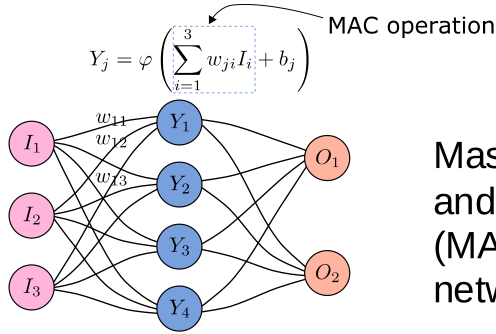
Efficient and Robust Hardware for Neural Networks

Grace Li Zhang
Assistant Professor on Hardware for AI
TU Darmstadt, Germany
https://www.etit.tu-darmstadt.de/hwai/
grace.zhang@tu-darmstadt.de
+49 6151 16-20256

Neural Networks

Synapse Network

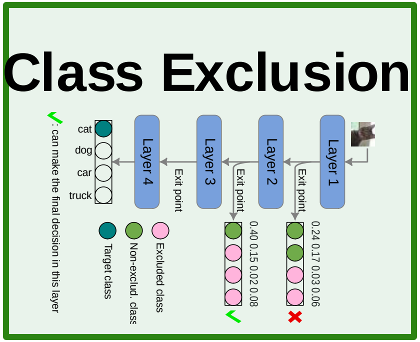


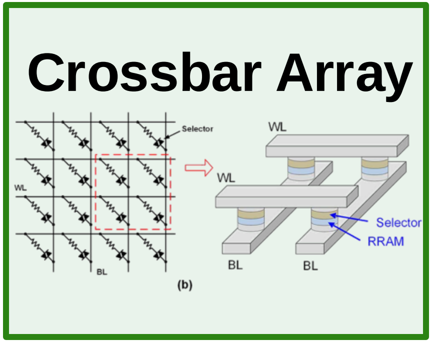
A Simple Neural Network

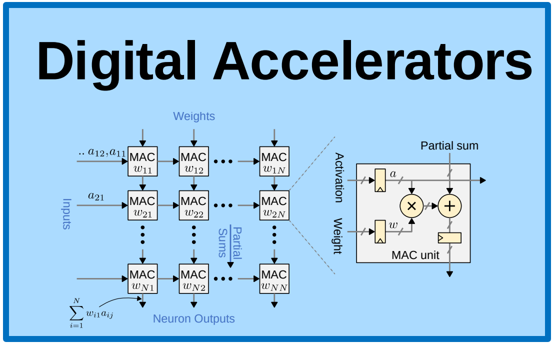
Massive multiplication and addition operations (MAC) in neural networks

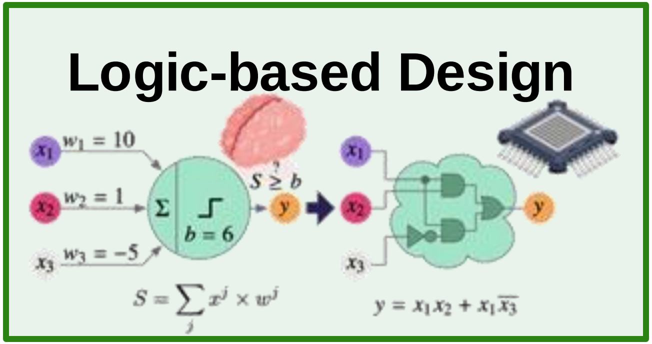
- GPT-3 in ChatGPT: 96 layers with 175 billion weights → trillions of MAC operations
- Training GPT-3 consumed 1287 MWh energy → 552 tons CO_2 equivalent emission → comparable to the electricity consumption of 120 years for an average U.S. household

Energy Reduction Techniques









Outlines

Class-Aware Pruning for Efficient Neural Networks

Early-Exit with Class Exclusion for Efficient Inference of Neural Networks

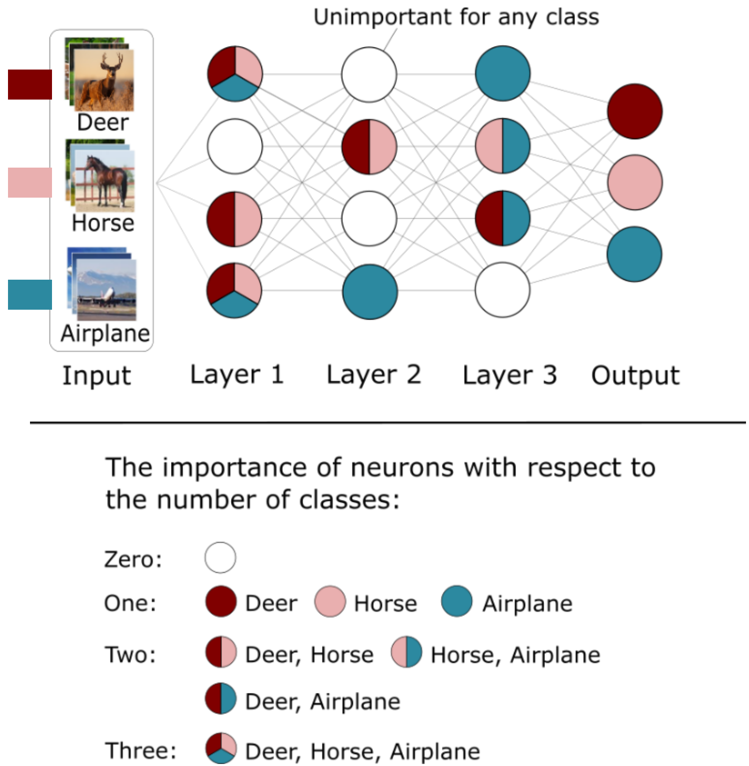
Power Reduction for Digital Accelerators of Neural Networks

