Ultra-Low Footprint Keyword-Spotting

Adam Fuks, Frans Widdershoven, NXP Semiconductors

Introduction

Low-footprint (compute/memory) Keyword Spotting(KWS) is widespread in edge compute devices.

This work is investigating how to implement smaller efficient KWS NNs which can run on smallest MCUs. Including LPC55S69 which we used in this work.

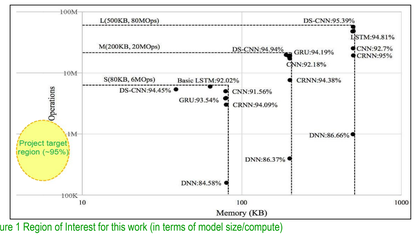


Figure 1 Region of Interest for this work (in terms of model size/compute)

Temporal Convolution Network (TCN)

TCNs were invented [1] as a method for extraction of features using temporal context. **Fully Connected layers** are used at **increasing dilation** to extract features for an **ever-increasing time period** (receptive field). The TCN takes spectral features as input, and outputs a probability of a Keyword in the span of the receptive field.

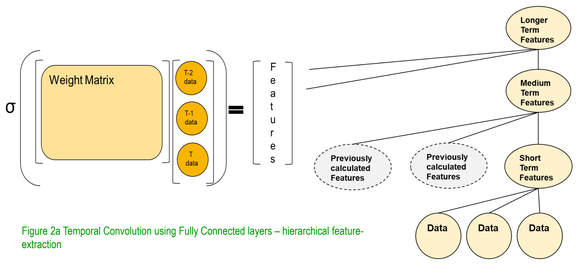


Figure 2a Temporal Convolution using Fully Connected layers – hierarchical feature-extraction

Convolution is achieved by re-use of FC matrices at each timestep.

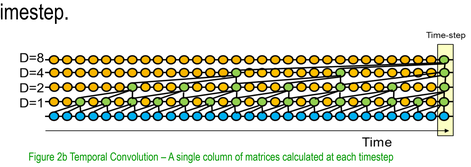


Figure 2b Temporal Convolution – A single column of matrices calculated at each timestep

Feature Extraction

FFT analysis of audio can be seen as a form of compression. For example in case of **256-pt FFT**, the power spectrum is comprised of 129 unique coefficients (FFT of Real signal). If the microphone is sampling at 16kHz, then only the bottom **64 coefficients** are needed for **4kHz bandwidth** (which are key for speech recognition). The DC co-efficient can be thrown away.

Fig 3 shows 63 coefficients representing 4kHz bandwidth. However, due to physics of harmonics of human voice, recognition can benefit from **higher resolution for lower frequencies** while higher frequencies do no require such resolution. We found that using overlapping (50%) windows of 512 samples and **rectangular log-spaced windowing**, the recognition could be improved.

We allow FFT coefficients **1-40 to go directly to NN**. Following 88 FFT Coefficients are added (rectangular windowing) in **logarithmically increasing time-spans into 23** more coefficients.

Figures 3 and 4 below shows same speech time window under both schemes.

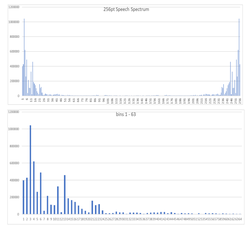


Figure 3 Using 256pt FFT @16KHz to examine bottom 4kHz of spectrum

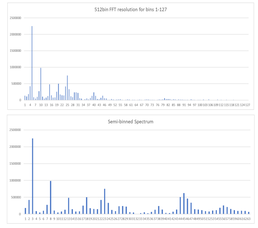


Figure 4 Using 512pt FFT @16KHz to look at bottom 4kHz of bandwidth – using our rectangular window binning

Next, we feed the selected features to a **learnt binning FC layer**. In order to ensure **normalized binning** between each binning kernel, we use **Softmax-ed** matrix rows. This means that all weights are between 0 and 1, and each row of the weight matrix adds up to 1.0.

The output of this binning layer is then passed through a **ln(x)** activation function.

After the binning layer (which acts on a single timestep of FFT spectral features), we use a TCN to search for the keyword.

Volume Independent Analysis

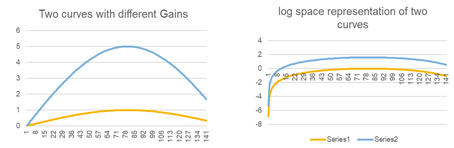


Figure 6 Amplitude of an arbitrary signal in linear and Log-scale

If we consider the spectral content of an audio fragment to be equal to a global loudness G times a sequence of local content vectors $f(t)$, then the input of the m-th neuron of the first TCN layer at time step t is

$$Input_n(t) = Bias_n + \sum_n W_{m,n} \ln\left(G f_n(t)\right)$$

$$= Bias_m + \sum_n W_{m,n} \ln(G) + \sum_n W_{m,n} \ln(f_n(t))$$

So, if $\sum_n W_{m,n} = 0$ then $Input_m(t)$ is independent of the momentary audio loudness G.

