Spoken Digit Recognition (Speech Recognition)

Avi Khemani
akhemani@stanford.edu
https://github.com/avikhemani/CS230Project

Abstract

Today, numbers make up most unique identifiers, such as bank information, social security numbers, registration codes, etc. With recent improvements in general speech recognition, this paper seeks out to achieve perfect accuracy for spoken digit recognition. This would enable people to fully automate the manual communication of numbers in person and over the phone. The dataset contains 30,000 labeled audio clips of digits from 0 to 9. We first test the effectiveness of linear neural networks on the amplitudes of the audio clip as a time series. We also test a convolutional neural network on a spectrogram of the audio clips. With the convolutional approach, we are able to achieve a 99%+ accuracy for spoken digit recognition on our test set.

1 Introduction

In today's society, numbers are used everywhere as forms of identification, such as in bank information, social security numbers, registration codes, access codes, etc. However, in many transactions, people still find themselves having to list out the digits one by one over the phone for the other party to ensure that they have it down correctly. With the significant improvement in speech recognition algorithms in recent years, this problem can be automated if deep learning is applied to digit recognition.

For this project, the inputs are one second audio clips of a specific digit, ranging from 0 to 9. We then use a deep neural network to classify this audio clip and output the specific digit that was spoken.

2 Related Work

In recent years, speech recognition has quickly been evolving with products such as Siri[1], Alexa[2], Google Home, and much more. These products can recognize speech and then, using NLP, carry out a conversation with the user. There have also been several attempts to recognize specific commands that a person may say[3].

In contrast to these aforementioned examples, my algorithm must produce a really high accuracy due to the confidentiality and precision that many of the unique identifiers require. This includes accounting for different pitch and accent possibilities as well as different speeds at which people talk. Reaching an accuracy of greater than 99% is the ultimate goal for this project.

3 Dataset and Features

The raw data consists of 30,000 1-second recordings from 60 different speakers saying a digit from 0 to 9. The recordings were captured at a sample rate of 48kHz and trimmed to minimize silence. Each of these audio clips is associated with the correct value that was spoken in the clip. The data also comes along with the gender, age, and nationality of each individual speaker.

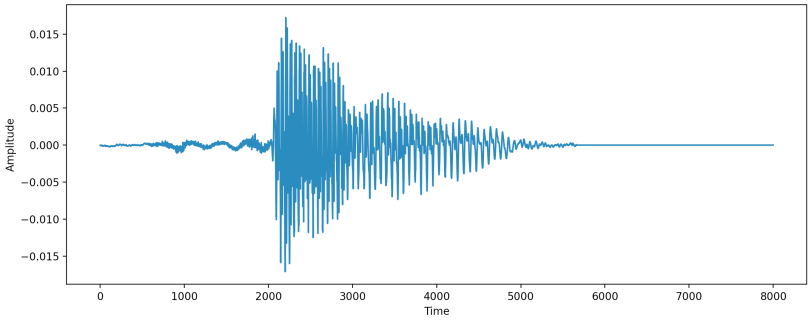
There are a couple steps to preprocess the data before inputting it into the neural network. First, all audio clips are downsampled to a sampling rate of between 8kHz and 22kHz. Since speech is low

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

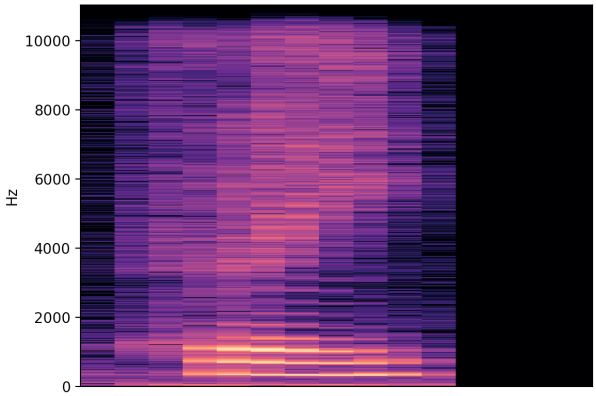
bandwidth (between 100Hz - 8kHz), 8kHz would probably be sufficient, but higher frequencies may provide more precision, so we will tune this hyperparameter in the project. Sampling rates of higher than 22kHz do not add much value and require much more computational power. Also, all audio clips are zero padded to make them exactly 1 second so that they are a equal inputs into the neural network.

3.1 Feature Extraction

I experimented with two different ways of inputting the audio file into my model. The first method is to extract the amplitudes of the audio clip into a large array. Below is the visual waveform of someone saying "zero" after being pre-processed:



Another common method in speech recognition is to convert the audio file into a spectrogram of frequencies. Below is a spectrogram of someone saying "zero" after being pre-processed:



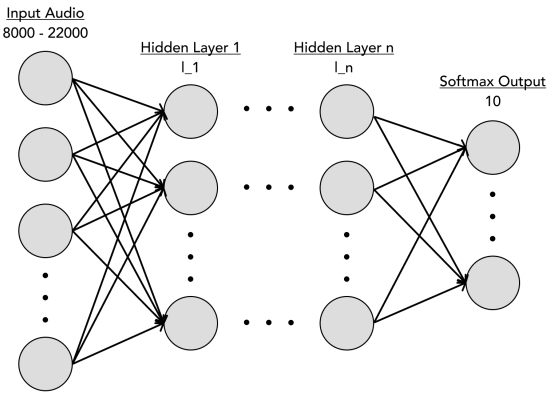
With 30,000 audio clips in the dataset, I went with a 90-5-5 random split for the training, dev, and test sets respectively. The dataset was collected for use in a paper titled, *Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals*[4], where they wanted to better understand the structure neural networks in speech recognition.