Logic-Based Design of Neural Networks

Robustness Enhancement of RRAM-based Neural Network Acceleration

A New Pruning Perspective: Class-Based Criteria

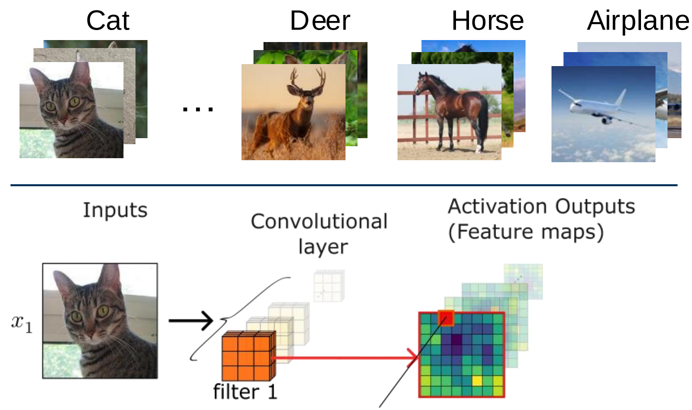
- Different neurons contribute to different number of classes
- Neurons contribute to a few number of classes can be pruned
- Retraining to compensate accuracy loss



Filter Importance Evaluation

- Importance evaluation of a filter with respect to one class → adding the importance for all classes

- Importance of a filter for one class: sensitivities of the resulting activations to cost function



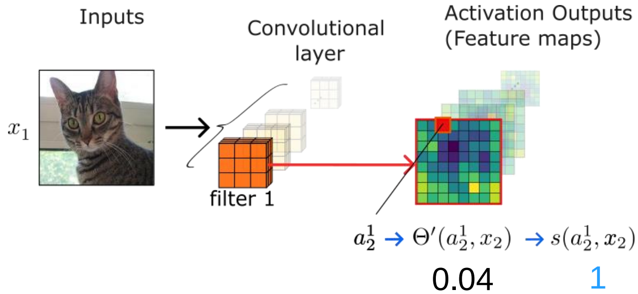
activation output a_1^2 set to zero

$$\Theta(a_i^f, x_j) = \left|\mathcal{L}(x_j) - \mathcal{L}\left(x_j; a_i^f \leftarrow 0\right)\right|$$

Filter Importance Evaluation

- First-order Taylor expansion approximation:

$$\Theta'(a_i^f, x_j) = \left| a_i^f \frac{\partial \mathcal{L}(x_j)}{\partial a_i^f} \right|$$



- Score of one activation output with ***one image:***

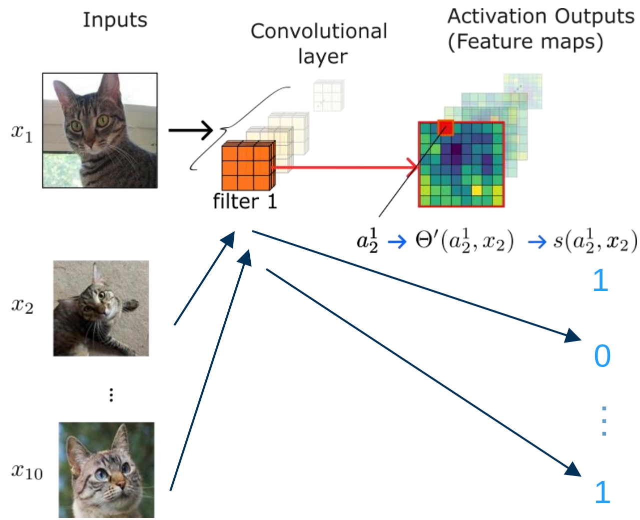
$$s(a_i^f, x_j) = \begin{cases} 1, & \Theta'(a_i^f, x_j) > \tau \\ 0, & \Theta'(a_i^f, x_j) \leq \tau \end{cases}$$

10^{-50}

Filter Importance Evaluation

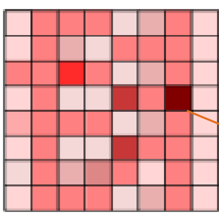
- Score of one activation output for ***one class***:

