Keyword Spotting for Google Speech Command

Civil ML
CS 172B Project Presentation

Agenda

- Background
- Data Preprocessing & Dataset
- Model Architecture
- Improvement
- Next Step

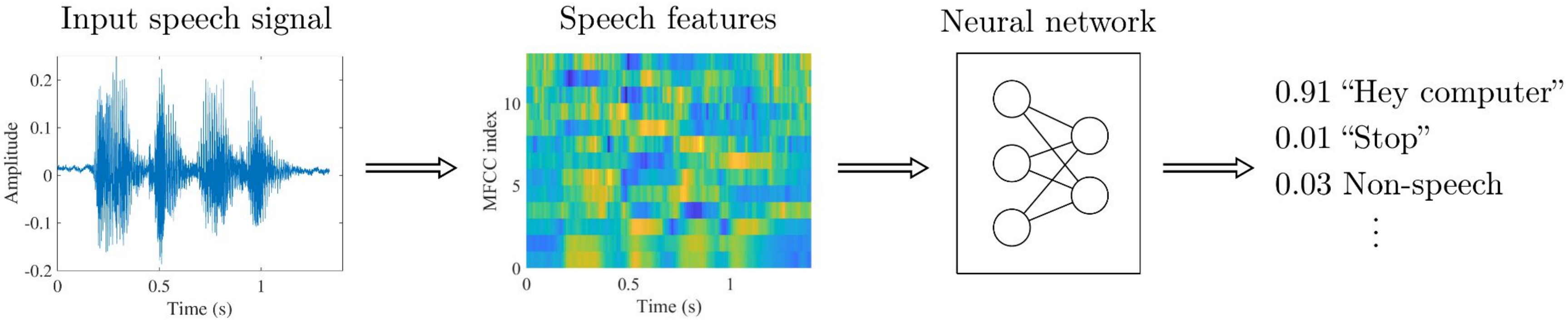
Background

Motivation

- With online resources drastically increasing in the era of information, processing videos and audios automatically become increasingly important, especially for children, who are likely to be exposed to explicit utterance, such as dirty words, violence, and sexual messages.
- Our original goal is to detect and filter out dirty utterances in a sequence of audios, but we are not able to find enough data like that currently, so we choose to detect common spoken words first.
- Knowing how powerful the neural networks are to classify data, we would like to build a neural network to detect specific words in audios, so that it is helpful for data filtering later.

What is keyword spotting?

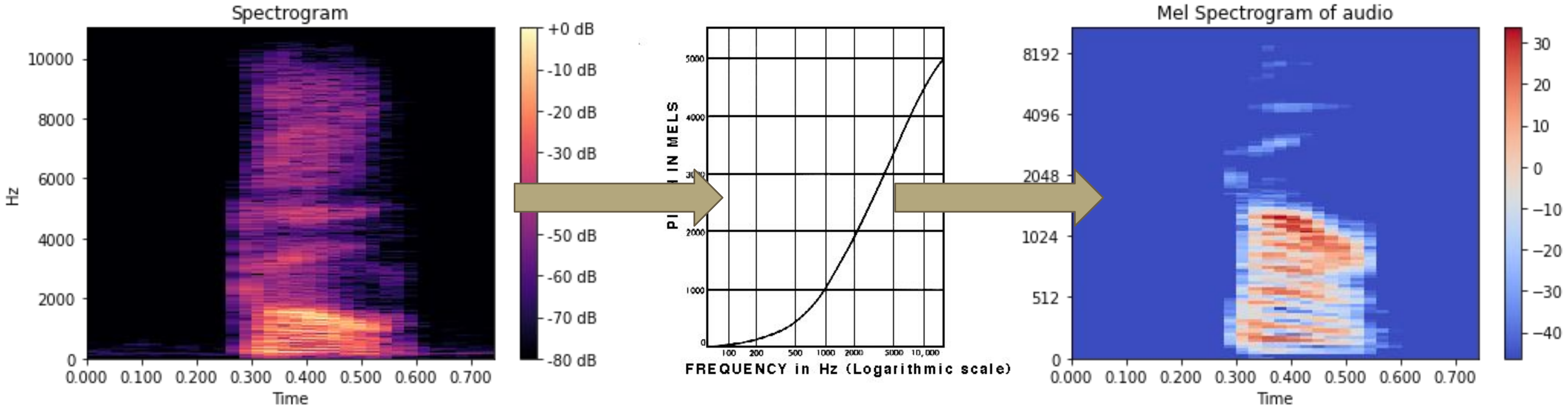
- In the context of speech processing
  - E.g. Special case: “Hey Siri”
- Task: Identification of keyword and recognize it in utterances
- Similar to image processing: Audio -> Spectrogram -> Extract Features
- Application: Google Assistant, Amazon Alexa, Apple Siri and self-driving car HCIs



