Tensor Flow Speech Recognition Challenge

Data Mining CS 653 - Team 1: Aztech

Abdullah I. Almahmoud, Vyshak Athreya Bellur Keshavamurthy, Vini Kalra, Xin Zhou

Abstract

We are living in an era where speech processing and recognition is needed in everyday life. Some of the main applications for speech processing recognition are customer services' centers, where customers need simple information that does not require a live operator, tapping calls by government agencies for security reasons, and for many entertainment devices where a person can control the device with speech commands. The main challenge with building a speech recognition model is to find a good dataset for testing and improving the system. In this project, a speech processing model has been designed using feedforward and LSTM. The model has been tested using the Tensor Flow Commands Dataset.

Introduction

What machines are able to do nowadays was not even imaginable few years back and it should not stop at this point. The industry of technology is moving forward at an exponential rate and it is not possible to predict what is coming next. Speech processing and recognition software are developing very fast. In this project, overcoming previous known issues of speech processing and recognition was the motivation to produce a more powerful and trustworthy model.

The main challenge of previous speech processing and recognition models was when a person needs to repeat a word several times because the machine was not able to understand it the first times. When digging on the reasons of this problem, it happens because of one of two main reasons, either the accent of the speaker was not recognizable by the model (shortage in the training data) or because of the speaker's surroundings (noise). The first step to overcome this problem is to choose the right dataset. In this project, TIDIGITS dataset was used for testing at first which include clips for 11 digits. However, A recently released dataset by Tensor Flow, named The Speech Commands Dataset, was chosen for training and testing the model of this project. The dataset will be discussed in detail later in this paper.

It is a fact that most of the time in building machine learning models are used for data handling and preprocessing. A more revised version of the data helps in increasing the accuracy of the model. For this project, time warping and merging speech clips with different noise clips was perfumed in addition to other preprocessing techniques. Preprocessing will be discussed in detail later in this report.

Choosing the right model is a crucial step to build a state of art system. Some of the options are support vector machine (SVM), K nearest neighbors (KNN) and Artificial Neural Network (ANN). ANN was chosen for this project. ANN with using some existed regularization methods such as L1, L2 and dropout can improve the model accuracy by 3% to 10%. It is not possible to know which regularizes to use beforehand. Testing and analyzing is needed to choose the best one. This will be discussed in detail later in this report.

Data Sets

"TIDIGITS" 11 digits dataset, contains speech clips from 0 to 9, and 'o'. That is there are 11 classes. Duration of the clip is about 1 second. TIDIGITS is a corpus collected in a quiet recording enclosure. With an aid of a microphone, it was sampled at 20000 samples per second. This dataset is an abridged version of the full data set comprising only of recordings from 111 men and 114 women. The original dataset also contains recordings of children.

This was used for getting hold of speech processing methods and algorithms.

Tensor Flow Speech Recognition data set was the one which we used next. This is a collection of various speakers obtained by Google. Audio clips are about 1 second and contain simple speech commands. Labels used here are yes, no, up, down, left, right, on, off, stop and go. These are part of the train data. Test data contains additional different labels which are not found in the training set. 20 core commands were recorded from different speakers. Best recording among the 5 attempts is categorized into appropriate folders. Care is taken not to replicate the data or speaker in the same folder. Recordings are taken from an uncontrolled environment with no record of the speaker maintained. Clips are of varied lengths, but the maximum duration is kept as 1 second.

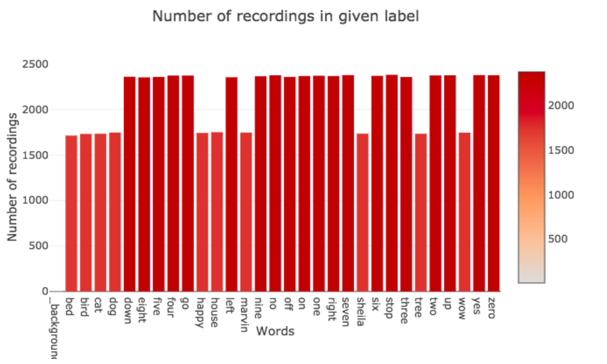

Figure 1, Tensor Flow Speech Challenge Dataset Visualization

From the above analysis, we can understand that number of files in each category is similar. Dataset seems to have a balance and would be suitable for training and testing on the neural networks.