$$s_{ave}(a_2^1) = \sum_{j=1}^{10} \frac{1}{10} s(a_2^1, x_j)$$



$$\frac{1}{10}(1 + 0 + \cdots + 1) = 0.6$$

- Importance score of **one filter** for one class:
  Maximum

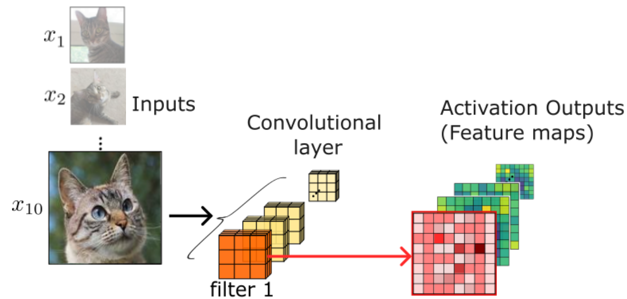


0.9

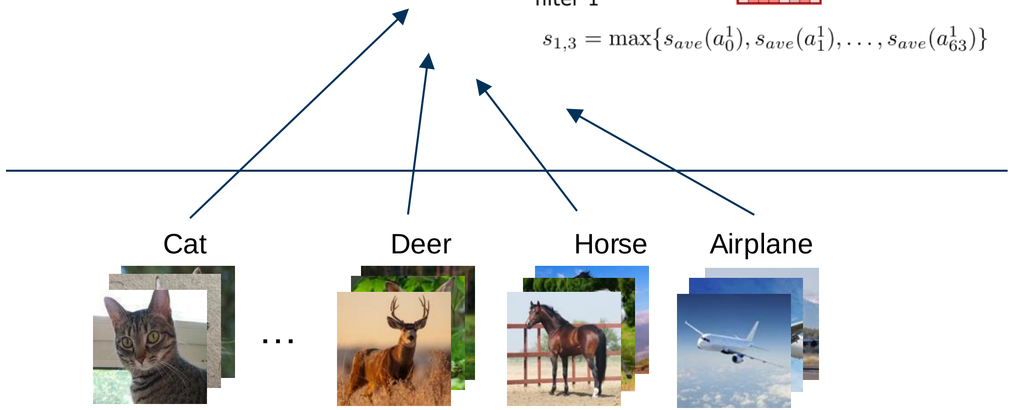
Filter Importance Evaluation

- Score of one activation output for ***all classes***:

  Sum



$$s_{1,3} = \max\{s_{ave}(a_0^1), s_{ave}(a_1^1), \ldots, s_{ave}(a_{63}^1)\}$$



Total importance score for ***filter 1*** is (0.9 + ⋯ + 0.4 + 0.5 + 0.8) = 6.5

Regularization In Training

- Polarized scores facilitate pruning
- Cost function:

$$\mathcal{L} = \mathcal{L}_{\mathcal{CE}} + \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_{orth}$$

- Convolutional-orthogonality regularization: Penalize similarity between filters

$$\mathcal{L}_{orth} = \sum_{l=1}^{H} \left\| \mathcal{K}\mathcal{K}^T - I \right\|_2$$

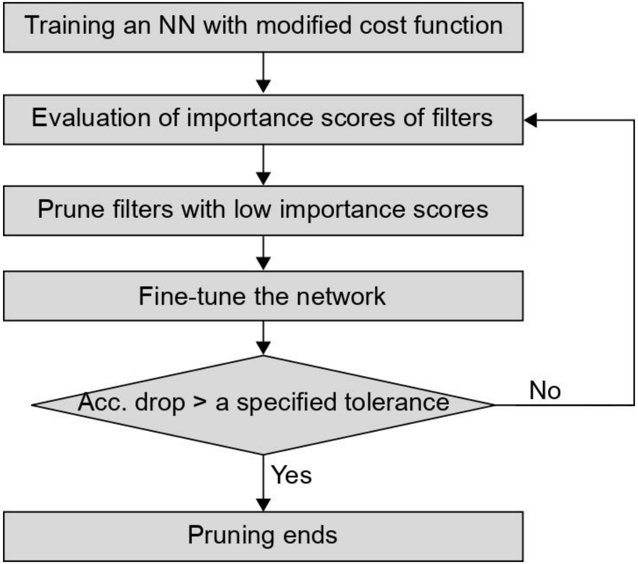
- L1 regularization: Penalty the number of weights, sparse the weight matrix

$$\mathcal{L}_1 = \sum_{l=1}^{H} \|\mathbf{W}_l\|_1$$

[Convolutional-orthogonality]: J. Wang, Y. Chen, R. Chakraborty, and S. X. Yu, "Orthogonal convolutional neural networks," in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020

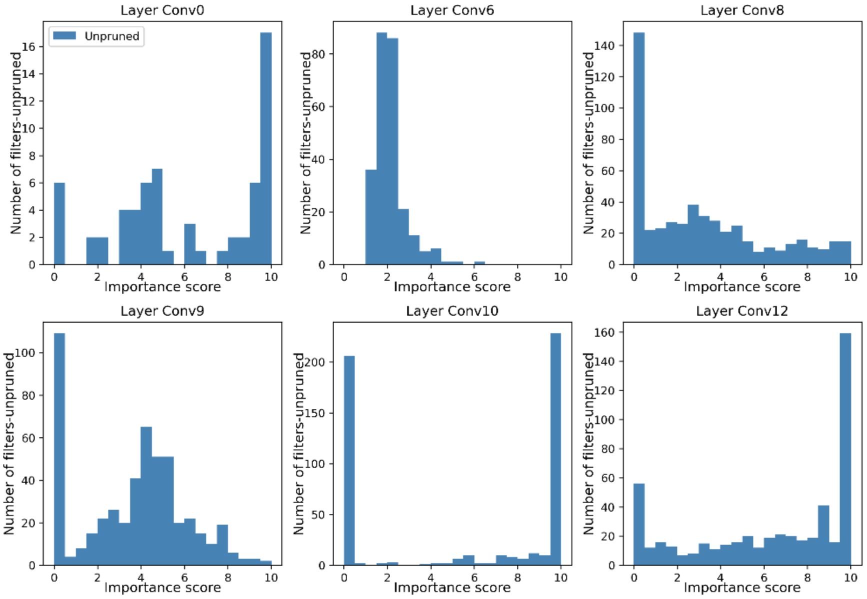
Pruning Workflow

- Iterative pruning:
  - Evaluation
  - Pruning
  - Fine-tune
  - Whether to loop



