Keyword Spotting with hardware constrains

Xingyi Yang
Kneron. Inc

Outline

- Keywords Spotting
  - Introduction
    - Tasks & Overall pipeline
  - Typical methods
    - Feature
    - Traditional: Template Matching, HMM
    - RNN, DNN, CNN, CRNN
  - Performance comparison under hw constraints

Outline

- Model Compression for Keywords Spotting
  - Quantization
  - Knowledge distillation
  - SVD Low-rank matrices
- Inspiration

Part I

Introduction

Keyword Spotting: Task

audio **classification** problem:

The goal of **keyword spotting(KWS)** is to detect a relatively small set of predefined keywords in a stream of user utterances

Keyword Spotting: Pipeline

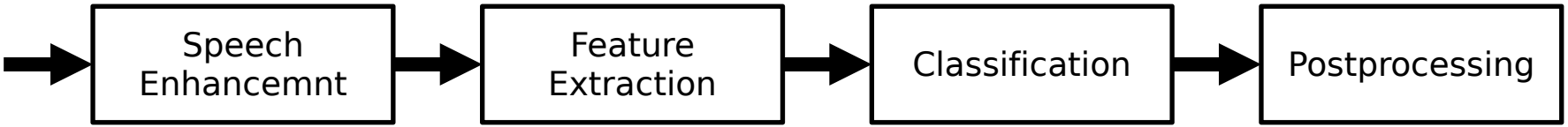


Fig 1 Keyword Spotting Pipeline

Kneron

Part I

Methodology review

Keyword Spotting: Feature

Audio of length L

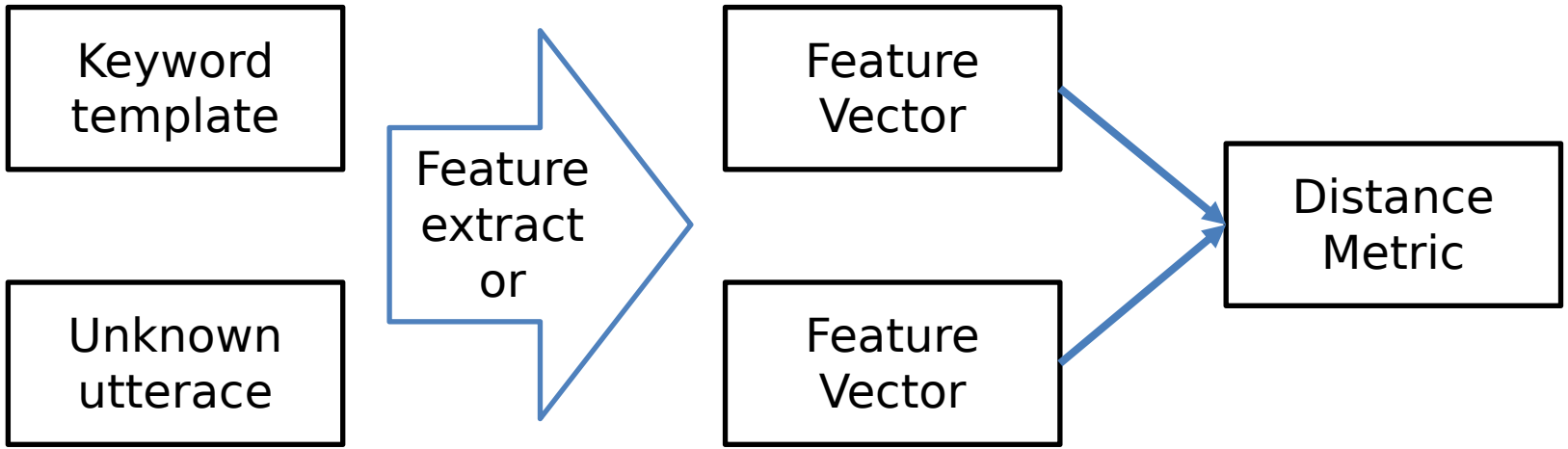
Devided into frames of length l and stride s
Frame number $T = \frac{L-l}{s}$

Each frame becomes
a feature vector of length F
Totally $T \times F$ input

Log-mel filter bank energies (LFBE)
Mel-frequency cepstral coefficients (MFCC)
Raw Audio