Preprocessing

To filter the noise and recognize the signal, there are various methods that can be used. Models such as Hidden Markov model and Gaussian Mixture Model, help in distinguishing speech from the noise based on the maximum probability of the distribution. Speech and silence distinguishing are carried out upon the highest likelihood. Higher mean is speech and lower mean is the noise.

As discussed in the dataset description, audio clips are time-varying. Few even are just mere noise while the others contain recordings with nothing really heard. Given that two speakers say the same word with different lengths, it is difficult for a neural network to identify patterns. Dynamic Time Warping(DTM) is a technique which measures the similarity between two sequences. It is used in speech recognition to cope up with different speaking speeds. [3] This helps us to analyze different signals of different lengths containing speech and silence at different time intervals. Sequences are warped at nonlinear times. Also, a warping path is produced allowing signals to be aligned in time. Fig 2 shows dynamic time warping of 2 signals in the folder cat. The white line shows the optimal length of the two signals, where speech has occurred.

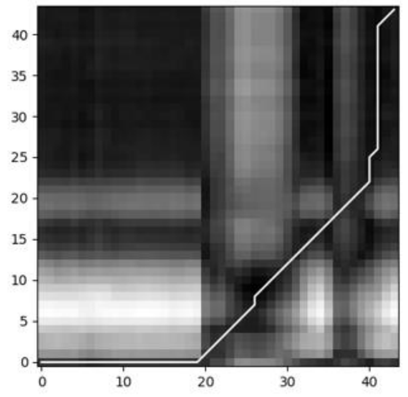

Figure 2, Dynamic Warping Spectrogram

With each folder containing clips of the same recording, a K-NN algorithm was used to understand dynamic clustering of the signals. Euclidean distance is a good measure of these algorithms. But DTM helps us in getting a clearer picture. With DTM as the distance criteria clustering was performed on each folder.

To cluster the data and understand the distribution, we first obtained the Mel frequencies of the sound clips. Mel-Frequency cepstral coefficients(MFCC) are frequency banks which relate closely to the behavior of human auditory system. Mel scale perceives pure tone as compared to the actual frequency. We then measured DTW distance between two MFCCs. By applying K-means algorithm using the above parameters, we could identify clusters in the data. Fig 3 shows K-means clustering of the samples in the folder “cat”. Data is grouped into 4 clusters. Frequencies other than these 4 groups are outliers. They could be mere noise or silence.

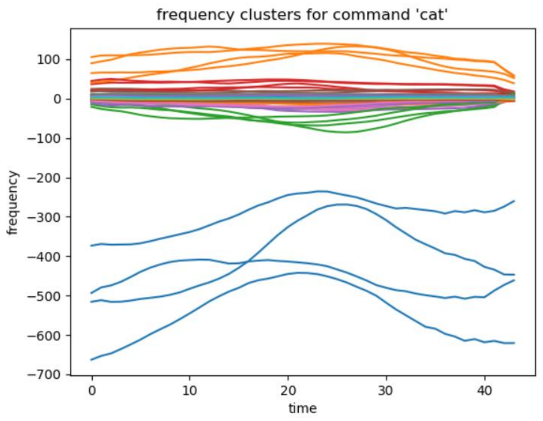

Figure 3, k-means classification

Next preprocessing step that we took was to mix noise with the signals. The dataset contains a folder with different kind of noise clips. These were mixed in random proportion and quantity using a pydub library. Noise clips in the data are of length 1 minute. For our experiments, we used them by trimming it to just one second. Wave Pad, a commercial software was used to do the same.

Feedforward Regularization Methods

L1 & L2

Assume loss function without regularization is F. L1 regularization technique is called Lasso Regression, and its penalty is:

$$\alpha \sum_i | w_i |$$

where w_i is the i^{th} input’s weight going into the activation node, α is a parameter used to adjust the ratio of penalty added to the cost function.