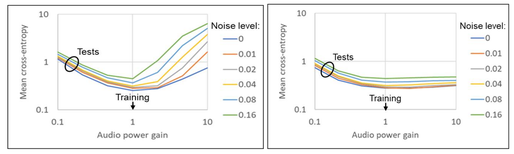
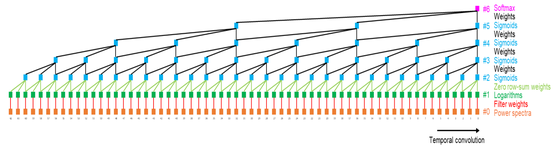


Figure 8 Training without zero-sum rows (left), Training with zero-sum rows (right)

Imposing a **zero-sum constraint** for each of the rows of the weight matrix of our **first layer of TCN** helps improve **volume independence** of our recognition significantly.

Zero-row sum can be achieved simply in NN frameworks by adding to the NN graph a tensor of the average of each row and subtracting it from the trained matrix rows. This ensures that used row adds up to zero. Back-propagation for learning is able to trace cost gradient through this structure. Thus no additional term is needed for the NN Cost function in training.

Putting It All Together



- ~3000 accurately labeled Alexa utterances
- ~3400 non-target sentences
- >1000 different background noices
- 5 × augmentation by pitch shifting (-2,-1,0,+1,+2 semitones)

TCN	Log_1	Sigm_2	Sigm_3	Sigm_4	Sigm_5	Softm_6	Test_cost	# weights	# nodes
1	21	21	21	21	21	2	0.0302	6678	107
2	8	21	21	21	21	2	0.0283	5040	94
3	8	8	21	21	21	2	0.0291	3909	81
4	8	8	8	21	21	2	0.0307	2778	68
5	8	8	8	8	21	2	0.0325	1647	55
6	8	8	8	8	8	2	0.0351	1296	42

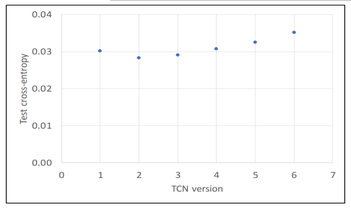


Figure 8 Complete TCN and Effects of Network Size on Test Set Cross-Entropy

Depending on the size of training set, there exists an **optimal network size**. Growing larger may increase training cross-entropy but does not increase validation (Test) cross-entropy. Using a larger network may make the TCN begin to overfit.

Decimated TCN

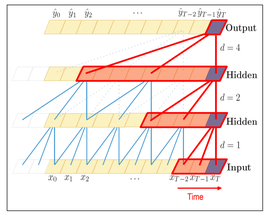


Figure 9 Working Memory Growth as TCN Dilation Increases [3]

Fig 9 illustrates how working memory (span of historic results to be kept) for TCN grows as dilation increases.

Using the intuition that human speech is comprised of certain spectral features which require short term frequency attention and separately a set of longer term context, we added a decimation layer in the middle of our TCN.

The decimation is achieved by max-pooling each feature over 4 timesteps :

- TCN is **less sensitive** to perfect **alignment** of features
- **Compute** load of latter layers is **reduced 4x** (since we now only compute the final layers once every 4 timesteps
- Working **memory is reduced 4x** for latter layers

Noting that as per Figure 9, a TCN's working memory increases in the latter layers, our decimation is able to reduce the working memory consumption of the most consuming portion.

Furthermore, accuracy is proportional to parameter count but we don't need to compute using all parameters at each timestep.

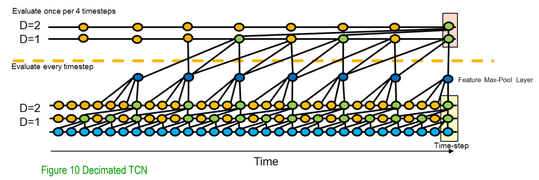


Figure 10 Decimated TCN

Results and Conclusions

Quantized (8bit) Decimated Alexa TCN - 5k Parameters, 1kB RAM, 1MHz compute load on NXP LPC55S69

(Figures in percentages for Accept Rates)

	Acceptance Threshold			
Fileset	0.75	0.8	0.85	0.9
Alexa US Region TEST	98.26923	97.88462	97.11538	95
Alexa IN Region TEST	90.41096	89.26941	84.93151	77.16895

Counter Test

	Total number of False Accepts				Total number of files
24 Hour testfile	71	42	20	4	29603
SpeechCommandset v1/six	42	26	11	3	2766

Table 1 Results of miniature Alexa Keyword Spotter

Our work has shown that it is possible to design very small TCNs to achieve fairly good recognition rates. They may be used standalone or as a smart and effective pre-detector of Keyword as a decision point to turn on a larger network to do final check.

Several novel methods we have presented in this work are :

- **Softmax-weight rows** for learnt frequency **binning**
- Ensuring **volume independence** for speech recognition through **zero-sum kernels**
- **Decimated TCN** to significantly reduce compute and memory footprint (**1.2MHz** using Cortex-M33 and PowerQuad)

References

1. Temporal Convolutional Networks for Action Segmentation and Detection, Colin Lea, et al., 2016
2. Google Speech Command Set : https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html
3. https://arxiv.org/pdf/1803.01271.pdf