Audio -> Spectrogram

- Spectrogram
  - X-axis: Time
  - Y-axis: Frequency
  - Color: Amplitude

- Mel Spectrogram
  - Human does not perceive audio in linear scale
    - Better perceived 500 ~ 1000 Hz
  - perform a log operation on frequencies to convert



Data Preprocessing & Dataset

Datasets

- Google Speech Command Dataset
  - Speech_commands_v0.02
  - An audio dataset which contains **30** 1-second spoken words collected from people with different demographics
  - Designed to help train and evaluate keyword spotting systems
  - Size: about **65,000** .wav audio
- Ted-Lium Dataset
  - English-language TED talks, with transcriptions, sampled at 16kHz.
  - Contains about 118 hours of speech in .sph audio

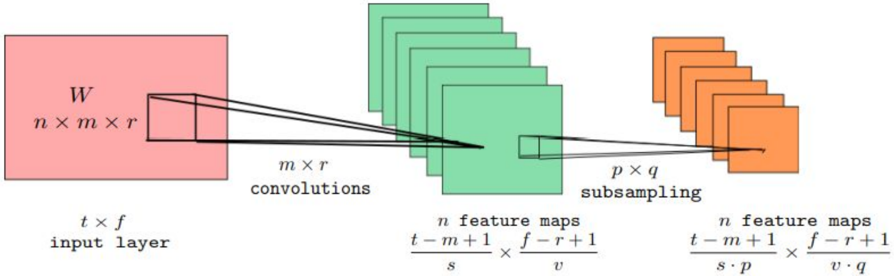
Data Preprocessing

1. Load Google Speech Command Audio
2. Filter out TED Talk with target command word
3. Chop TED Talk Audio into 1-second clips -> keep consistent
4. Convert all audio to Mel-Frequency Cepstral Coefficient (MFCC)
    - Coefficient that make up the Mel-Frequency Spectrogram
5. Divide the audio clips into 16 window segments, with an step = 4
6. Split Training/Testing Dataset with Ratio 8:2

Model Architecture

CNN Model Replication

- Given a paper published in 2014, convolutional neural networks (CNN) is an effective solution for small memory footprint keyword spotting task because of its limited parameters and multipliers.



Convolutional Neural Network for Small-footprint KWS. 2014.

- The architecture has 2 convolutional, one linear low-rank and one DNN layer.
- **Concerns**: the huge number of multiplies in the convolutional layers can be time consuming

type	m	r	n	p	q	Par.	Mul.
conv	20	8	64	1	3	10.2K	4.4M
conv	10	4	64	1	1	164.8K	5.2M
lin	-	-	32	-	-	65.5K	65.5K
dnn	-	-	128	-	-	4.1K	4.1K
softmax	-	-	4	-	-	0.5K	0.5K
Total	-	-	-	-	-	244.2K	9.7M

