTM. Now, we ran the same model with Data Augmentation. The train set of 64k was increased to 262k train observations after applying the techniques described in the data augmentation sections above.

Here is the table of the performance of best model – CNN + LSTM with and without data augmentation.

Data Augmentation	Trian Accuracy	Validation Accuracy	Test Accuracy
No	98.8	93.18	69.2
Yes	97.29	97.36	71.1

Test accuracy was obtained by getting model predictions on the 150k test set and uploading them to Kaggle [24] website for a score. One slight anomaly is that when we did data augmentation, train accuracy was slightly less than validation accuracy. This may be due to the fact that we built the same model for the 64k train set and the 262k augmented train set. So, in the latter case, our model is decapacitated for a larger train set, which is same reason why the augmented test accuracy is only slightly better than non-augmented test accuracy.

The dataset released by Tensorflow, can be categorized as an imbalanced dataset, with 'unknown' category containing 57% of train labels. So, we need to check our model performance based on other scores like precision, recall etc. Here is a below table of classification report of our best model on the augmented dataset

Label	Precision	Recall	F1- Score	Support
right	0.96	0.97	0.96	1911
yes	0.99	0.97	0.98	1823
no	0.97	0.92	0.95	1873
up	0.97	0.95	0.96	1923
down	0.99	0.89	0.94	1825
left	0.97	0.95	0.96	1827
on	0.96	0.94	0.95	1978
off	0.98	0.94	0.96	1900
stop	0.99	0.92	0.96	1985
go	0.95	0.92	0.93	1901
silence	0.99	1.00	1.00	818
unknown	0.97	0.99	0.98	32813

We can see that there are a greater number of unknown class examples than other labels. Recall is the measure of how well we are capturing positive cases, here, correct label. Precision value indicates, among all the positive cases, that were identified by the model, how many are actually positive cases.

We can see that our precision values for all the labels are pretty high, suggesting, our model is predicting very few false negatives. Recall for 'no', 'stop', 'go' and 'down' is less when compared to others, implying that our model is not able to capture/recognize these words as well as others. The above table is just on the training set, it would be curious to see a similar table for the test set but unfortunately, we do not have access to ground truth labels for the test set.

6. CONCLUSION AND FUTURE WORK

Using deep learning framework, Keras, and methods, CNNs and RNNs, we were able to build a simple speech detector that can find applications in a constrained word-space in the English language. Nevertheless, the algorithm can be scaled to any number of words with less extra effort because deep learning models act like a black box that learns and approximates any function in the world.

Based on the results we observed in phase I and II, we can say that the increase in the number of CNN layers and data augmentation improves test accuracy of our speech detector. And, we were unable to build a complex CNN model with more layers to train on the augmented train set due to lack of GPU resources.

As a future work, we are excited to try Connectionist Temporal Classification (CTC) [25]. CTC identifies phonemes in the input signal before sending them into RNNs and then outputs letters to combine them to a word, that can then be backpropagated or use as final output.

7. WORK BREAKDOWN

Divya did all the preprocessing of data and Goutham built all the models. In the write up of this documentation Divya had written the sections Abstract, Introduction, data preprocessing, and Deep Learning models. Rest of the sections were written by Goutham Arra.

8. ACKNOWLEDGEMENTS

The authors would like to thank Professor Dr. Minje Kim for giving us valuable feedback on this project. We would also like to thank the Associate Instructors who provided us with constructive feedback on our project.

9. REFERENCES

1. Tensorflow - https://www.tensorflow.org/
2. ASR - https://usabilitygeek.com/automatic-speech-recognition-asr-software-an-introduction/
3. Speech Recognition https://en.wikipedia.org/wiki/Speech_recognition
4. HMMs https://en.wikipedia.org/wiki/Hidden_Markov_model
5. DTW https://en.wikipedia.org/wiki/Dynamic_time_warping
6. Neural Networks https://en.wikipedia.org/wiki/Artificial_neural_network
7. RNN https://en.wikipedia.org/wiki/Recurrent_neural_network
8. CNN https://en.wikipedia.org/wiki/Convolutional_neural_network
9. MFCC http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/
10. Librosa https://librosa.github.io/librosa/
11. "Warden P. Speech Commands: A public dataset for single-word speech recognition, 2017. Available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz".
12. Numpy- http://www.numpy.org/
13. Data Augmentation - https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced
14. Pitch Shifting and stretching – https://en.wikipedia.org/wiki/Audio_time_stretching_and_pitch_scaling
15. Pitch Definition - https://en.wikipedia.org/wiki/Pitch_(music)
16. White Noise - https://en.wikipedia.org/wiki/White_noise
17. Audio Signal - https://en.wikipedia.org/wiki/Audio_signal
18. Audio Normalization - https://en.wikipedia.org/wiki/Audio_normalization
19. Neural Network Architecture - https://becominghuman.ai/artificial-neuron-networks-basics-introduction-to-neural-networks-3082f1dcca8c
20. CNN - how it works - http://cs231n.github.io/convolutional-networks/
21. Convolution - https://en.wikipedia.org/wiki/Convolution
22. LSTM - https://en.wikipedia.org/wiki/Long_short-term_memory
23. RNN - how it works - http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
24. Kaggle - https://www.kaggle.com
25. CTC - https://en.wikipedia.org/wiki/Connectionist_temporal_classification
26. Image - 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)