Importance Score Distribution

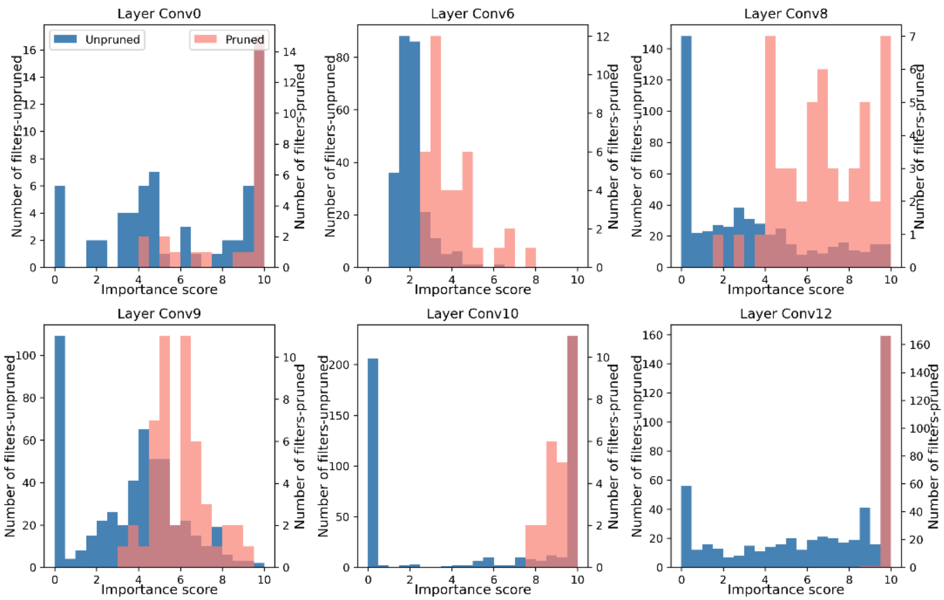
- Before pruning: lower importance scores



Filter importance scores distribution before pruning: VGG16-CIFAR10

Importance Score Distribution

- After pruning: shift right, higher average score



Filter importance scores distribution before and after pruning: VGG16-CIFAR10

Experimental Results

NN-Dataset	Accuracy comparison		Pruning performance	
	Original	Pruned	Pruning ratio	FLOPs reduction
VGG16-CIFAR10	93.90%	92.99%	95.6%	77.1%
VGG19-CIFAR100	73.49%	72.56%	85.4%	75.2%
ResNet56-CIFAR10	93.71%	92.89%	77.9%	62.3%
ResNet56-CIFAR100	72.36%	71.49%	50.0%	43.8%

Mengnan Jiang, Jingcun Wang, Amro Eldebiky, Xunzhao Yin, Cheng Zhuo, Ing-Chao Lin and Grace Li Zhang, "Class-Aware Pruning for Efficient Neural Networks", *Design Automation and Test in Europe,* 2024, **Nominated as Best Paper Award**

Outlines

Class-Aware Pruning for Efficient Neural Networks

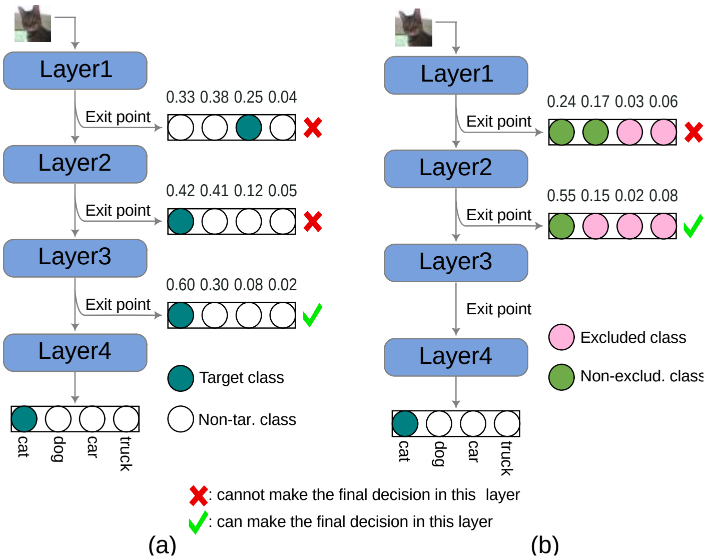
Early-Exit with Class Exclusion for Efficient Inference of Neural Networks

Power Reduction for Digital Accelerators of Neural Networks

Logic-Based Design of Neural Networks

Robustness Enhancement of RRAM-based Neural Network Acceleration

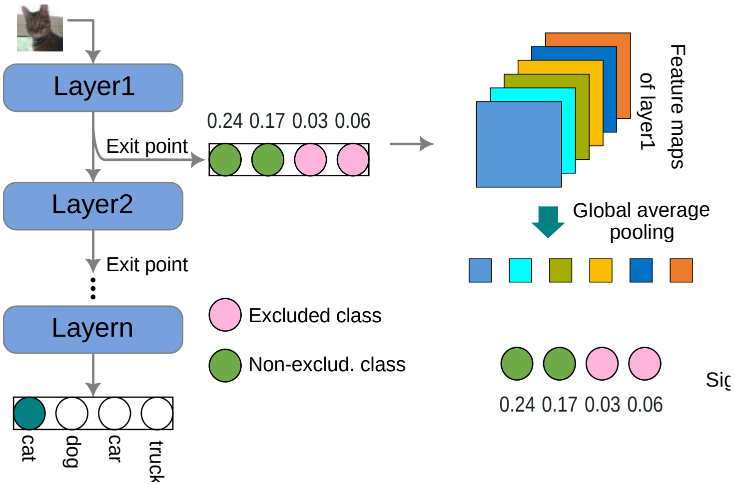
Early-Exit with Class Exclusion



- Previous early-exit discard intermediate results if an intermediate cannot decide the correct class
- Proposed: Learned features in early layers are used to exclude as many irrelevant classes as possible