Keyword Spotting: Template Matching

Template Matching



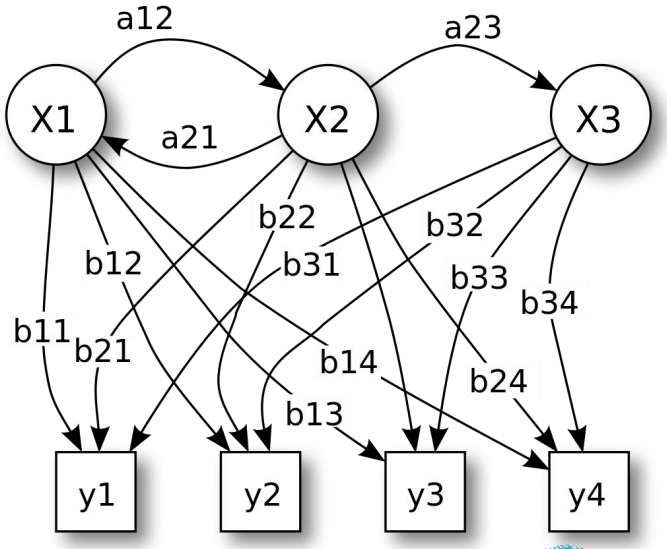
Alex, John Sahaya Rani, and Nithya Venkatesan. "Spoken Utterance Detection Using Dynamic Time Warping Method Along With a Hashing Technique." International Journal of Engineering and Technology (IJET) (2014).

Kneron

Keyword Spotting: HMM

Hidden Markov Models

Fig 2. Probabilistic parameters of a hidden Markov model (example)
X — states
y — possible observations
a — state transition probabilities
b — output probabilities



Wilpon, Jay G., et al. "Automatic recognition of keywords in unconstrained speech using hidden Markov models." IEEE Transactions on Acoustics, Speech, and Signal Processing 38.11 (1990): 1870-1878.

Kneron

Keyword Spotting: RNN

RNN

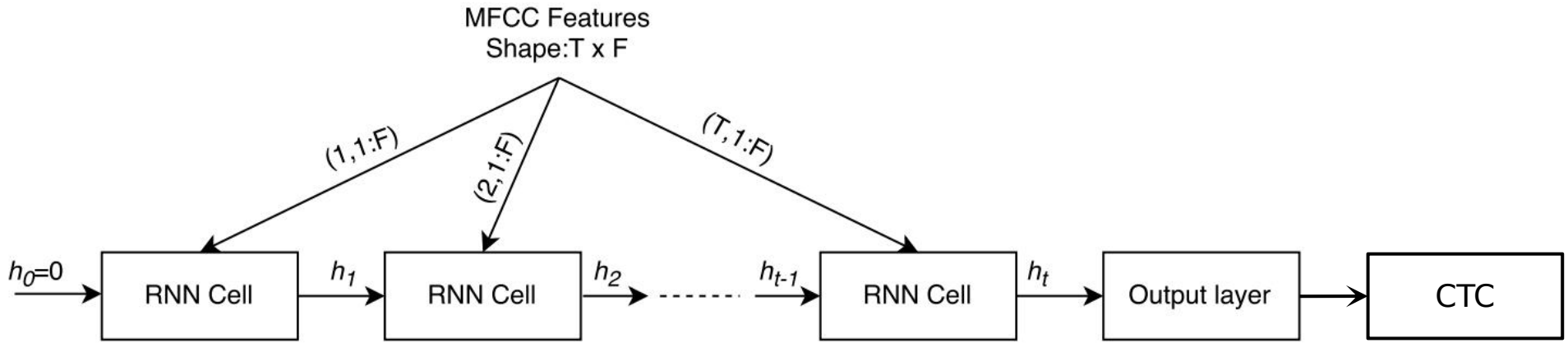


Fig 3 Model architecture of RNN for KWS

Sun, Ming, et al. "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting." 2016 IEEE Spoken Language Technology Workshop (SLT). IEEE, 2016.