L1 cost function:

$$F + \alpha \sum_i | w_i |$$

L2 regularization method is called Ridge Regression, its cost function is:

$$F + \alpha \sum_i w_i^2$$

As we can tell the differences between L1 and L2 from above equations, the L2 is just sum of squared of L1. This means L2 is more sensitive to the values of F. If values in F is larger, than L2 adds much more penalty to its cost function comparing to L1.

Dropout

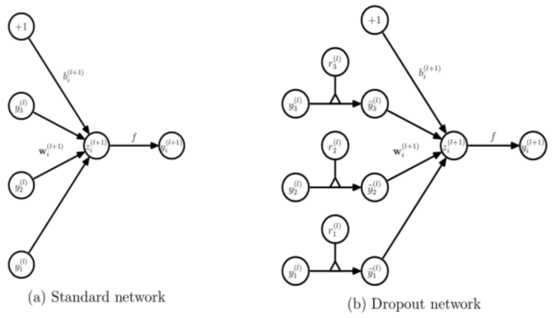

Figure 4, Standard network and Dropout network (Srivastava et al. 2014)

Figure 4 (b) shows a dropout neural net, which randomly drops nodes based on a vector of independent Bernoulli random variables. This dropout layer basically acts like a switch between two layers. In this report paper, there are two different implementations of dropout tested. Inserting a dropout layer before inputs give much better results comparing inserting dropout between hidden layers.

Feedforward Neural Network Regularization Comparison

Here are a few tests results comparing model performance, which the models are implemented with different single regularizer and combination of regularizers.

Other constant model parameters: 1 hidden layer, 20 nodes per layer, regularization parameter 0.01

Table 1Regularization Comparison

Model Regularization	Mean Error Rate	Standard Deviation Error
One Hidden Layer Dropout	0.246	0.061
Two Hidden Layer Dropout	0.545	0.134
L1 only	0.098	0.012
L2 & Two Hidden Dropout	0.90	0.012
L2 only	0.066	0.024
L2 & Two Hidden Dropout	0.090	0.012
L2 & Visible Dropout	0.063	0.014
Visible Dropout only	0.085	0.018

From given table above, using dropout between the input layer and the hidden layer or between hidden layer and output layer is not performing well. One hidden layer dropout means one dropout layer between input and hidden layer, which it has 24.6% error rate. This is error rate increased even comparing to a model without a regularization. The well-performed regularizer is L1 only, L2 only, Visible Dropout only and any combinations with L2 and L1. Even though the combinations of L1 or L2 with Dropout have a low error rate, that only means L1 and L2 are the dominating regularizers. The only well-performed regularizer type is the visible dropout only. In the later experiment, any regularization named dropout means visible layer dropout. As a test result, there are only three regularizers which are L1 only, L2 only and visible layer dropout will be used in other experiments.

Feedforward ANN Hyperparameters Optimization

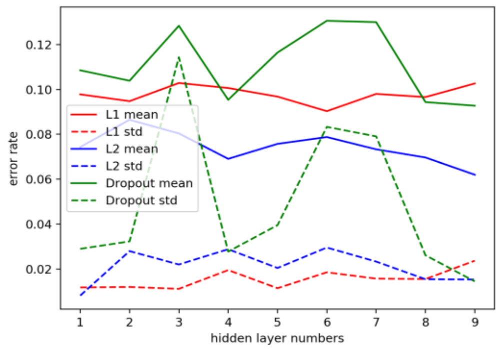

Figure 6, Increase Hidden Layers from 1 to 9

As figure 3 shows what happens when the neural network becomes deeper, L2 regularized models are slightly performing better. L1 regularized model is very stable and layers number does not effect on its performance. Dropout has very random performance which the error rate bouncing between 9% and 15%, this means the layer number does not effect on its performance either.

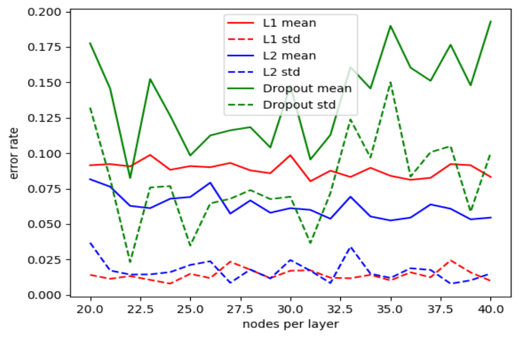

Figure 5, Increase Nodes from 20 to 40

Figure 5 shows how L1, L2 and Dropout perform when we increase nodes per layer from 20 to 40. When the neural network becomes wider, L2 regularized model is slightly decreasing the error rate. L1 is also decreasing the error rate, but it is not obvious. Even though the Dropout has a very random result, its error rate is increasing generally when we the network becomes wider.