Table 1: CNN Architecture for `cnn-trad-fpool3`

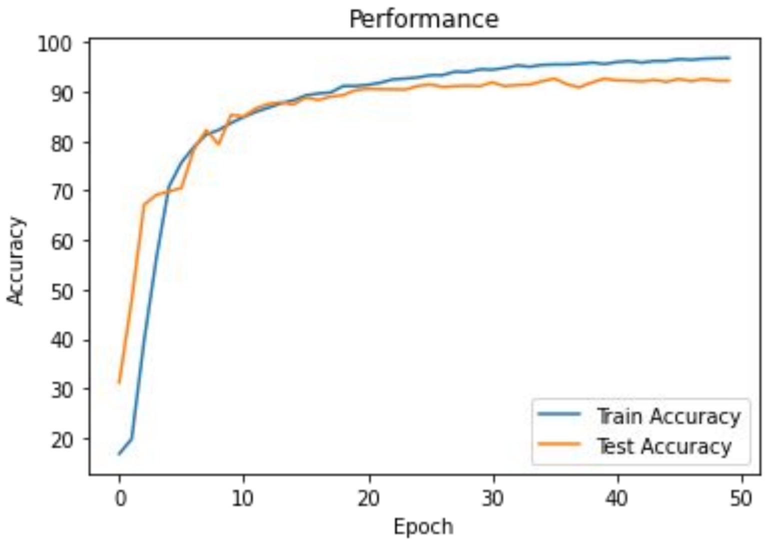
CNN Model Replication

- n_Classes: 5
- Training Dataset: 16,846
- Validating Dataset: 4212
- Batch Size: 100
- Epoch: 50
- Drop out: 0.3
- Learning Rate: 0.001
- Optimizer: Adam
- Criterion: Cross Entropy Loss

Architecture

```
NN(
  (conv1): Conv1d(1, 8, kernel_size=(13,), stride=(1,))
  (dropout1): Dropout(p=0.3, inplace=False)
  (conv2): Conv1d(8, 16, kernel_size=(11,), stride=(1,))
  (dropout2): Dropout(p=0.3, inplace=False)
  (conv3): Conv1d(16, 32, kernel_size=(9,), stride=(1,))
  (dropout3): Dropout(p=0.3, inplace=False)
  (conv4): Conv1d(32, 64, kernel_size=(7,), stride=(1,))
  (dropout4): Dropout(p=0.3, inplace=False)
  (fc1): Linear(in_features=6080, out_features=256, bias=True)
  (dropout5): Dropout(p=0.3, inplace=False)
  (fc2): Linear(in_features=256, out_features=128, bias=True)
  (dropout6): Dropout(p=0.3, inplace=False)
  (fc3): Linear(in_features=128, out_features=31, bias=True)
)
```

Parameter Reference: [Speech Command Classification using PyTorch and torchaudio]

