Stanford | ENGINEERING

Speech Recognition - From Speech to Text

Ben Limonchik, Rysen Otomo

CS230

Background

Currently, the giant tech companies are fighting to build the best speech based assistant. Nevertheless, Siri, Alexa, Cortana and Google now are good at recognizing words but they still make many mistakes when there is background noise. In addition, no single assistant is good enough at understanding complex human intents such as "Siri what is an easy dish to cook for 6 people". In this project we explored speech recognition through classifying audio files into written words. In order to build a more robust model which can be used in the outdoors, we augmented the audio data using various background noises in order to make our training data set more generalized

amazon alexa

Hey Cortana

Data and preprocessing

We preprocess the data from its raw 16kHz form into a 2D MFCC matrix.
The steps of the transformation are:
1) Take the Fourier transform of (a windowed excerpt of) a signal: we used 98 time windows (one time stepfor every 160 amplitude readings and we remove the first and last time step)
2) Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
3) Take the logs of the powers at each of the mel frequencies. Based on our literary review we saw that the recommended number of bins to use for the MFCC fingerprint is between 26 and 40 for an audio file sample rate of 16kHz. In our project we used 40 bins.
4) Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5) The MFCCs are the amplitudes of the resulting spectrum.Overall our MFCC data representation is a 2D matrix of size 40x98, for 40 frequencies and 98 time-steps.

yes	no	up	down	left	right	on	off	stop	go	silence	unknown
2377	2375	2375	2359	2353	2367	2367	2357	2380	2372	4000	4000

Figure 1: number of examples of each class in our dataset

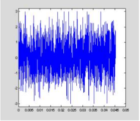
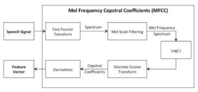


Figure 2: Raw waveform of an audio file



Mel Frequency Cepstral Coefficients (MFCC)

Speech Signal → Fast Fourier Transform → Spectrum → Mel Scale Filtering → Mel Frequency Spectrum → Log(.) → Discrete Cosine Transform → Cepstral Coefficients → Derivatives → Feature Vector

Figure 3: summary of raw sound to MFCC conversion

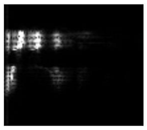
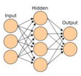


Figure 4: a 40x92 MFCC of the word 'happy'

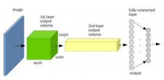
Baseline model

1) Model: One Fully connected layer
input->FC->softmax
accuracy: 48.2%
Parameters: 3920

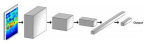


Advanced models

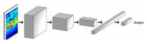
2) Model:
2 Layer CNN (input->conv2D->RELU-> maxpool->conv2D->RELU->fully connected->softmax)
Hyperparameters: learning rate, filter sizes
Number of Filters per layer: 64,64
accuracy:90%



3) Model:
3 Layer CNN (input->conv2D->BatchNorm-> RELU->maxpool->conv2D->BatchNorm->RELU->conv2D->BatchNorm->fully connected->softmax)
Hyperparameters: learning rate, filter sizes
Number of Filters per layer: 64, 64, 128
accuracy:92%

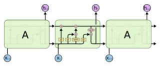


4) Model:
3 Layer CNN (input->conv2D->BatchNorm->RELU->maxpool->conv2D->BatchNorm->RELU->conv2D->BatchNorm->fully connected->softmax)
Hyperparameters: learning rate, filter sizes
Number of Filters per layer: 64, 128, 256
accuracy:94%

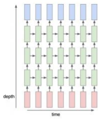


5) Model: Single LSTM cell,
Hidden layers: 100
Learning rates: 0.05 for 6000 iterations and 0.005 for 12000 iterations
accuracy: 83.7%

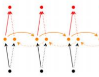
$$
\begin{aligned}
i &= \sigma(x_t U^i + s_{t-1} W^i)\\
f &= \sigma(x_t U^f + s_{t-1} W^f)\\
o &= \sigma(x_t U^o + s_{t-1} W^o)\\
g &= tanh(x_t U^g + s_{t-1} W^g)\\
c_t &= c_{t-1} \circ f + g \circ i\\
s_t &= \tanh(c_t) \circ o
\end{aligned}
$$



