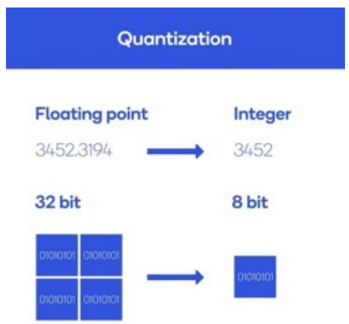
Quantization Through Search: A Novel Scheme to Quantize Convolutional Neural Networks in Finite Weight Space

Qing Lu, Weiwen Jiang, Xiaowei Xu,
Jingtong Hu, and Yiyu Shi

2023.01.18

Quantization of Deep Neural Networks

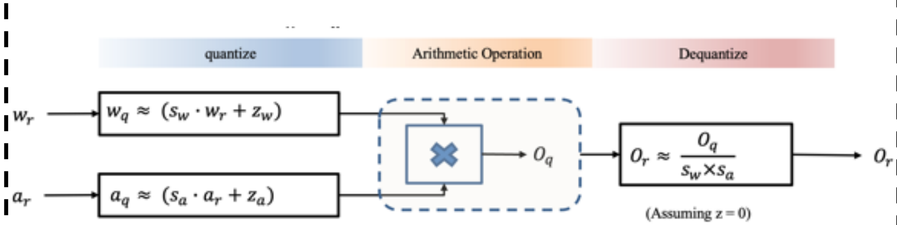
Growth of Interest



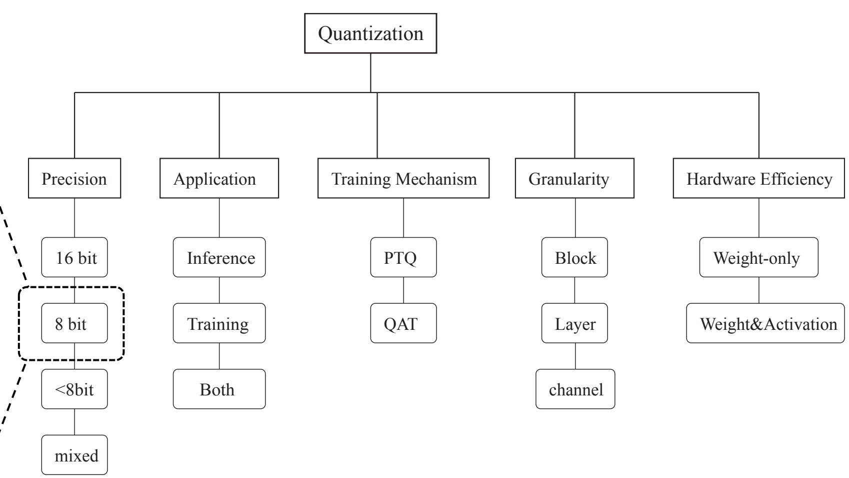
PyTorch → PyTorch

NVIDIA TAO

Taxonomy

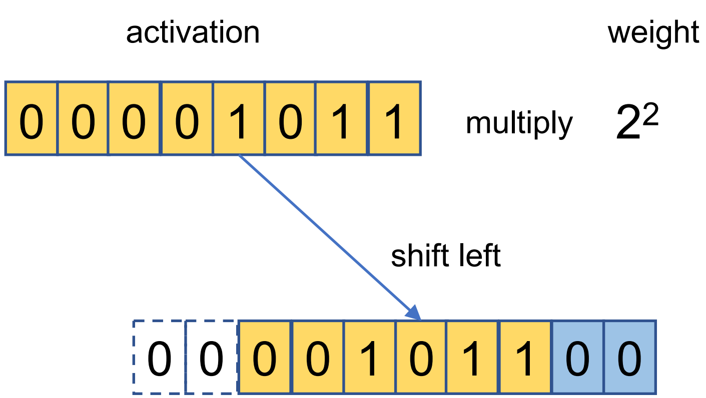


2x ~ 4x CPU inference speedup