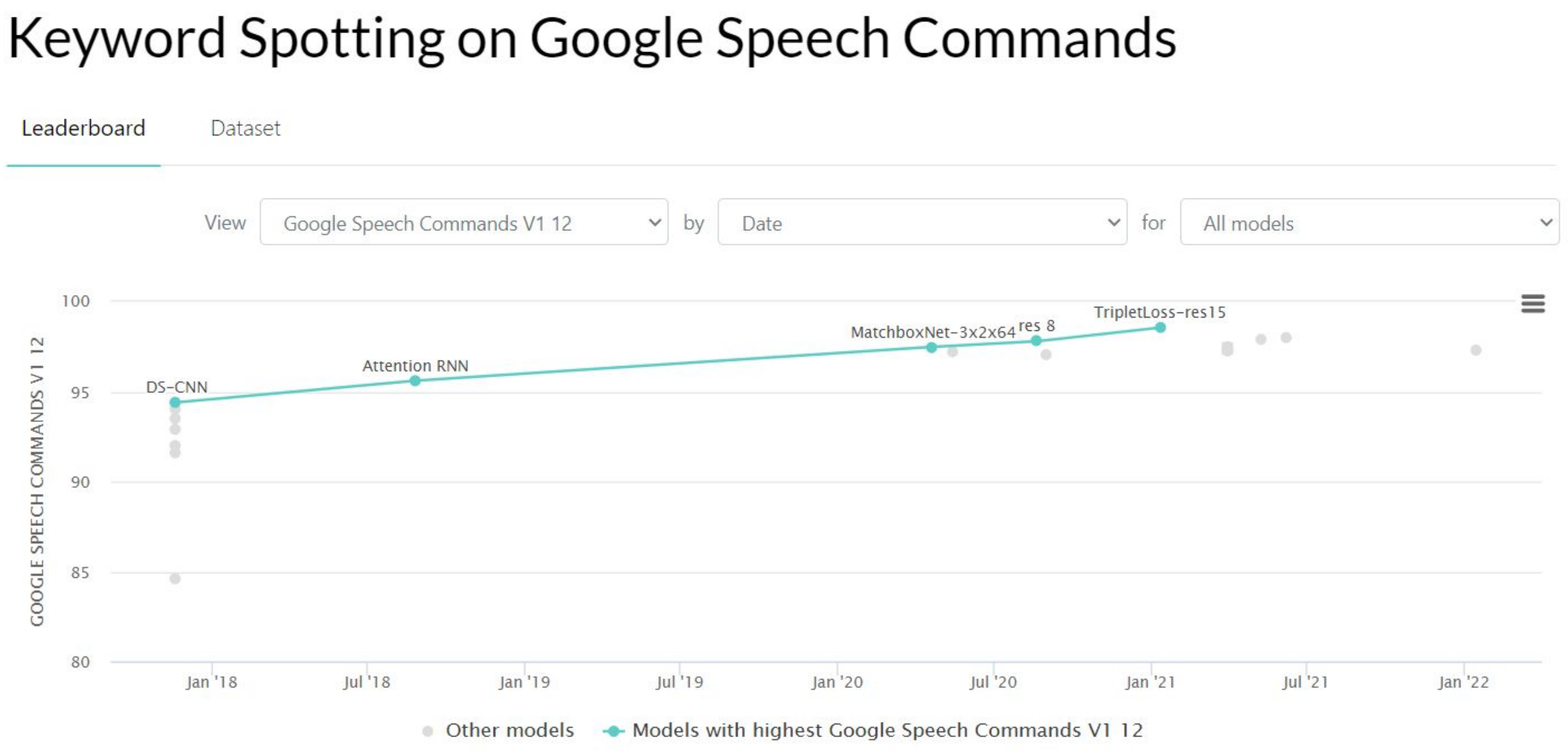


Best accuracy rate of the test dataset is about 93%

CNN Performance

Model Improvement

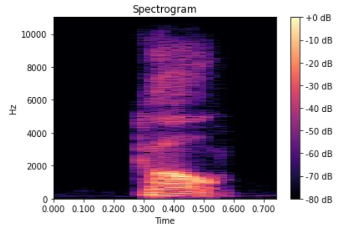
Even Better?

