Convolutional Neural Networks for Small-footprint Keyword Spotting

Wafaa Mohammed

Keyword spotting (KWS)

- In the context of speech processing, keyword spotting is the process of identifying keywords in speech utterances.
- personal assistants such as Google Now, Apple’s Siri, Microsoft’s Cortana and Amazon’s Alexa, all utilize speech recognition to interact with them.
- KWS system must have a small memory footprint and low computational power.
- In 2015, the KWS system at Google used a (DNN) which outperformed Hidden Markov Model system, which was a commonly used technique for KWS.
- This paper investigates the use of CNN for KWS for 2 reasons:

  - DNNs ignore input topology.

  - DNNs are not explicitly designed to model translational variance within speech signals.

Paper Implementation

Dataset:

- They collected a dataset of 14 instances (keywords), and a much larger set for “negative” examples.
- I used the publicly available google speech commands dataset, I used 8 instances, half of them are considered keywords, and the other half is negative samples.
- The data is represented as spectrograms (a visual representation of the spectrum of frequencies of a signal as it varies with time) to be input to the CNN.

Models:

Traditional CNN architecture:

Has two issues:

1- Large number of multiplies.

2- Large number of parameters.

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`

Limiting multiplies

1- one convolutional layer, pooling in frequency

type	m	r	n	p	q	Params	Mult
conv	32	8	54	1	3	13.8K	456.2K
linear	-	-	32	-	-	19.8K	19.8K
dnn	-	-	128	-	-	4.1K	4.1K
dnn	-	-	128	-	-	16.4K	16.4K
softmax	-	-	4	-	-	0.5K	0.5K
Total	-	-	4	-	-	53.8K	495.6K

Table 2: CNN Architecture for `cnn-one-fpool3`

2- striding in frequency

model	m	r	n	s	v	Params	Mult
(a)	32	8	**186**	1	4	47.6K	428.5K
(b)	32	8	**336**	1	8	86.6K	430.1K

Table 3: CNN for (a) `cnn-one-fstride4` and (b) `cnn-one-fstride8`

Limiting parameters

1- striding in time

model	layer	m	r	n	s	q	Params
cnn-tstride2	conv	16	8	78	**2**	3	10.0K
	conv	9	4	78	1	1	219.0K
	lin	-	-	32	-	-	20.0K
cnn-tstride4	conv	16	8	100	**4**	3	12.8K
	conv	5	4	78	1	1	200.0K
	lin	-	-	32	-	-	25.6K
cnn-tstride8	conv	16	8	126	**8**	3	16.1K
	conv	5	4	78	1	1	190.5K
	lin	-	-	32	-	-	32.2K