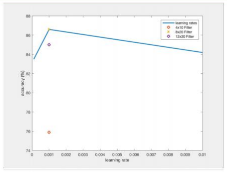
6) Model: Stacked LSTMs
2 and 3 stacked LSTM cells
Hidden layers: 100
Learning rates: 0.05 for 6000 iterations and 0.005 for 12000 iterations
accuracy: 83.4%



7) Model: Bidirectional LSTM cells
Hidden layers: 100
Optimizer: Adam
Learning rates: 0.05 for 6000 iterations and 0.005 for 12000 iterations
accuracy: 88.7%



Results / Hyperparameter Tuning



Hyperparameters in CNN:
The hyperparameters are the learning rate and filter sizes. Iterating over different learning sizes shows 0.001 is the best learning rate and an 8x20 first convolutional layer filter size is the best filter size. Other learning rates tested were 0.01 and 0.0001 while other first filter sizes tested were 4x10 and 12x30. Tuning occured over 2000 training steps. This is when the most learning occurs.

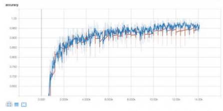


Figure 5: accuracy versus iteration for the top CNN architecture

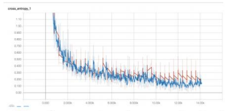


Figure 6: cross entropy loss versus iteration for the top CNN architecture

Dev Set Accuracy:

silence	unknown	yes	no	up	down	left	right	on	off	stop	go
94.2%	86.7%	97.3%	92.0%	90.4%	91.4%	92.2%	98.8%	95.7%	95.1%	97.9%	97.8%

Test Set Accuracy:

silence	unknown	yes	no	up	down	left	right	on	off	stop	go
98.8%	85.9%	96.0%	93.4%	88.7%	91.3%	92.8%	97.1%	95.2%	96.8%	93.1%	96.4%

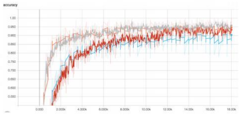


Figure 7: Accuracy versus iteration: Orange curve: best CNN validation, Grey curve: best CNN training, Red curve: best RNN training, Blue curve: Best RNN validation

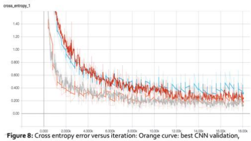


Figure 8: Cross entropy error versus iteration: Orange curve: best CNN validation, Grey curve: best CNN training, Red curve: best RNN training, Blue curve: Best RNN validation

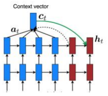
Modifications to RNN model:
1) **Stacking LSTM cells**: Making our networks deeper by stacking more LSTM cells on top of each other had no major effect on final accuracy besides the longer time it took to train the network.
2) **Hidden units**: The number of hidden units used had a significant effect on the final accuracy of the model. We tested the basic feed forward LSTM network with 50, 100 and 200 hidden units:

hidden units	50	100	200
dev/test accuracy	80.4%/80.2%	85.1%/83.7%	84.8%/82.9%

3) **Switching to Bi-directional networks**: This change of architecture led to the most prominent increase in overall accuracy. The best test accuracy achieved with the bi-directional model was 88.7%.
4) **Learning rate**: We used a 0.05 learning rate for the first 6,000 iterations and then reduced it to 0.005 after 6,000 iterations when the accuracy started to plateau.

Future Directions

As seen above our best CNN model outperformed our optimized bi-directional LSTM network. The best CNN network achieved 93.7% testing accuracy while the best RNN model achieved 88.7% testing accuracy. This may be the case because of how the error signal flows through the RNN network. In the RNN the error signal might have to travel up to 98 time-steps to modify the weights of a sound input based on another future input. Given more time we would have experiment with additional network modifications. Specifically, it may be the case that using attention in our RNN model could help speed up the learning process since the relationship between audio features of different times would be better captured by the alpha weights of an attention model.


Context vector
c_t
a_t
h_t