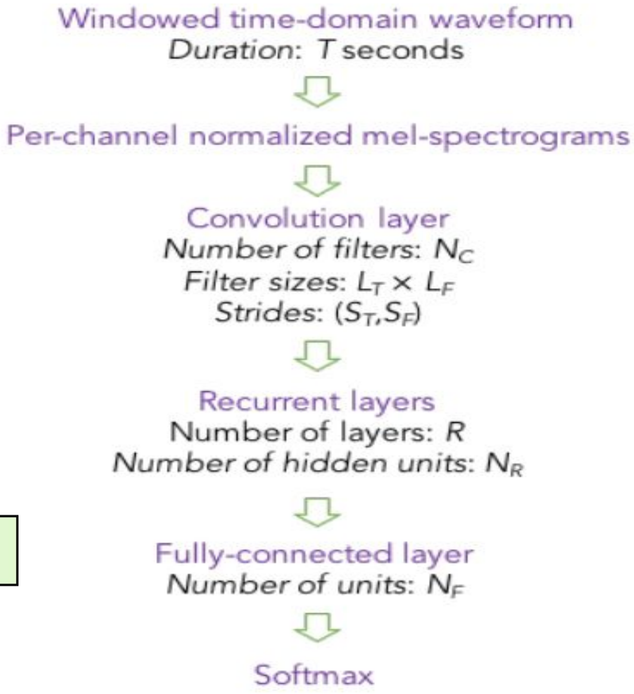

Keyword Spotting on Google Speech Commands
Leaderboard
Dataset
View
Google Speech Commands V1 12
by
Date
for
All models
100
95
90
85
80
GOOGLE SPEECH COMMANDS V1 12
DS-CNN
Attention RNN
MatchboxNet-3x2x64
res 8
TripletLoss-res15
Jan '18
Jul '18
Jan '19
Jul '19
Jan '20
Jul '20
Jan '21
Jul '21
Jan '22
Other models
Models with highest Google Speech Commands V1 12

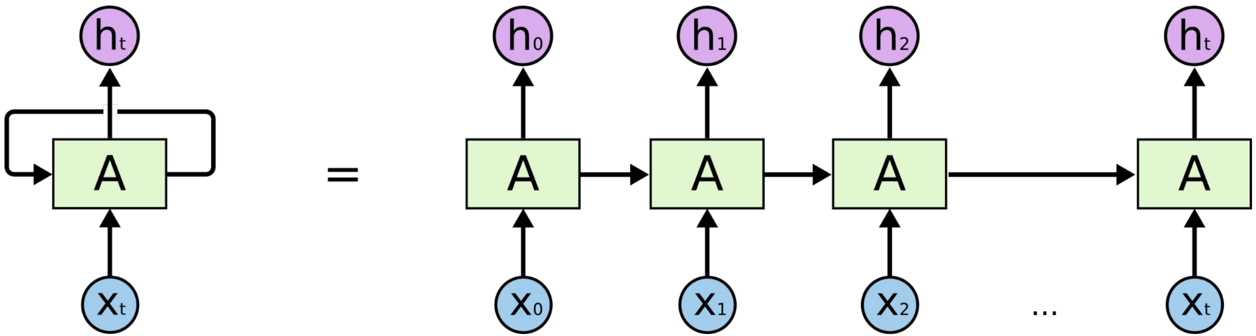
CRNN Model

- According to another paper by Google in 2017, using recurrent neural networks (**RNNs**) on the basis of CNN allows modeling the contexts in the entire frame **with wide filters or great depths**.
- **RNNs** take advantages of modeling sequential data and are effective in many machine learning tasks, like text classification and NLP.

Convolutional Recurrent Neural Networks for Small-Footprint Keyword Spotting. 2017.