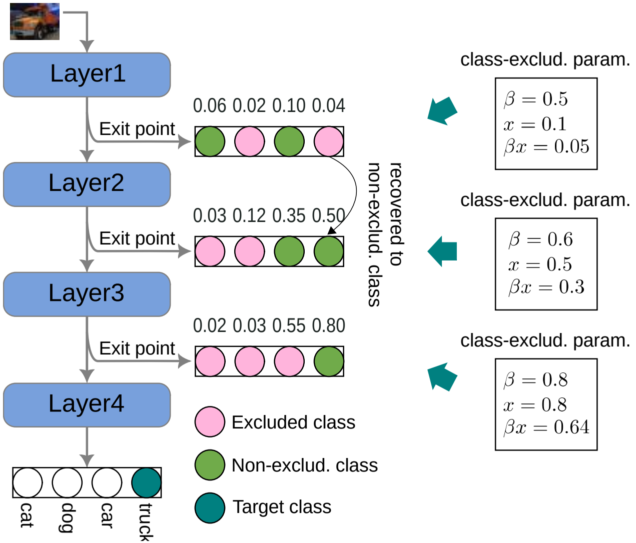
Jingcun Wang, Bing Li and Grace Li Zhang, Early-Exit with Class Exclusion for Efficient Inference of Neural Networks, International *Conference* on Artificial Intelligence Circuits and Systems (AICAS), 2024

Class-Exclusion Neural Network Construction



- Individual class-exclusion network to each class
- A fully-connected network + Sigmoid function for class exclusion

Class-Exclusion Strategy for Dynamic Inference

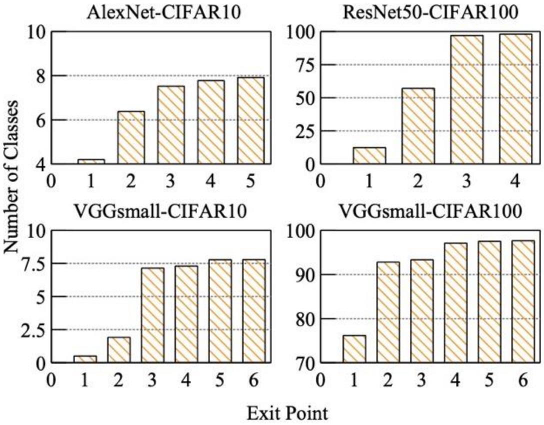


- Relative magnitude of probabilities generated by class-exclusion networks used to exclude classes
- A search algorithm is used to determine a class-exclusion coefficient, denoted as β

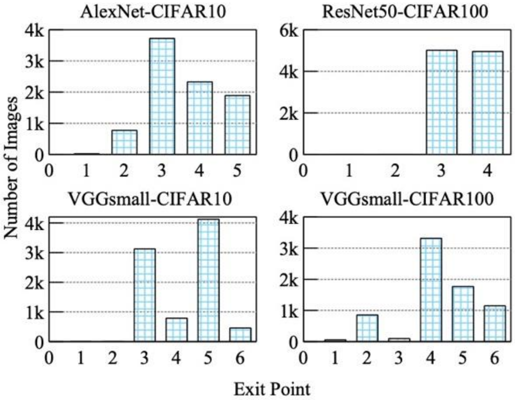
Experimental Results

NN	Original	Proposed	FLOPs(G) Ori.	FLOPs(G) Pro.	Red.
AlexNet-CIFAR10	90.54%	89.34%	1.4386	1.0375	27.88%
VGGs-CIFAR10	93.89%	91.93%	1.5460	1.0668	31%
VGGs-CIFAR100	72.19%	71.11%	1.5463	1.1535	25.4%
ResNet50-CIFAR100	76.46%	74.39%	2.6008	1.7411	33.06%

Experimental Results



The average number of excluded classes in each exit point of intermediate layers in neural networks.



The number of input images that can be classified in the intermediate exit point of neural networks.

Outlines

Class-Aware Pruning for Efficient Neural Networks

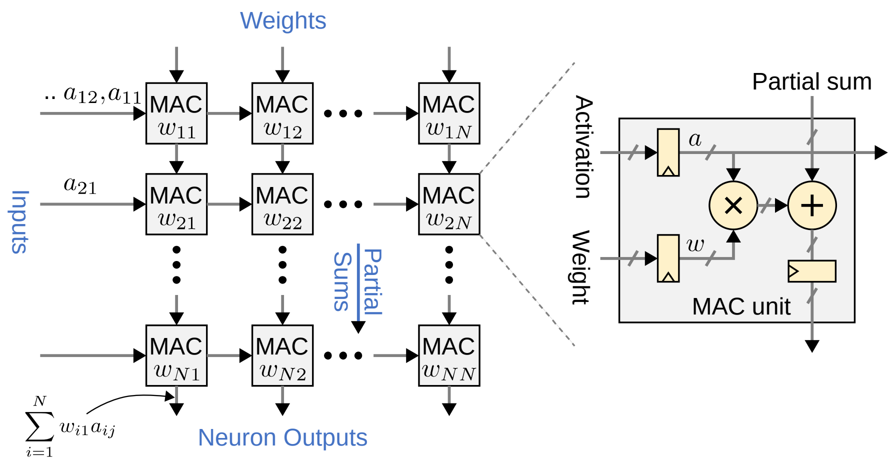
Early-Exit with Class Exclusion for Efficient Inference of Neural Networks