LSTM Model

Long Short-Term Memory networks LSTMs are a special kind of Recurrent Neural Network (RNN) that are capable of learning long-term dependencies. LSTMs are specifically designed to avoid the long-term dependency problem. Storing information for a lengthy period of time is practically their default behavior, not something they struggle to learn.

RNNs have the form of a chain of repeating modules of the neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

LSTMs too have this chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four layers which interact in a very special way.

The cell state behaves like a conveyor belt that runs through the entire chain, with only some minor linear interactions.

The first step in LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It outputs a number between 00 and 11 for each number in the cell state. 11 represents "completely keep this" while 00 represents "completely get rid of this."

The next step is to decide what additional information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, that could be added to the state. In the next step, these two are combined to create an update to the state.

It's now time to update the old cell state into the new cell state. The previous steps already decided what to do, we just need to do it.

We multiply the old state by ftft, forgetting the things we decided to forget earlier. Then we add it. This is the new candidate values, scaled by how much we decided to update each state value.

Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output.

Then, we put the cell state through tanh (to push the values to be between −1−1 and 11) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

Feedforward and LSTM Experiment Result

Table Description:
Feedforward Model Parameter Format:
[Hidden Layers, Nodes/Layer, Regularizer, Batch Size, Epochs]
LSTM Model Parameter Format:
[Layers, Units(Cells), timesteps, Batch Size, Epochs]
Timesteps = d means the default of data set which is determined by windows size and time length of the file.
TSRC = Tensor Flow Speech Recognition Challenge Dataset

Table 2 Feed forward and LSTM results

Data Set	Model	Model Parameter	Training Accuracy	Training Time (min)	Test Accuracy (or Validate)
TIDIGITS	Feed forward	[1, 20, L2, 100, 30]	95.42%	1.5	82.31%
TIDIGITS	LSTM	[2, 32, 1, 100, 30]	98.98%	3.8	89.45%
TIDIGITS	LSTM	[2, 64, 1, 100, 100]	99.92%	5.6	91.15%
TIDIGITS	LSTM	[2, 64, d, 100, 100]	100.00%	7.8	94.31%
TSRC	Feed forward	[2, 60, L2, 100, 200]	31.75%	60	5.32%
TSRC	LSTM	[3, 32, 1, 100, 100]	71.56%	600	15.19%
TSRC	LSTM	[3, 64, 1, 100, 200]	81.79%	200	21.45%
TSRC	LSTM	[3, 128, 1, 100, 200]	92.34%	100	35.79%
TSRC	LSTM	[3, 256, 1, 100, 200]	98.98%	40	55.52%
TSRC	LSTM	[3, 256, d, 100, 200]	99.21%	70	67.00%

Kaggle Leaderboard: Rank 437 (800 teams), Score 0.67

Some of the training time was estimation based on a few test experiments, but we know that if we increase units of LSTM model, the training time will be much less and the loss converges to the local minimum much faster. That is because we have too many training data, and the model will be underfitting if the LSTM units are not enough. One other key parameter of the model is epoch size, the accuracy increases as epoch size increases, but the training time increases as well. Feedforward model performs well for the data set TIDIGITS, but it has really substandard performance on the Tensor Flow Speech Recognition Challenge (TSRC) dataset. TIDIGITS dataset has decent size of data files with good time-related features. However, TSRC dataset has variety of real speeches in the real environment. For example, TSRC has a high diversity of speeches with all English speakers all over the USA. One of the biggest problem comparing these two datasets is that TSRC does not have time-related speech files. Therefore LSTM works much better than feedforward model in speech words recognition, which LSTM memorize features in time instead of feature dimensions.