Kneron

Keyword Spotting: DNN

DNN

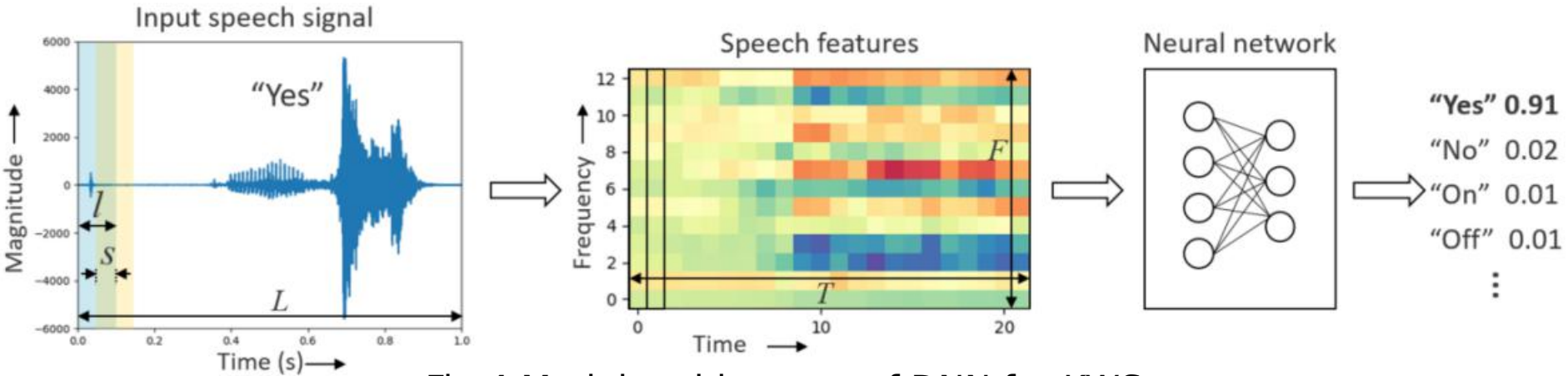


Fig 4 Model architecture of DNN for KWS

Chen, Guoguo, Carolina Parada, and Georg Heigold. "Small-footprint keyword spotting using deep neural networks." 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014.

Kneron

Keyword Spotting: DNN

- Posterior smoothing.
  - $p'_{ij} = \frac{1}{j - h_{smooth}} \sum_{k=h_{smooth}}^{j} p_{ik}$
  - where $h_{smooth} = \max\{1, j - w_{smooth} + 1\}$, w_{smooth} is the window size.
- Confidence
  - $C = \sqrt[n-1]{\prod_{i}^{n-1} \max_{h_{smooth}<k<j} (p_{ik})}$
  - where $h_{max} = \max\{1, j - w_{max} + 1\}$, w_{max} is the window size.

Keyword Spotting: CNN

CNN

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`

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`

Low-rank linear layer

Sainath, Tara, and Carolina Parada. "Convolutional neural networks for small-footprint keyword spotting." (2015).

Keyword Spotting: CRNN

CRNN

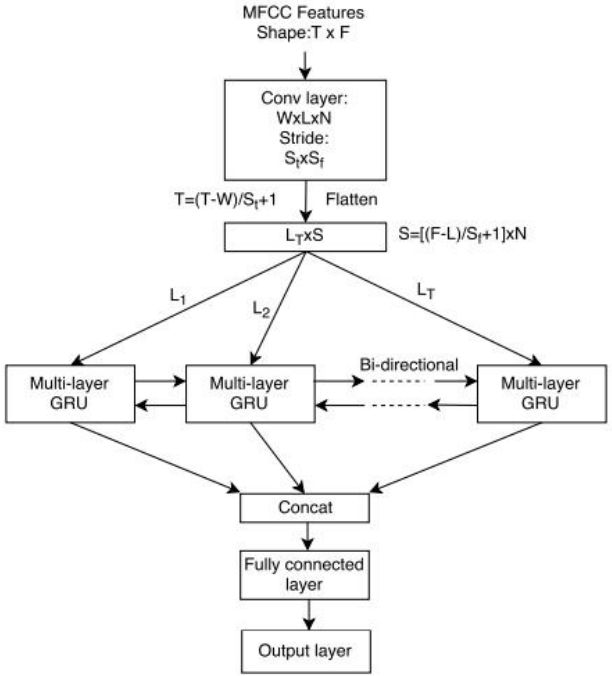


Fig 4 Model Architecture of CRNN

Arik, Sercan O., et al. "Convolutional recurrent neural networks for small-footprint keyword spotting." arXiv preprint arXiv:1703.05390 (2017).