Power Reduction for Digital Accelerators of Neural Networks

Logic-Based Design of Neural Networks

Robustness Enhancement of RRAM-based Neural Network Acceleration

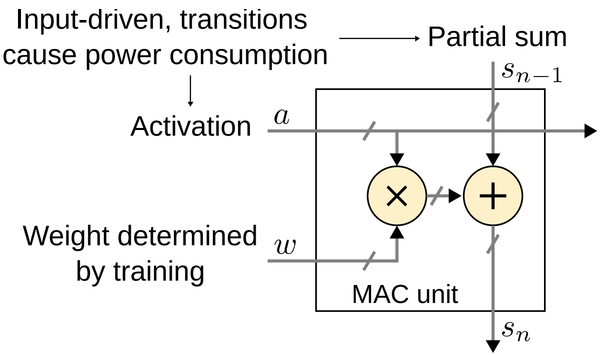
Digital Hardware Acceleration of Neural Networks

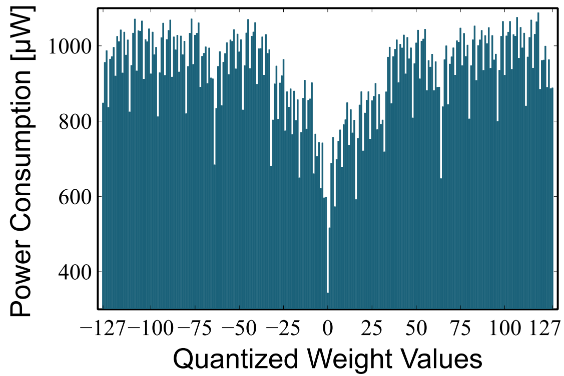


- Weights and inputs pipelined through a systolic array for a better performance
- MAC operations implemented in digital circuits → need to balance PPA

N. P. Jouppi et al., “In-datacenter performance analysis of a tensor processing unit,” Int. Symp. Comput. Arch. (ISCA), 2017.

Power-driven Weight Selection

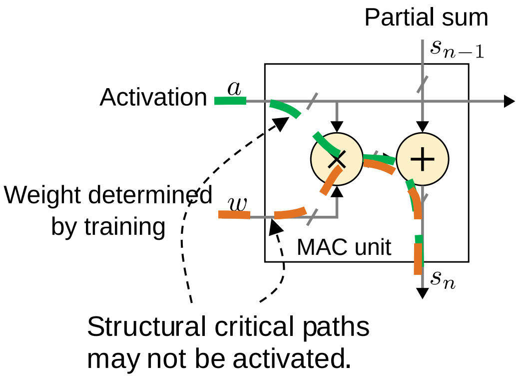


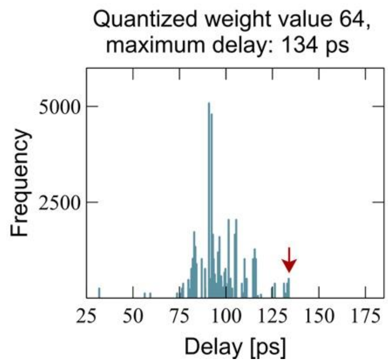


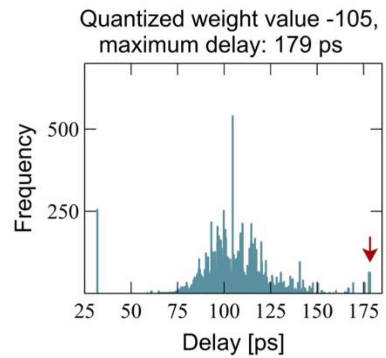
- Power consumption of a MAC unit determined by input transitions
- Select weight values according to average power consumption

R. Petri, L. Zhang, Y. Chen, U. Schlichtmann, B. Li, "PowerPruning: Selecting Weights and Activations for Power-Efficient Neural Network Acceleration", *ACM/IEEE Des. Autom. Conf. (DAC)*, 2023

Delay-driven Weight Selection







- Select weight values and activations according to the delays of circuit paths triggered in the MAC units
- Voltage scaling V↓ to reduce power consumption P~V^2

R. Petri, L. Zhang, Y. Chen, U. Schlichtmann, B. Li, “PowerPruning: Selecting Weights and Activations for Power-Efficient Neural Network Acceleration”, *ACM/IEEE Des. Autom. Conf. (DAC)*, 2023

Power Reduction Results

NN	Original	PowerPruning	Power reduction	# selected weights	#selected actiations
LeNet-5-CIFAR-10	80.6%	78.5%	78.3%	35	210
ResNet-20-CIFAR-10	91.9%	89.6%	56.6%	35	210
ResNet-50-CIFAR-100	79.9%	78.5%	77.6%	41	223
EfficientNet-B0-Lite-ImageNet	73.8%	72.9%	20.8%	50	236

- Powerpruning can reduce power significantly with a slight accuracy loss.