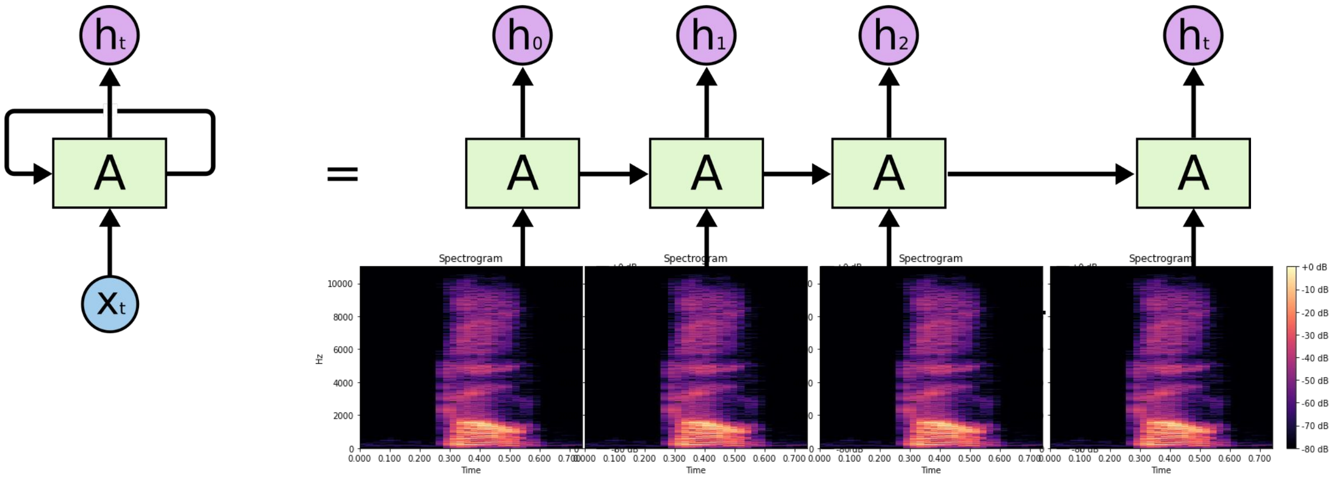


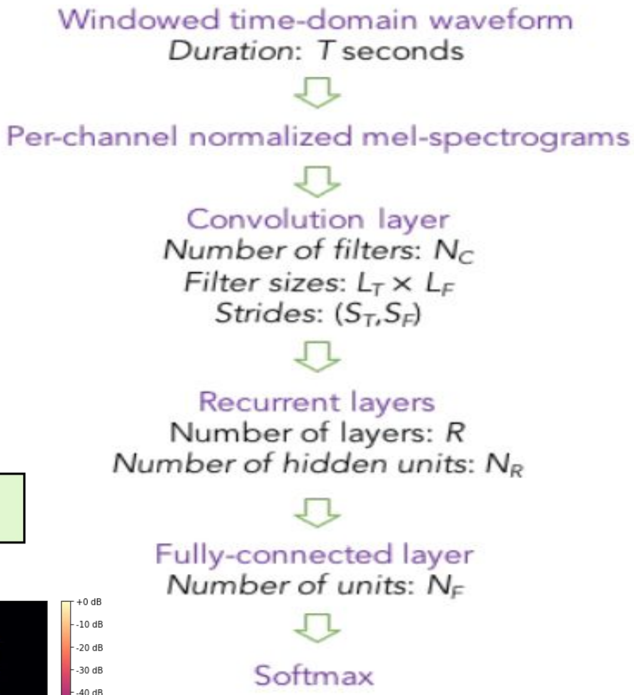




CRNN Model

Convolutional Recurrent Neural Networks for Small-Footprint Keyword Spotting. 2017.





CRNN Model Implementation

- Hyperparameters:
  - batch_size = 100
  - input window segment size: (20, 16)
  - input_channels = 1
  - hidden_channels = 5
  - CNN_out_channel = 2
  - num_classes = 11
  - learning_rate = 0.05

```
CRNN(
  (conv1): Conv2d(1, 5, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), padding_mode=reflect)
  (conv2): Conv2d(5, 2, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), padding_mode=reflect)
  (rnn): RNN(570, 570, batch_first=True)
  (linear): Linear(in_features=570, out_features=10, bias=True)
  (softmax): Softmax(dim=1)
  (maxpool): MaxPool2d(kernel_size=2, stride=1, padding=0, dilation=1, ceil_mode=False)
  (relu): ReLU()
)
```

Next Step

What else can we do?

Feed in more negative examples during training (TED-LIUM dataset)

- The model currently is only exposed to select keywords and background noise (natural or mathematically generated background noise), but no background noise in conversational settings.

More robust testing

- Instead of using Google Speech Command dataset for testing, which only contains utterance of 30 keywords, use longer conversation audio to identify keyword in a conversation.

What else can we do?

Use LSTM?

- It's been shown that some LSTM + CNN models achieve outstanding KWS accuracy. Longer term memory can be beneficial because the previous phoneme, word, or even sentence can provide cues of the likelihood of the next word.

CNN - LSTM with Attention Mechanism

- A more advanced solution is to use RNNs with attention mechanism.
    - **Conv2Ds:** extract local relations
    - **Bidirectional LSTMs**: be able to capture long-term dependency of audios
    - **Attention Layer**: find the most relevant part of LSTM output
    - **Dense Layers:** classify the keywords