Kneron

Part I

Performance comparison under HW constraints

Keyword Spotting: Performance

NN Architecture	Accuracy	Memory	Operations
DNN [5]	84.3%	288 KB	0.57 MOps
CNN-1 [6]	90.7%	556 KB	76.02 MOps
CNN-2 [6]	84.6%	149 KB	1.46 MOps
LSTM [8]	88.8%	26 KB	2.06 MOps
CRNN [7]	87.8%	298 KB	5.85 MOps

Table 3: Neural network model accuracy. CNN-1, CNN-2 are (cnn-trad-fpool3, cnn-one-fstride4)

Zhang, Yundong, et al. "Hello edge: Keyword spotting on microcontrollers." arXiv preprint arXiv:1711.07128 (2017).

Kneron

Keyword Spotting: Performance

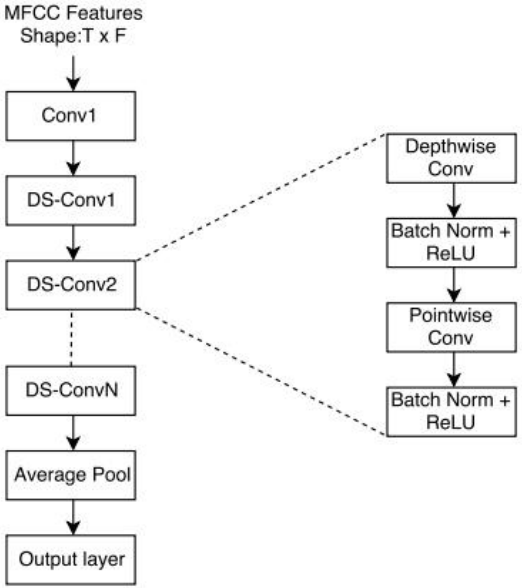


Fig 5 Depthwise separable CNN architecture

NN size	NN memory limit	Ops/inference limit
Small (S)	80 KB	6 MOps
Medium (M)	200 KB	20 MOps
Large (L)	500 KB	80 MOps

Table 3 Neural network (NN) classes for KWS models

NN model	Model hyperparameters
DNN	Number of fully-connected (FC) layers and size of each FC layer
CNN	Number of Conv layers: features/kernel size/stride, linear layer dim., FC layer size
Basic LSTM	Number of memory cells
LSTM	Number of memory cells, projection layer size
GRU	Number of memory cells
CRNN	Conv features/kernel size/stride, Number of GRU and memory cells, FC layer size
DS-CNN	Number of DS-Conv layers, DS-Conv features/kernel size/stride

Table 4 Neural network searchable hyperparameters

Keyword Spotting: Performance

NN model	S(80KB, 6MOps)			M(200KB, 20MOps)			L(500KB, 80MOps)		
	Acc.	Mem.	Ops	Acc.	Mem.	Ops	Acc.	Mem.	Ops
DNN	84.6%	80.0KB	158.8K	86.4%	199.4KB	397.0K	86.7%	496.6KB	990.2K
CNN	91.6%	79.0KB	5.0M	92.2%	199.4KB	17.3M	92.7%	497.8KB	25.3M
Basic LSTM	92.0%	63.3KB	5.9M	93.0%	196.5KB	18.9M	93.4%	494.5KB	47.9M
LSTM	92.9%	79.5KB	3.9M	93.9%	198.6KB	19.2M	94.8%	498.8KB	48.4M
GRU	93.5%	78.8KB	3.8M	94.2%	200.0KB	19.2M	94.7%	499.7KB	48.4M
CRNN	94.0%	79.7KB	3.0M	94.4%	199.8KB	7.6M	95.0%	499.5KB	19.3M
DS-CNN	94.4%	38.6KB	5.4M	94.9%	189.2KB	19.8M	95.4%	497.6KB	56.9M

Table 5: Summary of best neural networks from the hyperparameter search.

Zhang, Yundong, et al. "Hello edge: Keyword spotting on microcontrollers." arXiv preprint arXiv:1711.07128 (2017).

Keyword Spotting: Performance

NN model	32-bit floating point model accuracy			8-bit quantized model accuracy		
	Train	Val.	Test	Train	Val.	Test
DNN	97.77%	88.04%	86.66%	97.99%	88.91%	87.60%
Basic LSTM	98.38%	92.69%	93.41%	98.21%	92.53%	93.51%
GRU	99.23%	93.92%	94.68%	99.21%	93.66%	94.68%
CRNN	98.34%	93.99%	95.00%	98.43%	94.08%	95.03%

Table 6: Accuracy comparison of representative 8-bit quantized networks with full-precision networks. Quantized network is either same or marginally better.