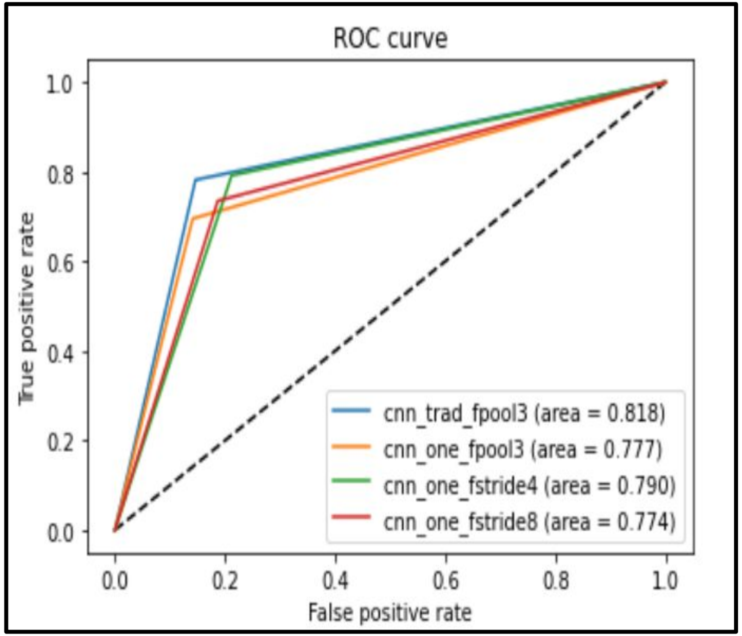
Table 4: CNNs for Striding in Time

2- pooling in time

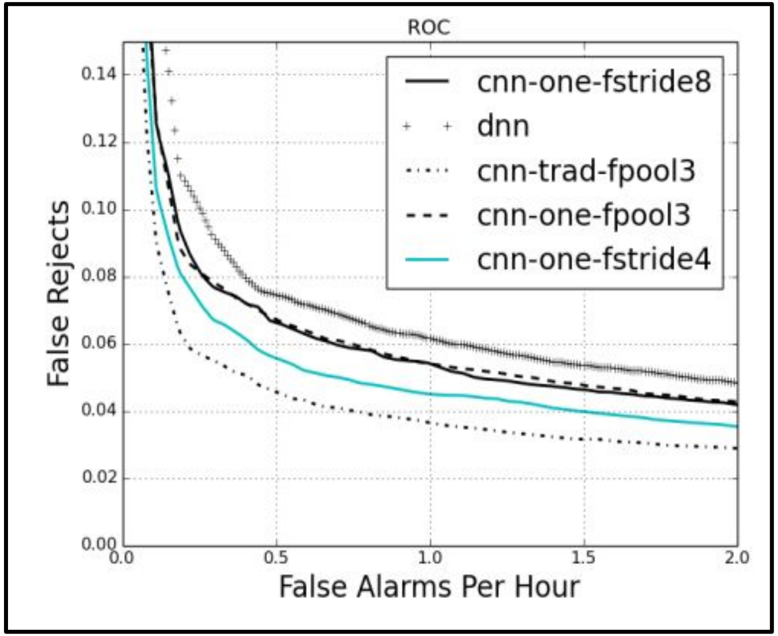
model	layer	m	r	n	p	q	Params
cnn-tpool2	conv	21	8	94	**2**	3	5.6M
	conv	6	4	94	1	1	1.8M
	lin	-	-	32	-	-	65.5K
cnn-tpool3	conv	15	8	94	**3**	3	7.1M
	conv	6	4	94	1	1	1.6M
	lin	-	-	32	-	-	65.5K