R. Petri, L. Zhang, Y. Chen, U. Schlichtmann, B. Li, "PowerPruning: Selecting Weights and Activations for Power-Efficient Neural Network Acceleration", *ACM/IEEE Des. Autom. Conf. (DAC),* 2023

Outlines

Class-Aware Pruning for Efficient Neural Networks

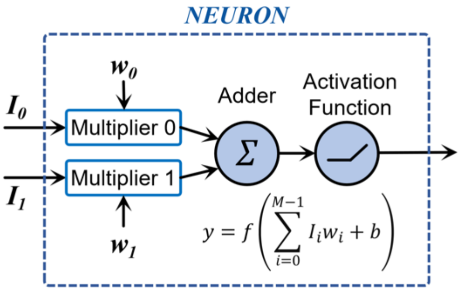
Early-Exit with Class Exclusion for Efficient Inference of Neural Networks

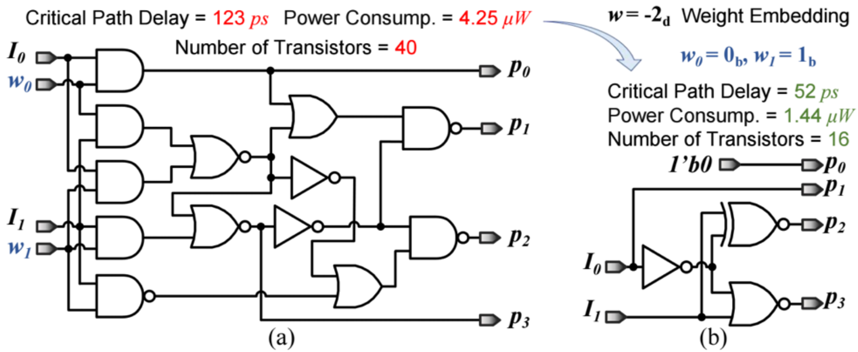
Power Reduction for Digital Accelerators of Neural Networks

Logic-Based Design of Neural Networks

Robustness Enhancement of RRAM-based Neural Network Acceleration

Logic Implementation: Multiplier

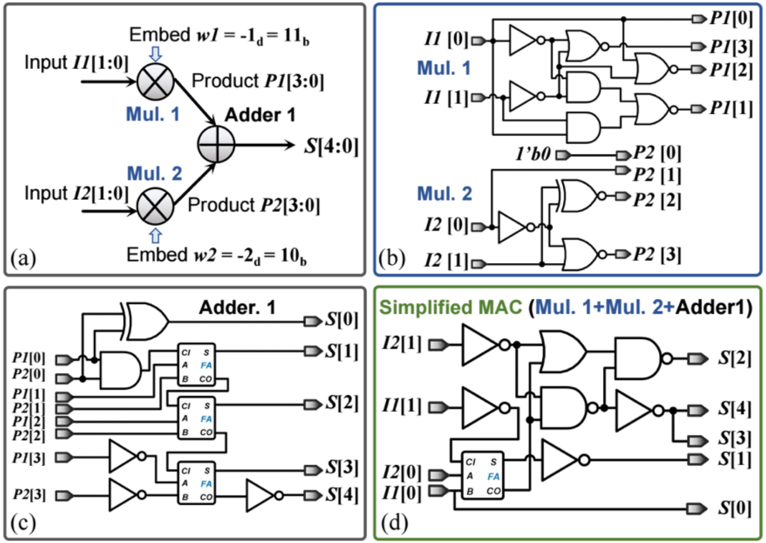




Logic circuit of a 2-bit signed multiplier. (a) The original circuit; (b) The logic circuit simplified with a fixed quantization weight (decimal: -2, binary: 10).

- Use the fixed weights after training to simplify the logic of the multipliers at neurons.
- Example of a 2-bit signed multiplier after embedding a weight:
  - Delay: ↓57.72% Power consumption: ↓66.12% Number of transistors: ↓60%

Logic Implementation: Adder and MAC

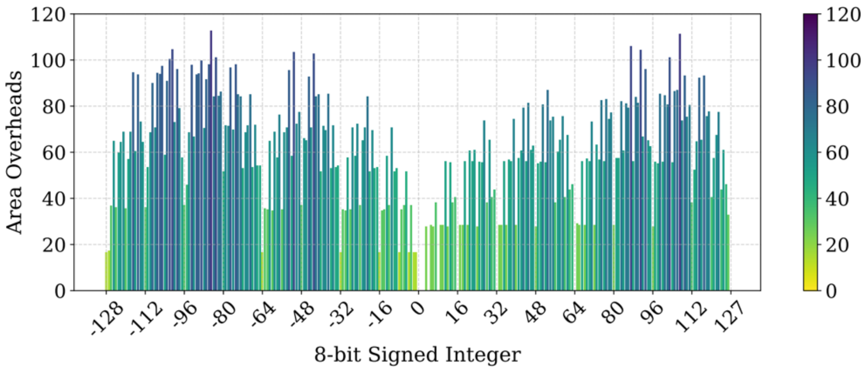


(a) MAC operations at a neuron; (b) 2-bit signed multipliers simplified with the fixed quantized weights; (c) 4-bit signed adder circuit before logic simplification, FA is a 1-bit full adder; (d) Circuit of the simplified MAC.

MAC after weight embedding:

- Delay: ↓70.07%
- Power consumption: ↓60.94%
- Transistors: ↓65%

Hardware-Aware Training



Area of multipliers simplified with 8-bit quantized weights.