Zhang, Yundong, et al. "Hello edge: Keyword spotting on microcontrollers." arXiv preprint arXiv:1711.07128 (2017).

Kneron

Part II

Model Compression for Keywords Spotting

Model Compression:Quantization

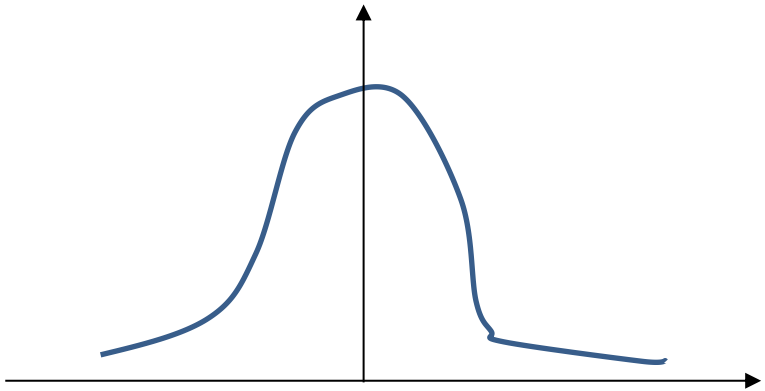


Fig 6 A example weight histograms for neural network

$$Q^*(x) = argmin_{Q(x)} \frac{1}{n} \sum_{i}^{n} h(x_i) dis(x_i, Q(x_i))$$

- $Q^*(x)$ is the optimal quantization function
- $h(x_i)$ is the frequency of x_i
- $dis(x_i, Q(x_i))$ is the cost after quantization

Kneron

Model Compression: Quantization

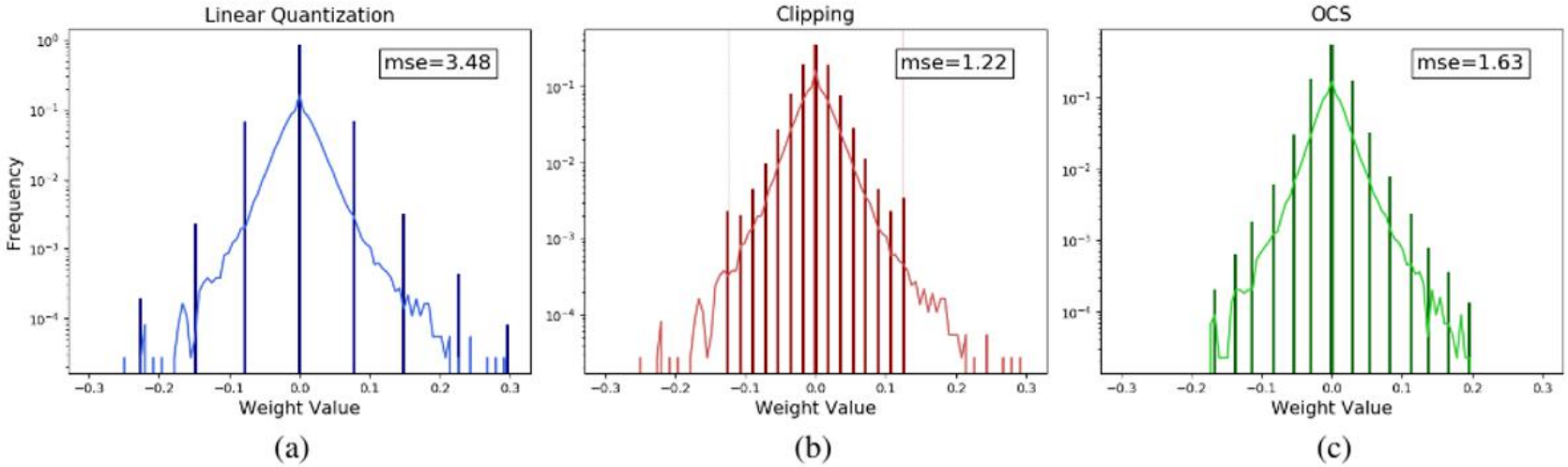
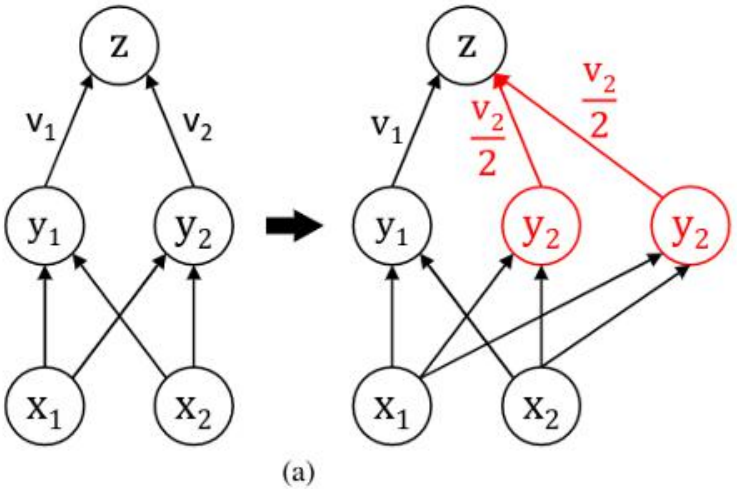