Finite Weight Space Quantization

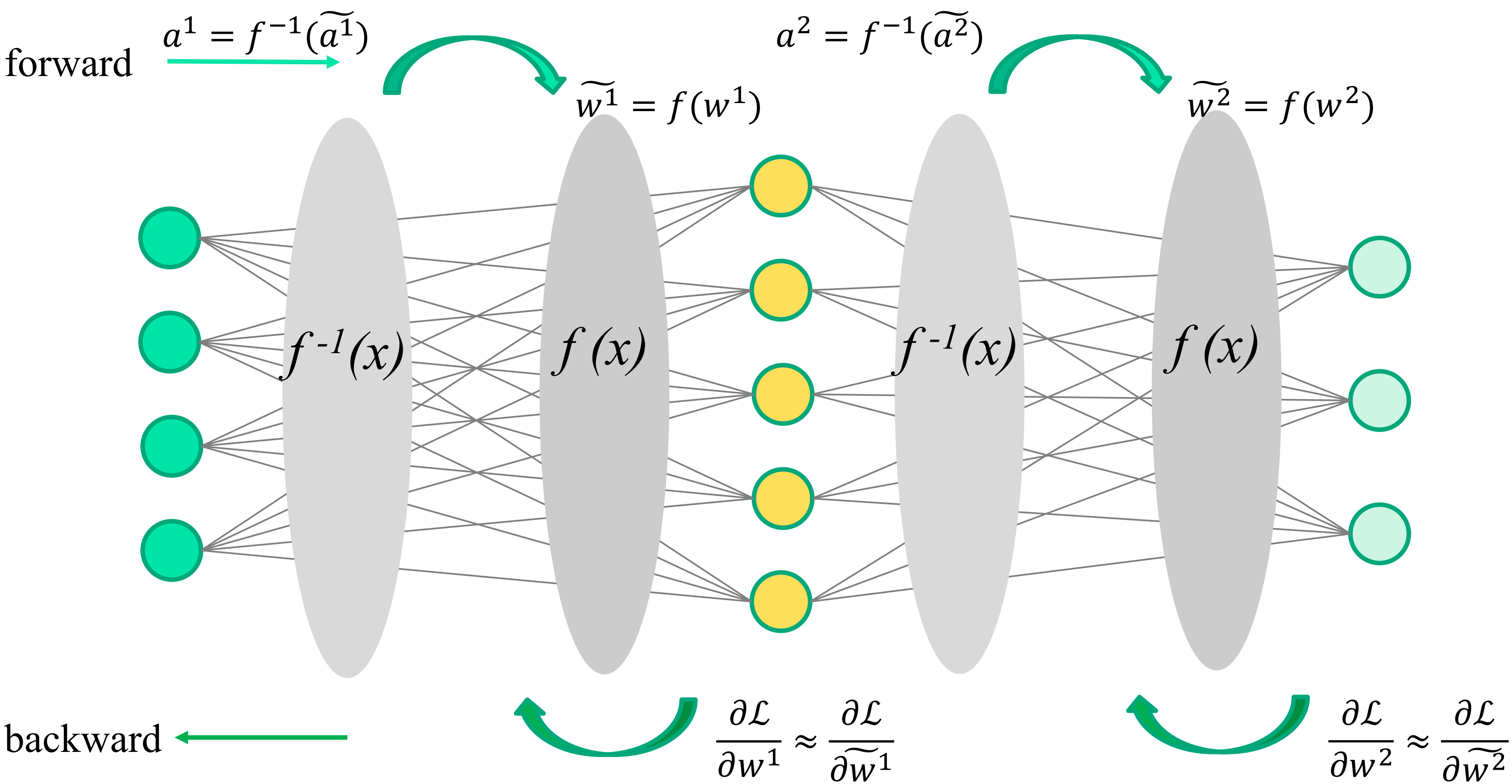
- Constraining weights into a definite set can maximize the execution efficiency
  - Memory is proportional to cardinality of value space;
  - Hardware-friendly value set lowers complexity;
  - Multiplication has equal instances to addition;



Operator	Logic Gates per Unit	Total Operations per Image
addition	~40	~37M
Multiplication	~320	~38M

8-bit quantization. Operations are profiled by assuming image of size 32x32 on ResNet18.