4 Algorithms and Results

Because of the specificity of this digit recognition task, I have coded the algorithms myself from the ground up with inspiration from speech recognition applications in other areas. The inputs are the audio files of a spoken digit and the output is a digit between 0 to 9 for which the audio most likely corresponds to. I am using Librosa to extract necessary information from the audio files, PyTorch to create and run my neural networks, and Google Colab Pro for stronger computational power and higher RAM. To assess the performance of my algorithms, I calculated the accuracy on the training and dev/test sets.

4.1 Neural Networks

I have currently implemented a deep neural network that takes the amplitudes of the audio clips as a time series and inputs them as a vector into the network. The last layer of the network is pushed through a softmax activation function to output the 10 probabilities that the audio is each of the digit possibilities. A diagram of the network architecture is provided below:



Given training input x, and softmax output y, the network optimizes the Multi-Label Soft Margin Loss function, defined as:

$$\text{loss}(\mathbf{x}, \mathbf{y}) = -\frac{1}{\mathrm{C}} * \sum_{\mathrm{i}} \mathbf{y}[\mathrm{i}] * \log((1 + \exp(-\mathbf{x}[\mathrm{i}]))^{-1}) + (1 - \mathbf{y}[\mathrm{i}]) * \log\left(\frac{\exp(-\mathbf{x}[\mathrm{i}])}{(1 + \exp(-\mathbf{x}[\mathrm{i}]))}\right)$$

I begin by training a 3-layer neural network with Relu activation functions, Adam optimization, a learning rate of 0.001, 25 epochs, and a batch size of 64. I will first search for the most accurate and computationally efficient sampling rate for the audio. The following table demonstrates several of the trials to tune layer sizes for each sampling rate:

Sampling Rate	Layer Sizes	Training Accuracy	Dev Accuracy
8 kHz	8000 - 1000 - 500 - 10	91.2%	55.2%
16 kHz	16000 - 1000 - 500 - 10	91.2%	55.1%
22 kHz	22000 - 1000 - 500 - 10	91.0%	54.2%

From the results above, it seems that sampling rate has a low impact on accuracy, but 8kHz is the most computationally efficient. Thus, we will proceed with 8kHz. It also is evident that our current model is overfitting to the training set, and that we are seeing a lot of variance in our model. For the next set of trials, we will experiment with techniques to decrease variance, such as fewer epochs, L2 regularization, and a smaller network. Below are several of the trials (red indicates changes):

Layer Sizes	Epochs	L2 Regularization	Training Accuracy	Dev Accuracy
8000 - 2000 - 10	25	No	96.0%	60.3%
8000 - 1000 - 10	25	No	95.3%	60.9%
8000 - 500 - 10	25	No	93.7%	59.6%
8000 - 1000 - 10	10	No	84.3%	57.7%
8000 - 1000 - 10	15	No	91.4%	60.6%
8000 - 1000 - 10	20	No	94.3%	60.0%
8000 - 1000 - 10	20	Lambda = 1e-6	87.6%	62.0%
8000 - 1000 - 10	20	Lambda = 1e-7	91.0%	60.9%
8000 - 1000 - 10	20	Lambda = 1e-8	94.2%	61.3%

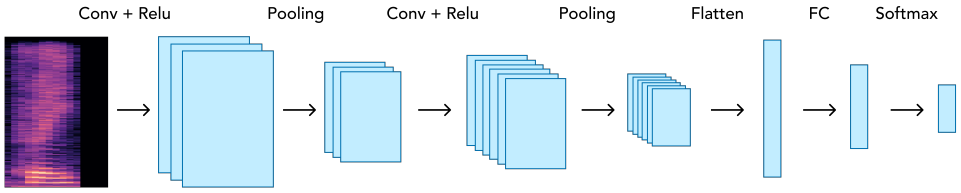
Using a smaller network, fewer epochs, and L2 regularization, the model reduced its variance, and increased the dev accuracy from 55.2% to 62.0%. However, the model is still experiencing quite a

high variance that may require more techniques such as inverted dropout or a different architecture. Running the most optimal combination of hyperparameters on our test set gives us the following:

Layer Sizes	Epochs	L2 Reg	Train Accuracy	Dev Accuracy	Test Accuracy
8000 - 1000 - 10	20	Lambda=1e-6	87.6%	62.0%	61.5%

4.2 Convolutional Neural Networks

Because the linear neural networks were underperforming, I experimented with a different approach. Specifically, I implemented a deep convolutional neural network that creates a spectrogram of the audio clips and inputs them as 2-dimensional matrices into the network. The last layer of the network is pushed through a softmax activation function to output the 10 probabilities that the audio is each of the digit possibilities. A diagram of the initial convolutional network architecture is provided below:



Based on related speech recognition tasks, I decided to use a 16kHz sampling rate instead of 8kHz sampling rate so that the images could encode more information. This network also optimizes the Multi-Label Soft Margin Loss function as defined in the previous section.

