Speech Command Recognition

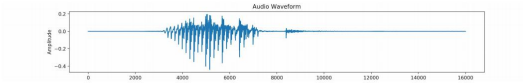
Stanford University
tkarpati@stanford.edu
https://youtu.be/zGTnLnctWL0

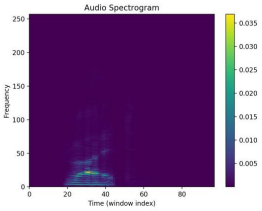
Objective

Speech recognition is allowing more accessible interactions with agents. The use of deep learning networks has produced high accuracy in speech recognition. We look at using methods from image processing applied to traditional speech recognition systems, such as the use of 2D convolutional and pooling layers for time and frequency invariance in the speech patterns. Our system takes audio waveforms and provides probabilities of 12 different command classes which can be acted on by the agent. Our focus is on efficiency of computation and the latency to determine if an utterance is of a certain class.

Dataset

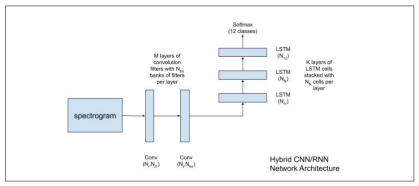
The dataset is taken from the Kaggle TensorFlow Speech Command Recognition Challenge. This consists of 30 classes of spoken commands of which 10 commands are to be recognized. The input data is approximately 1 second of audio sampled at 16KHz. The dataset is 64,727 examples and split into training/validation/test set. The examples are fairly uniformly distributed across the 30 classes. Input data is normalized and converted to spectrogram data to be used as input to our learning network. The FFT and time window are 512 samples in length with 160 sample stride between windows.



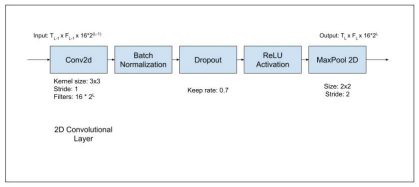


Architecture

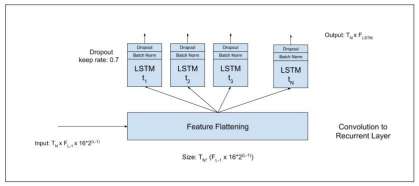
Our model consists of 3 layers of 2D convolutional filtering followed by a layer of feature flattening over each time sample in the sequence. These features are then fed into 3 more layers of LSTM cells. The output is connected through a dense layer to a 12-class softmax layer to determine the probability of spoken command.



Each convolutional layer consists or a 2D convolutional step and a max pooling step for time and frequency invariance.



All of the filter bank outputs and frequency features associated with a single time sample are flattened into a single feature vector and used as the input to the LSTM layers.



Results

We were able to achieve high accuracy with the 2D convolutional and pooling layers of approximately 95% accuracy. This performed better than our baseline convolutional only and recurrent only networks with 1D convolutional layers.