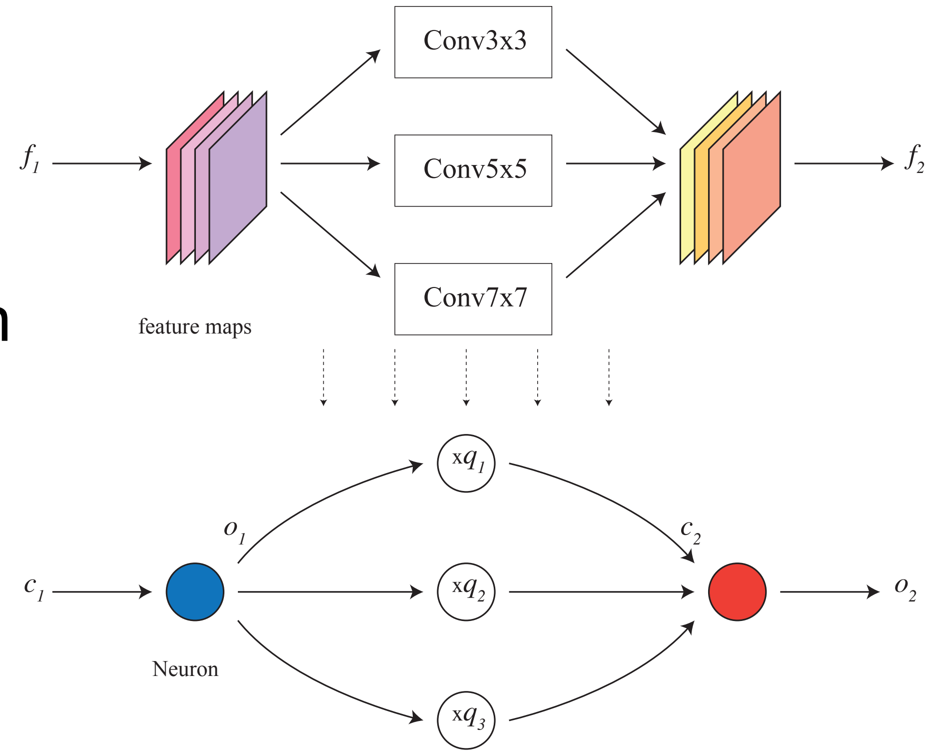
Quantization Through Approximated Function