Future Improvement

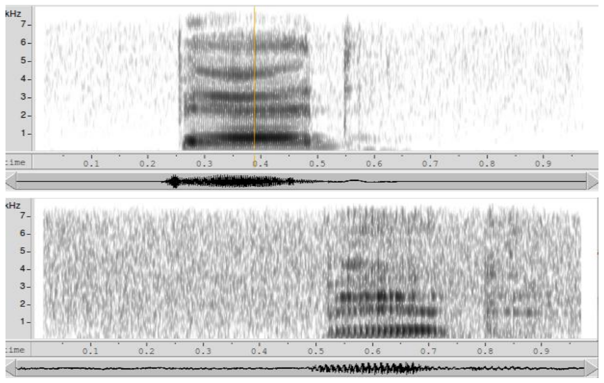
Figure 7, Spectrograms of "bed", Normal and Noise

67% test accuracy rate is not an ideal result. As we analyze the result, we found that most of the command speeches such as "right", "left" have very high accuracy which is approximately 90%. Some of the files which have background noise are more likely to be labeled wrong. The two spectrograms above represent two different speech clips of "bed". As we can tell the second spectrogram is mixed with a lot of noises, which will corrupt our training model's weights if this file is in training dataset. In order to fix such problem, we need two different classifier models. One classifier is trained with clean speech data and the other classifier is trained with speech data which are mixed with noises.

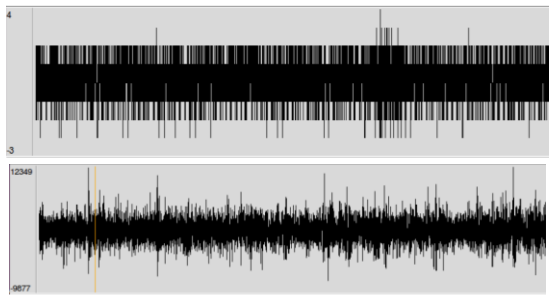
Figure 8, Waveforms of Silence and Noise

The most frequently wrong labeled speeches are silence and noise files as shown above. The first waveform is a program generated silence waveform, and the second one is a recorded or generated noise waveform. As we input these data into our trained model, we cannot control what result we will obtain. In order to fix this issue, we need a filter in front of our trained model.

Conclusion

In this project, several preprocessing methods and two different neural network structures had been practiced on two speech data sets. K-means was one of the unsupervised clustering method we had tried to group faster Fourier transformed data into formants clusters, and we also tried to use time warping. However, both methods resulted in very complicated data dimensions handling. Some of the preprocessing methods were not adopted when we found that LSTM could handle time. As an experiment result of comparing feedforward neural network and long short-term memory (LSTM), feedforward could not handle time domain data well. As we investigated more on the LSTM model, timesteps and cell units are the most important parameters. Increasing cell units would speed up training and provide more weights to prevent underfitting. 256 units is not an optimal number, but it is the best number in our experiment on Tensor improve 67% to a higher test accuracy in the future, we plan to implement silence/noise filters in front of two trained classifiers. One classifier is used to train speech clips mixed with noise, and the other classifier is for speech only. Moreover, our method

Flow Speech Recognition Challenge data set. To of extracting speech words was based on the very center of each file which is inaccurate and may lost some of the important data. Padding small numeric number onto each data file to get fixed time speech data would solve this problem in the future.

Reference

[1] Knight W. (May 29, 2012). Where Speech Recognition Is Going. Retrieved from https://www.technologyreview.com/s/427793/where-speech-recognition-is-going/

[2] Feldon B. (Mar 22, 2017). The Top Five Uses of Speech Recognition Technology. Retrieved from https://www.callcentrehelper.com/the-top-five-uses-of-speech-recognition-technology-1536.htm

[3] https://catalog.ldc.upenn.edu/ldc93s10

[4] Data download, http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz

[5] Guptha et all, Noise Classification Using Gaussian Mixture Models , Int’l Conf. on Recent Advances in Information Technology | RAIT-2012

[6]Arribas-Gil, A time warping approach to multiple sequence alignment, https://arxiv.org/pdf/1606.06017.pdf

[7] Colah’s Blog http://colah.github.io/posts/2015-08-Understanding-LSTMs/