Architecture	Convolutional Kernel			Recurrent		Other	Parameters		Accuracy	
	Size	Stride	# Layers	# Cells	# Layers		Total	Trainable	Train	Validation
Baseline	3	1	4	-	-	Batch Norm, Dropout = 0.5, FC (N = 128), Activation = ELU, Global pooling	-	-	0.8066	0.8073
Keras Baseline	3	1	4	-	-	Batch Norm, Dropout = 0.5, FC (N = 128), Activation = ELU, Flattening	2.048K	2.048K	0.9805	0.9206
Conv1d LSTM	3	1	1	128	1	No Batch Norm, No Dropout	88K	88K	0.7061	0.7651
Conv1d LSTM	3	1	2	128	2	No Batch Norm, No Dropout	219K	219K	0.7940	0.7924
Conv1d LSTM	3	1	1	128	1	Batch Norm, No Dropout	89K	88K	0.9482	0.8856
Conv1d LSTM	3	1	1	128	2	Batch Norm, No Dropout	221K	220K	0.9530	0.8726
Conv1d LSTM	3	1	2	128	1	Batch Norm, No Dropout	99K	98K	0.9497	0.9028
Conv1d LSTM	3	1	2	128	2	Batch Norm, No Dropout	231K	230K	0.9422	0.8850
Conv1d LSTM	3	1	3	128	3	Batch Norm, No Dropout	388K	388K	0.9610	0.8898
Conv1d LSTM	5	3	2	128	3	Batch Norm, No Dropout	374K	373K	0.9709	0.8945
Conv1d LSTM	7	3	2	128	3	Batch Norm, No Dropout	383K	382K	0.9706	0.8982
Conv1d LSTM	15	3	2	128	3	Batch Norm, No Dropout	421K	419K	0.9648	0.8942
Conv1d LSTM	3	1	2	128	1	Batch Norm, Dropout = 0.7	100K	99K	0.9041	0.8470
Conv1d LSTM	3	1	2	128	1	Batch Norm, Dropout = 0.7	100K	99K	0.9631	0.8923
Conv1d LSTM	3	1	3	128	3	Batch Norm, Dropout = 0.5	388K	388K	0.9107	0.8705
Conv1d LSTM	3	1	3	128	3	Batch Norm, Dropout = 0.7	388K	388K	0.9445	0.8480
Conv1d LSTM	7	3	2	128	3	Batch Norm, Dropout = 0.7	383K	382K	0.8857	0.8853
Conv1d LSTM	7	3	2	128	3	Batch Norm, Dropout = 0.7, lr_decay = 0.001	383K	382K	0.8684	0.9060
Conv1d LSTM	3	1	2	128	1	Batch Norm, Dropout = 0.7, lr_decay = 0.001	383K	382K	0.9005	0.9081
conv2d LSTM	3	1	2	128	1	Batch Norm	1.336K	1.336K	0.9822	0.8497
conv2d LSTM	3	1	2	128	2	Batch Norm	1.439K	1.437K	0.9822	0.9347
conv2d LSTM	3	1	3	128	3	Batch Norm	1.439K	1.437K	0.9822	0.9347
conv2d LSTM	3	1	3	128	3	Batch Norm, Dropout = 0.7, lr_decay = 0.0	1.439K	1.437K	0.9779	0.9525
conv2d LSTM	3	1	3	128	3	Batch Norm, Dropout = 0.7, lr_decay = 0.001	1.439K	1.437K	0.9824	0.8756
conv2d LSTM	3	1	3	128	3	Batch Norm, Dropout = 0.7, lr_decay = 0.002	1.439K	1.437K	0.9860	0.9459
conv2d LSTM	5	3	3	128	3	Batch Norm, Dropout = 0.7, lr_decay = 0.001	466K	465K	0.8308	0.8536
conv2d LSTM	7	3	3	128	3	Batch Norm, Dropout = 0.7, lr_decay = 0.001	526K	525K	0.8737	0.8872
conv2d LSTM	3	1	3	256	3	Batch Norm, Dropout = 0.7, lr_decay = 0.001	3.509K	3.507K	0.9875	0.9385
conv2d LSTM	3	1	3	384	3	Batch Norm, Dropout = 0.7, lr_decay = 0.001	6.235K	6.232K	0.9892	0.9335
conv2d LSTM	3	1	3	128	3	Batch Norm, Dropout = 0.7, lr_decay = 0.001, fft_length = 256, ft_stride = 64	914K	913K	0.9883	0.9029
conv2d LSTM	3	1	3	128	3	Batch Norm, Dropout = 0.7, lr_decay = 0.001, fft_length = 128, ft_stride = 64	652K	651K	0.9839	0.9319
conv2d LSTM	3	1	3	128	3	Batch Norm, Dropout = 0.7, lr_decay = 0.001, fft_length = 64, ft_stride = 32	521K	520K	0.9831	0.9272

Experiments were performed in Keras backed TensorFlow running on Google Cloud with NVidia K80 GPU for processing.

References

[1] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In International conference on machine learning, pages 173–182, 2016.

[2] Sercan O Arik, Markus Kliegl, Rewon Child, Joel Hestness, Andrew Gibiansky, Chris Fougner, Ryan Prenger, and Adam Coates. Convolutional recurrent neural networks for small-footprint keyword spotting. arXiv preprint arXiv:1703.05390, 2017.

[3] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. IEEE Signal processing magazine, 29, 2012.

[4] Kaggle. Tensorflow speech recognition challenge, 2019.

[5] Alex Oxenin. End-to-end baseline tf estimator lb 0.72, 2019.

[6] Tara Sainath and Carolina Parada. Convolutional neural networks for small-footprint keyword spotting. 2015.

[7] Pete Warden. Speech commands: A public dataset for single-word speech recognition. Dataset available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz, 2017.

[8] YundongZhang,NaveenSuda,LiangzhenLai,andVikasChandra.Hello edge:Keywordspotting on microcontrollers. arXiv preprint arXiv:1711.07128, 2017.