Quantization Through Search

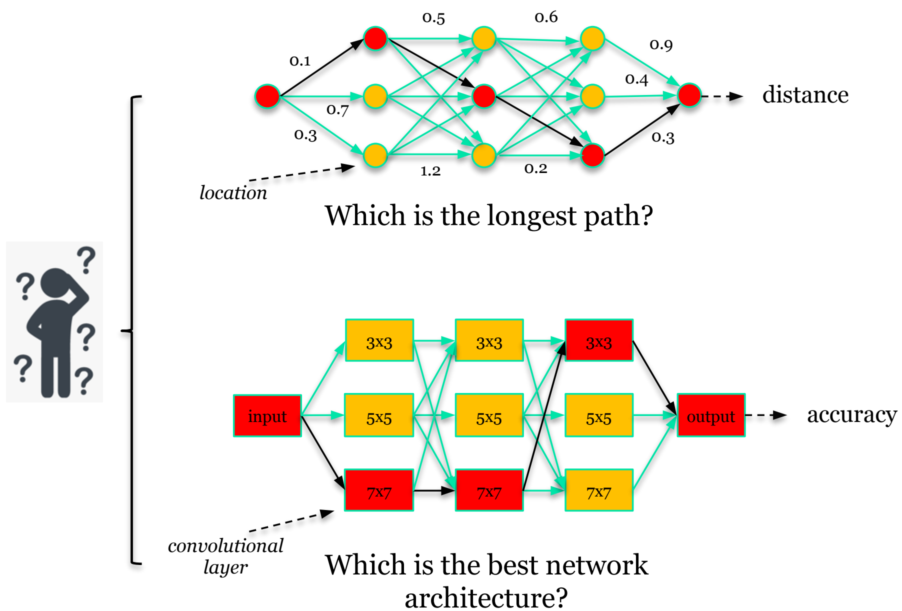
Problem Formulation

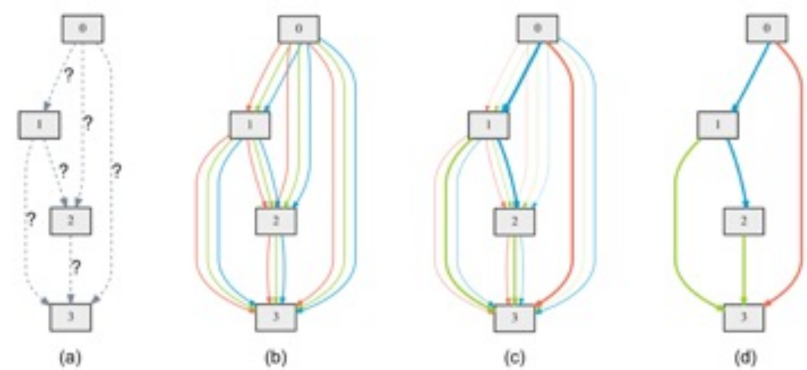
- Discrete search space $\{q^{(m)}\}^n$;
- Splitting path between each connected pair of neurons;
- Selecting weight value is effectively selecting operations as in NAS.



Neural Architecture Search Methods

Considering the huge search space ($\{q^{(m)}\}^n$), differentiable search algorithm is required.





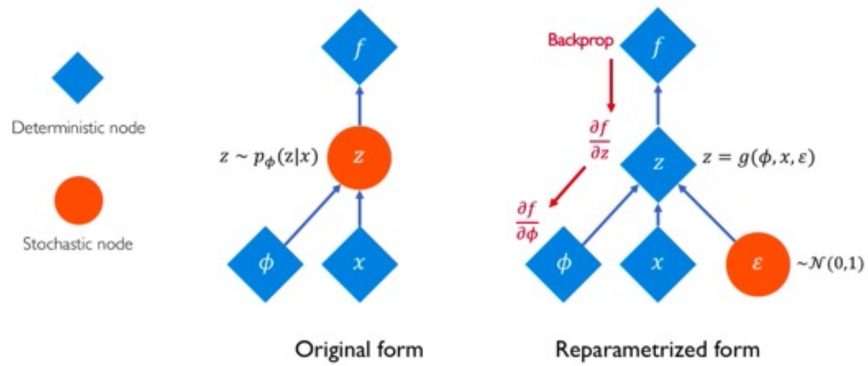
$$\bar{o}^{(i,j)}(x) = \sum_{o\in\mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'\in\mathcal{O}}\exp(\alpha_{o'}^{(i,j)})} o(x)$$

RL-NAS: Markov decision process

Differentiable NAS

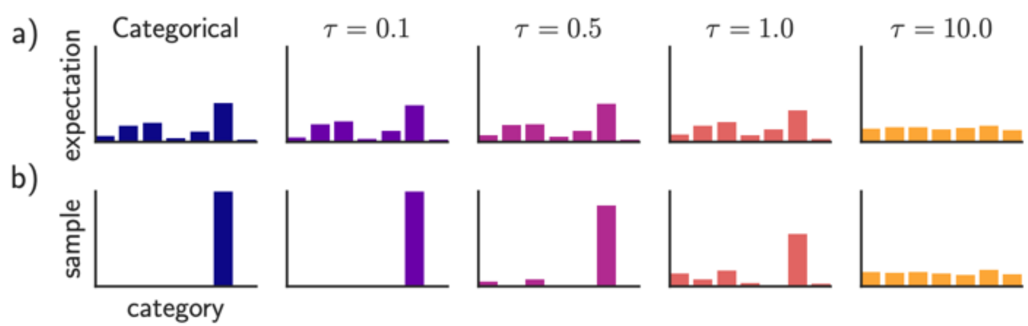
Indifferentiable Node in Networks

Gumbel Softmax



Reparameterization

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)} \qquad \text{for } i = 1, ..., k.$$