Table 5: CNNs for Pooling in Time

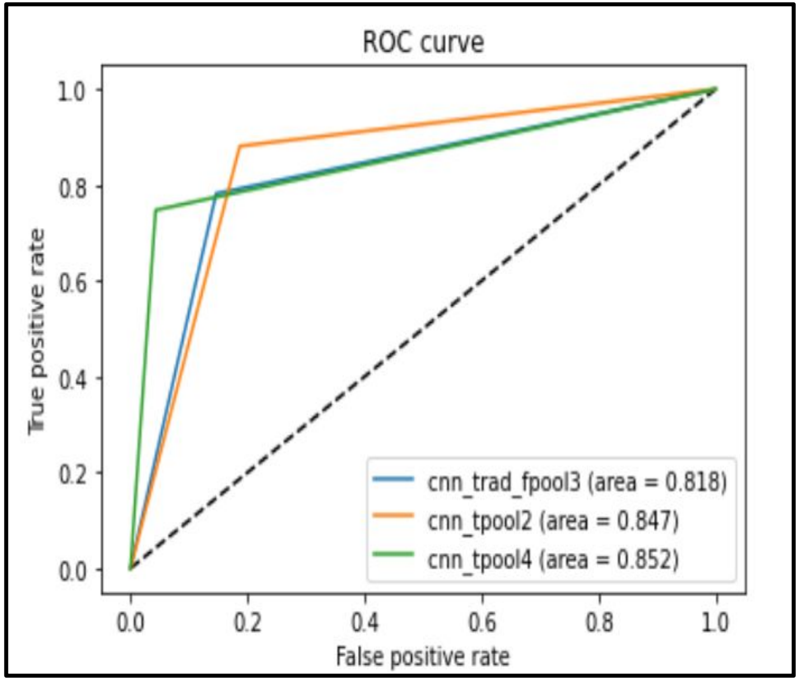
Results

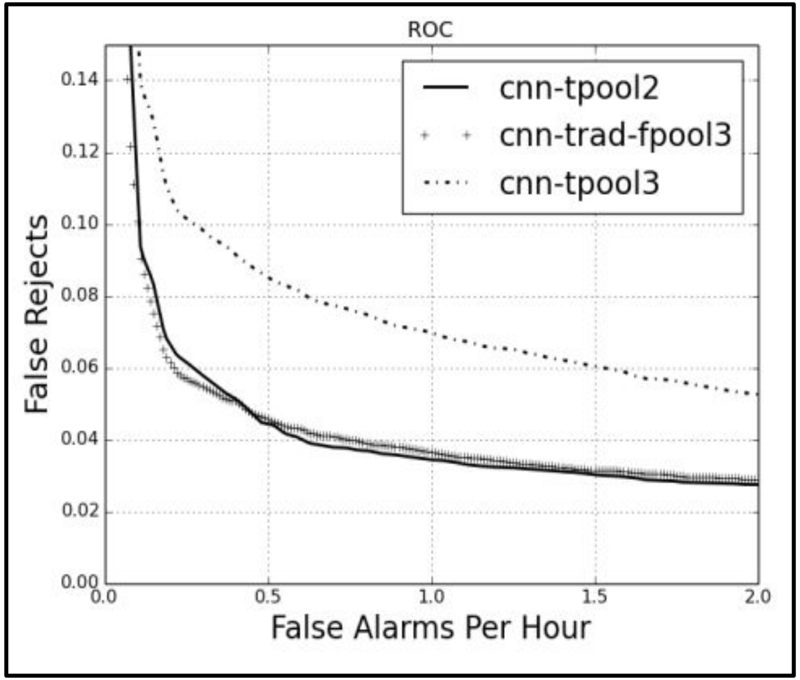


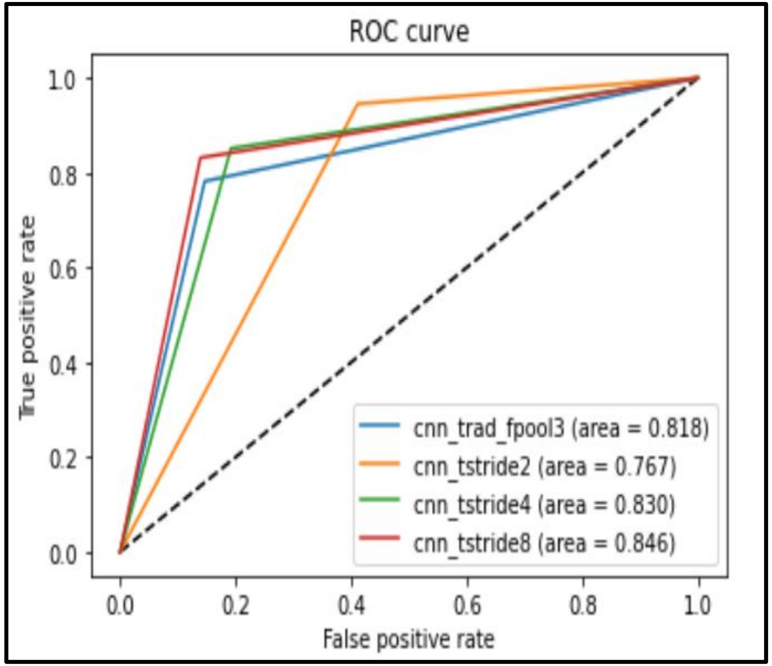
ROC curve
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
cnn_trad_fpool3 (area = 0.818)
cnn_one_fpool3 (area = 0.777)
cnn_one_fstride4 (area = 0.790)
cnn_one_fstride8 (area = 0.774)

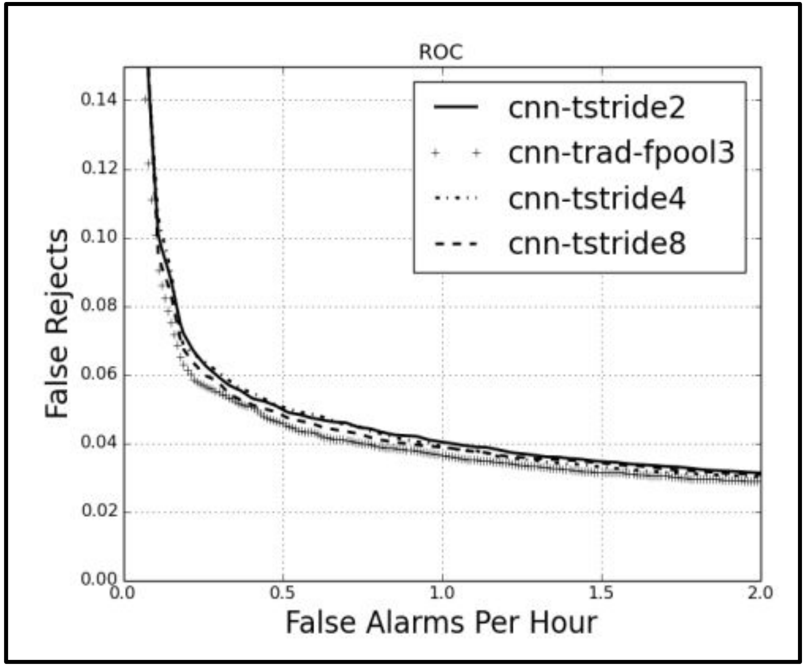


ROC
False Rejects
False Alarms Per Hour
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.0
0.5
1.0
1.5
2.0
cnn-one-fstride8
dnn
cnn-trad-fpool3
cnn-one-fpool3
cnn-one-fstride4


ROC curve
True positive rate
False positive rate
cnn_trad_fpool3 (area = 0.818)
cnn_tpool2 (area = 0.847)
cnn_tpool4 (area = 0.852)
0.0
0.2
0.4
0.6
0.8
1.0


ROC
False Rejects
False Alarms Per Hour
cnn-tpool2
cnn-trad-fpool3
cnn-tpool3
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.0
0.5
1.0
1.5
2.0


ROC curve
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
cnn_trad_fpool3 (area = 0.818)
cnn_tstride2 (area = 0.767)
cnn_tstride4 (area = 0.830)
cnn_tstride8 (area = 0.846)


ROC
False Rejects
False Alarms Per Hour
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.0
0.5
1.0
1.5
2.0
cnn-tstride2
cnn-trad-fpool3
cnn-tstride4
cnn-tstride8

Conclusions

- When limiting multiplies, shifting convolutional filters in frequency results in a better performance than DNN.
- When limiting parameters, pooling in time results in improvement of DNN results.

Future steps

- Adding noise to the input data and investigating the models' performance, two ways for adding noise: Gaussian noise, or real world noise.
- Using an imbalanced dataset to model the real world situation of KWS.
- Creating a generic model that accepts parameters to generate different architectures.
- Using python scripts.

Thank you!