I begin by training variations of the above architecture with constant 16 kHz sampling rate, 1e-6 learning rate, Adam optimization, 30 epochs and a batch size of 64. (red indicates variations)

Channels	Kernels	FC Layers	Train Accuracy	Dev Accuracy
1 - 5 -10	5 - 5	Flat - 1000 - 10	97.8%	97.8%
1 - 3 - 5	5 - 5	Flat - 1000 - 10	91.6%	93.5%
1 - 5 - 10	9 - 5	Flat - 1000 - 10	97.8%	97.7%
1 - 5 - 10 - 15	5 - 5 - 3	Flat - 1000 - 10	85.1%	86%
1 - 5 - 10	5 - 5	Flat - 5000 - 1000 - 10	99.6%	99.2%

From the results above, we can see that the convolutional neural network architecture is much more accurate and does not suffer from high variance, as seen with linear neural networks. In fact, we were able to achieve a 99.2% accuracy on the dev set. As such, we can focus on reducing bias in the algorithm with methods such as longer training and deeper architectures. We will build off of the higher performing algorithms in the table below. (red indicates changes)

Channels	Kernels	FC Layers	Epochs	Train Accuracy	Dev Accuracy
1 - 5 - 10	5 - 5	Flat - 1000 - 10	50	99.6%	99.3%
1 - 10 - 20	5 - 5	Flat - 1000 - 10	50	99.6%	99.3%
1 - 5 - 10	9 - 5	Flat - 1000 - 10	50	98.6%	98.3%
1 - 5 - 10	5 - 5	Flat - 5000 - 1000 - 10	50	99.7%	98.6%
1 - 10 - 20	5 - 5	Flat - 5000 - 1000 - 10	30	99.1%	99.0%

Using methods to reduce bias, we were able to increase both training accuracy and dev accuracy for all of the trials, with a high of 99.3% accuracy on the dev set. However, with the deeper networks, the models started to experience variance, which can potentially be solved with varied architectures and/or regularization. Running the most optimal combination of hyperparameters on our test set gives us the following:

Channels	Kernels	FC Layers	Train Accuracy	Dev Accuracy	Test Accuracy
1 - 5 - 10	5 - 5	Flat - 1000 - 10	99.6%	99.3%	99.3%

5 Evaluation

While the linear neural network model was able to achieve 90%+ training accuracy, it only reached a 62% accuracy on the dev and test set, indicating a large variance problem. Even with several attempts to reduce the variance (fewer epochs, shallower network, L2 regularization), the model did not see any significant increase in the dev and test set accuracy. This could mean that the amplitudes of the audio clip are not enough information to learn from or that there should be more data pre-processing before training.

In contrast, with the convolutional neural network approach, I was able to achieve my goal of greater than 99% accuracy. The CNN did not suffer from much variance, indicating that this approach was much better at generalizing. With 99.6% training accuracy and 99.3% test accuracy, we can next try methods that reduce bias or reduce variance to get closer to perfect accuracy.

5.1 Challenges

The main challenge I faced in this project was working with such a large dataset. With 30,000 audio clips, it was very time consuming to pre-process and extract features from all of the clips for training. In addition, it was very slow to run them through a deep neural network. Due to difficulties with setting up an AWS instance, I ran my linear neural networks on Google Colab, using the free GPU that they provide. However, the free GPU did not support enough RAM for my convolutional neural networks. As a result, I had to purchase Google Colab Pro for a higher RAM GPU to train my CNNs.

5.2 Future Work

With a more diverse dataset and stronger computational power, a 99.9% accuracy for spoken digit recognition is very possible. An optimized algorithm could replace human labor of communicating important identifiers between people. These neural network architectures can also be used for many other applications, such as spoken commands to turn on/off the lights in a household as an example. Even though current speech recognition algorithms are effective, none have reached near perfect accuracy and turning to more specific input-output behavior could be the solution for more reliable, targeted products.

6 Contributions

Avi Khemani was the sole contributor to this project. He coded up all of the algorithms from scratch, wrote all of the reports, created the poster, and filmed the video.

7 References

[1] Hey Siri: An On-device DNN-powered Voice Trigger for Apple's Personal Assistant. https://machinelearning.apple.com/research/hey-siri

[2] What Is Automatic Speech Recognition? https://developer.amazon.com/en-US/alexa/alexa-skills-kit/asr

[3] TensorFlow Speech Recognition Challenge. 2018. https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/overview

[4] Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals. https://arxiv.org/abs/1807.03418