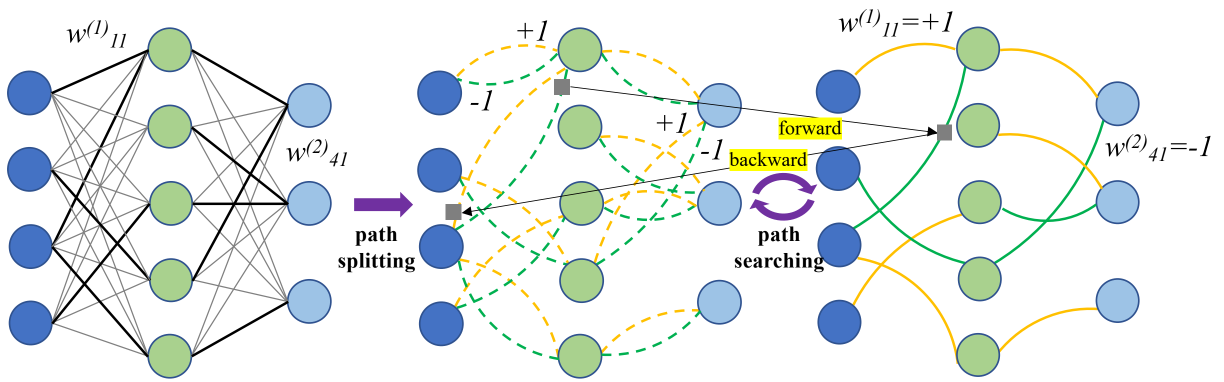
Problem Formulation

For each neuron pair <*i, j*>, we create *m* paths between them and:

- Associating probability variable;
- Define the forward selection mechanism;

$$f(q^1, q^2, \cdots q^m | \alpha_{i,j}^1, \alpha_{i,j}^2, \cdots \alpha_{i,j}^m) = q^k$$

- Derive the backward gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_{i,j}}$



Method: RDHS

Reparameterization with deterministic hard-sampling

General form of sampling function with reparameterization trick

randomness

soft sample

$$y_{i,j}^k = \frac{exp((log(\alpha_{i,j}^k) + \mu n_{i,j}^k)/\tau)}{\sum_k exp((log(\alpha_{i,j}^k) + \mu n_{i,j}^k)/\tau)}$$

temperature

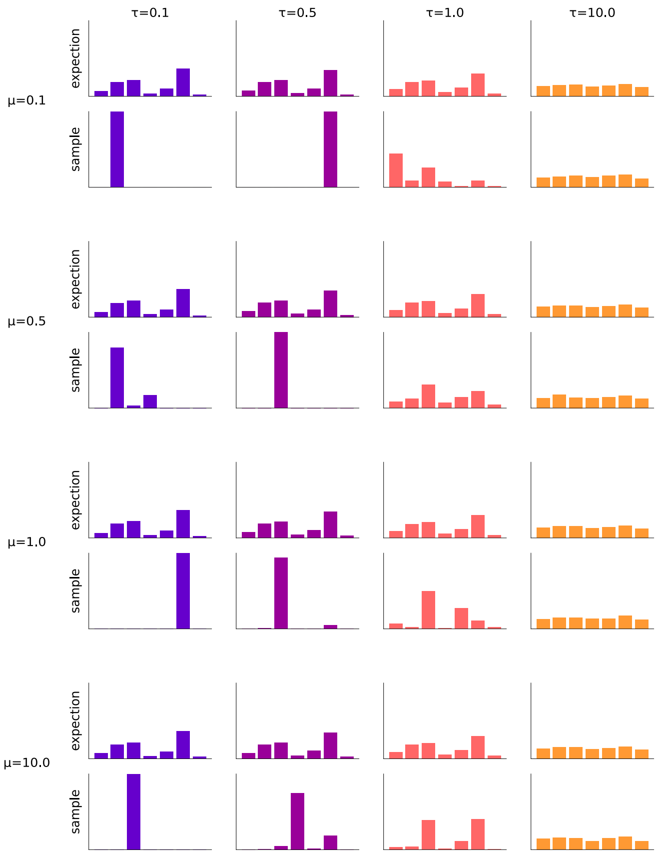
$$z_{i,j} = \text{one\_hot}\left(\arg\max_k \left(y_{i,j}^k\right)\right)$$