Fig 7 Weight histograms for linear, clipping, and OCS quantization techniques

Zhao, Ritchie, et al. "Improving Neural Network Quantization without Retraining using Outlier Channel Splitting." International Conference on Machine Learning. 2019.

Kneron

Model Compression: Outliner channel split



$$[x_1 \;\; x_2] \times \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} = [y_1 \;\; y_2]$$

$$\left[x_1 \;\; \frac{x_2}{2} \;\; \frac{x_2}{2}\right] \times \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \\ w_3 & w_4 \end{bmatrix} = [y_1 \;\; y_2]$$

$$[x_1 \;\; x_2 \;\; x_2] \times \begin{bmatrix} w_1 & w_2 \\ w_3 & \frac{w_4}{2} \\ 0 & \frac{w_4}{2} \end{bmatrix} = [y_1 \;\; y_2]$$

(b)

Figure 2. OCS network transformation – after duplicating a neuron, we can divide either the neuron's output value or its outgoing weights in half to preserve functional equivalence

Zhao, Ritchie, et al. "Improving Neural Network Quantization without Retraining using Outlier Channel Splitting." International Conference on Machine Learning. 2019.

Kneron

Model Compression:Dynamic Quantization

Dynamic-precision data quantization

For a fixed-point number, its value can be expressed as

$$n = \sum_{i=0}^{bw-1} B_i \cdot 2^{-f_l} \cdot 2^i$$

Adopt a layer-wise quantization by predicting f_l for each layer

Weight quantization: $f_l = argmin_{f_l} \sum |W_{float} - W(bw, f_l)|$,

Feature quantization: $f_l = argmin_{f_l} \sum |x_{float} - x(bw, f_l)|$.

Qiu, Jiantao, et al. "Going deeper with embedded fpga platform for convolutional neural network." *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016.

Model Compression:Knowlege distillation

$$L(s(x), y) = \alpha \sum_i log(s(x)_i)\, y_i + \frac{1-\alpha}{T^2} \sum_i log(s(x,T)_i)\, t(x,T)$$

$$s(x,T)_i = \frac{s(x)_i^{1/T}}{\sum_j s(x)_j^{1/T}}, t(x,T)_i = \frac{s(x)_i^{1/T}}{\sum_j s(x)_j^{1/T}}$$

where T is a temperature that is normally set to 1. Using a higher value for T produces a softer probability distribution over classes.

Tucker, George, et al. "Model Compression Applied to Small-Footprint Keyword Spotting." INTERSPEECH. 2016.

Kneron

Model Compression:SVD Low-rank matrices

Considering an FC layer f

$$f_{out} = W f_{in} + \mathrm{b}$$

$W \in R^{M \times N}$ can be decomposed using Singular Value Decomposition(SVD) as

$$W = U^T \mathrm{SV}$$

$\mathrm{S} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ is a diagonal matrix. We can select the first d singular values in SVD to form $S_d = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_d, 0, \ldots 0)$

$$W \approx W_d = U^T S_d \mathrm{V}$$

$$\mathrm{O(M \times N)} \rightarrow \mathrm{O((M + N) \times d)}$$

Qiu, Jiantao, et al. "Going deeper with embedded fpga platform for convolutional neural network." *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016.

Inspiration

- **NAS may help to find a more efficient network architecture**
- **Low-bit network could serve as a regularizer to prevent overfiting, with better performance**
- **Outliner split methods worth more attention**

Thanks For watching