hard sample

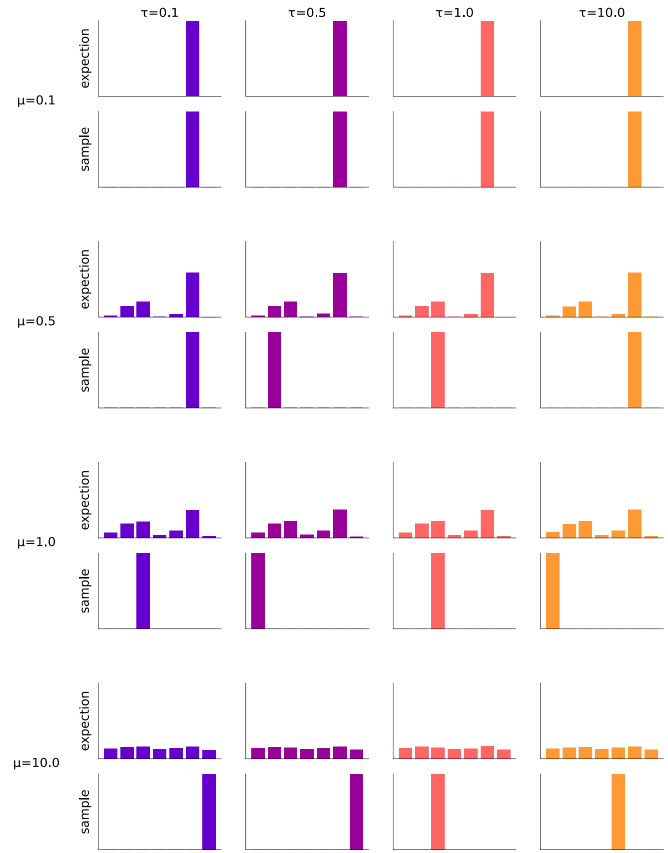
$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{y}_{i,j}} \approx \frac{\partial \mathcal{L}}{\partial z_{i,j}}$$

ST estimator

Hard-Sampling vs Soft-Sampling

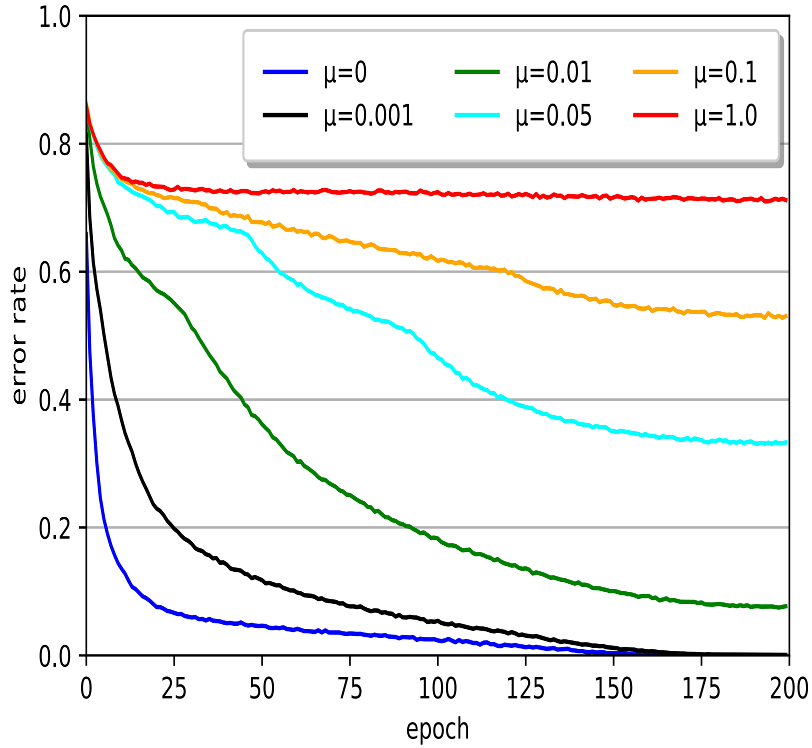


Gumbel-Softmax



Gumbel-Max

Error Rate Floor vs Randomness



Training VGG-small with binary weights ({-1, 1}).

Table 3: Exploration of the relationship between accuracy and randomness in sampling. For all the test networks and bit-widths, the best accuracy occurs when deterministic sampling ($\mu = 0$) is adopted.

Network	#Bits	μ	Acc. (%)
VGG-small	1	0	93.6
		0.001	93.4
		0.01	85.6
ResNet20	1	0	90.9
		0.001	90.9
		0.01	90.7
		0.1	64.7
ResNet18	1	0	64.6/85.4
		0.001	64.1/84.1
		0.01	45.1/66.3
	2	0	67.4/87.5
		0.001	55.1/69.7
	3	0	68.1/88.2
		0.001	48.8/64.4

Method: SGT

Stochastic Gradient Transfer

- Hard-sampling: only one path needs to be involved in the computational graph.

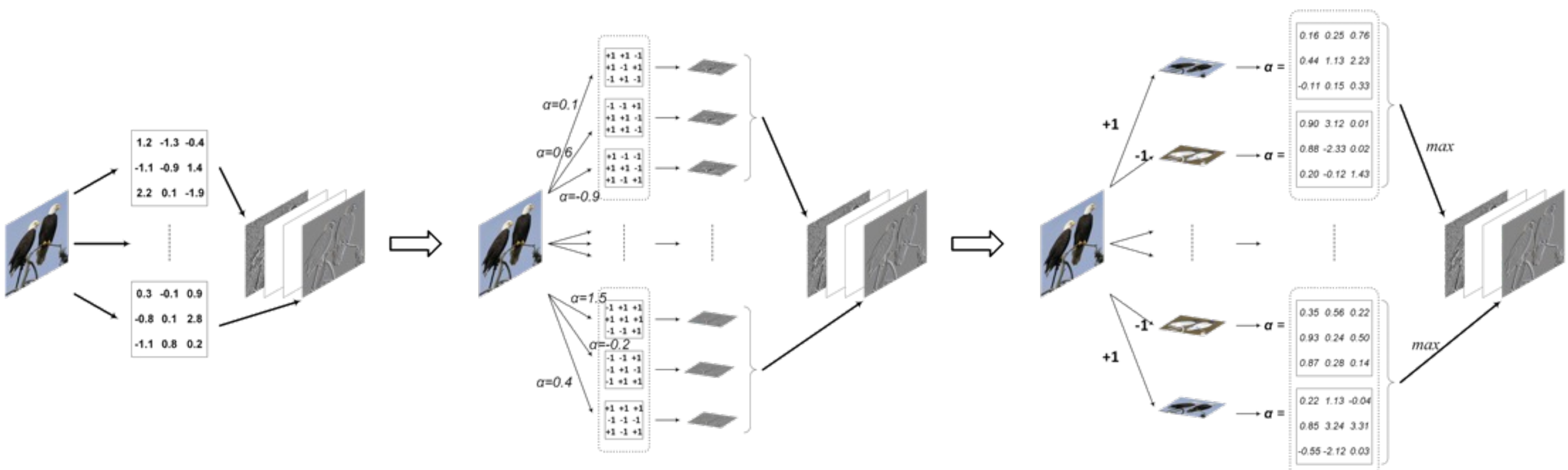
$$k = \arg\max_{r} \alpha_{i,j}^{r}.$$

- Heuristic: when the gradient of q^k is positive, it should decrease, other candidates should gain a higher odd, their associated parameters should increase; vice versa.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_{i,j}} = \frac{\partial \mathcal{L}}{\partial q_{i,j}^{k}} \cdot sign(\boldsymbol{q}_{i,j} - q_{i,j}^{k}) \qquad \text{where} \qquad \alpha_{i,j}^{k} = \max\{\alpha_{i,j}^{r}\}$$

Implementation Optimization for ConvNets

Neuron-to-neuron splitting requires exponentially increased parameters for updating, therefore we assign the associated parameters to the convolutional kernels scaled by each of *m* masks.



Results on CNN (CIFAR10, ImageNet)

Table 1: Results using different methods on CIFAR-10: accuracy (%), accuracy loss (%) against unquantized network (FP32), and whether multiplication is required are reported.

Network	Method	Value Space	Acc. (%)	ΔAcc. (%)	Req. Mult
VGG-small	BC [2]	$\{\pm 1\}$	91.7	2.1	N
	BWN [14]	$\mathbb{R}$	90.1	3.7	Y
	LAB [7]	$\{\pm 1\}$	89.5	4.3	N
	LQ-Net [17]	$\mathbb{R}$	93.5	0.3	Y
	Ours	$\{\pm 1\}$	**93.6**	**0.2**	N
ResNet-20	DoReFa [19]	$\mathbb{R}$	90.0	2.1	N
	LQ-Net [17]	$\mathbb{R}$	90.1	2.0	Y
	Ours	$\{\pm 1\}$	**90.9**	**0.9**	N
MobileNet	*ours*	$\{\pm 1\}$	93.78	0.3	N

Table 2: Results of ResNet18 quantized by different methods on ImageNet: Top-1/Top-5 accuracy (%) and accuracy loss (%) (shown in the parentheses) from the unquantized network.

#Bits	Method	Value Space	Top-1	Top-5	Req. Mult
1	BWN [14]	$\mathbb{R}$	60.8(8.5)	83.0(6.2)	Y
	ABC-Net [12]	$\mathbb{R}$	62.8(6.5)	84.4(4.8)	Y
	DSQ [5]	$\{\pm 1\}$	63.7(6.2)	-	N
	Ours	$\{\pm 1\}$	64.6(**4.5**)	85.4(**3.5**)	N
2	TWN [10]	$\mathbb{R}$	61.8(3.6)	84.2(2.6)	Y
	INQ [18]	$\{\pm 1/2^n\}$	66.0(2.3)	87.1(1.6)	N
	TTQ [20]	$\mathbb{R}$	66.6(3.0)	87.2(2.0)	Y
	ADMM [11]	$\mathbb{R}$	67.0(2.1)	87.5(1.5)	Y
	ABC-Net [12]	$\mathbb{R}$	63.7(5.6)	85.2(4.0)	Y
	LQ-Net [17]	$\mathbb{R}$	68.0(2.3)	88.0(1.5)	Y
	Ours	$\{\pm 1, \pm \frac{1}{2}\}$	67.4(**1.7**)	87.5(**1.4**)	N
3	INQ [18]	$\{\pm 1/2^n\}$	68.1(0.2)	88.4(0.3)	N
	ABC-Net [12]	$\mathbb{R}$	66.2(3.1)	86.7(2.5)	Y
	LQ-Net [17]	$\mathbb{R}$	69.3(1.0)	88.8(0.7)	Y
	Ours	$\{\pm 1, \pm \frac{1}{2}, \pm \frac{1}{4}, \pm \frac{1}{8}\}$	68.1(**1.0**)	88.2(**0.9**)	N

Performance of RDHS vs SGT

SGT

- Faster
- Less memory
- Higher variance

RDHS

- Slower
- Higher memory
- Lower variance

Table 4: Full comparison between RDHS and SGT in training ResNet18 on ImageNet. Training time and memory footprint are reported in relative to training in full precision.

#Bits	Method	Avg. Acc. (%)	Best Acc. (%)	Time	Memory
1	RDHS	64.3/83.9	64.6/85.4	1.47	1.34
	SGT	64.1/81.8	64.5/85.4	1.33	1.10
2	RDHS	67.5/87.0	67.4/87.5	1.66	1.48
	SGT	66.7/86.2	67.4/87.5	1.48	1.24
3	RDHS	67.9/88.1	68.1/88.2	1.96	1.73
	SGT	66.3/85.6	68.9/87.9	1.74	1.45

Conclusion

- Proposed to quantize deep neural networks from a "search" perspective.
- Identified the necessity of using deterministic hard-sampling in differentiable searching algorithm for quantization.
- Implemented and evaluated two methods for ConvNets with optimization tricks, achieving best trade-off between accuracy and execution efficiency.
- Our code is available at https://github.com/qinglu0330/DNN-weight-search

Thank You

Presenter: Qing Lu
Email: qlu2@nd.edu

ASIA SOUTH PACIFIC DAC DESIGN AUTOMATION CONFERENCE