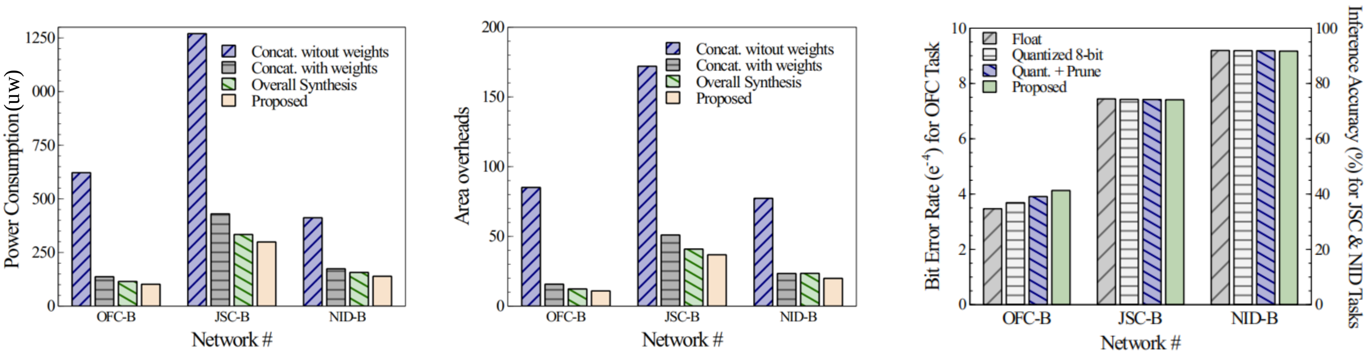
Weight Selection

① **Rank the weight values** according to the area of the simplified multipliers

② **Select the top n weights** that lead to the smallest multiplier area

③ If the validation accuracy is much lower, **more weight values are selected**

Power and Area Results



(a) Power consumption; (b) Area overhead; (c) Accuracy comparison.
Neurons in layers: OFC-B 21, 50, 25; JSC-B 64, 32, 32, 32; NID-B 593, 20, 20

- Concatenate MAC units without weight embedding
- Embedding weights into multipliers
- Simplification between different logics
- Hardware-aware training

- Power Consumption:
  - 83.60% for the OFC task
  - 76.46% for the JSC task
  - 66.26% for the NID task

Optical Fiber Communication (OFC); Jet Substructure Classification (JSC); Network Intrusion Detection (NID)

K. Xu, L. Zhang, U. Schlichtmann, B. Li, “Logic Design of Neural Networks for High-Throughput and Low-Power Applications”, *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2024

Outlines

Class-Aware Pruning for Efficient Neural Networks

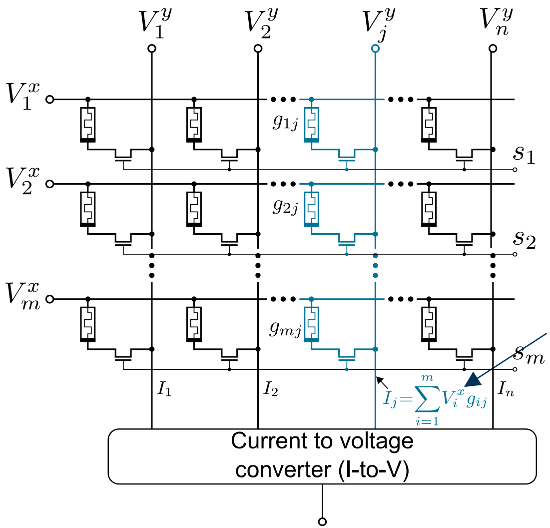
Early-Exit with Class Exclusion for Efficient Inference of Neural Networks

Power Reduction for Digital Accelerators of Neural Networks

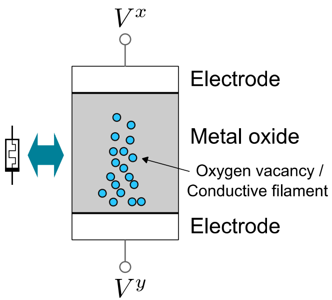
Logic-Based Design of Neural Networks

Robustness Enhancement of RRAM-based Neural Network Acceleration

RRAM-based Neural Network Acceleration



Accumulated current as result of multiplication



$i = V^x g$, assume $V^y = 0$

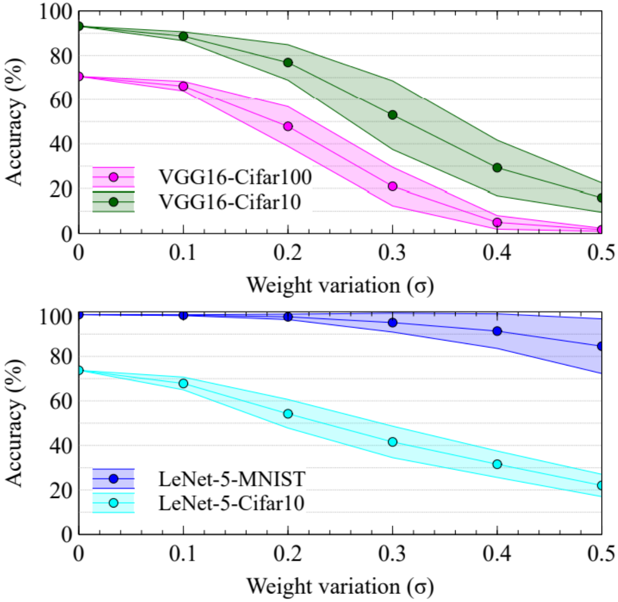
- Weights and inputs are represented with RRAM conductances and voltages, respectively.
- Vector-matrix multiplication is implemented based on Ohm's law and Kirchhoff's law.

Accuracy Degradation due to Process Variations

- Process variations → weight deviations → erroneous feature maps → error amplification across layer → accuracy loss

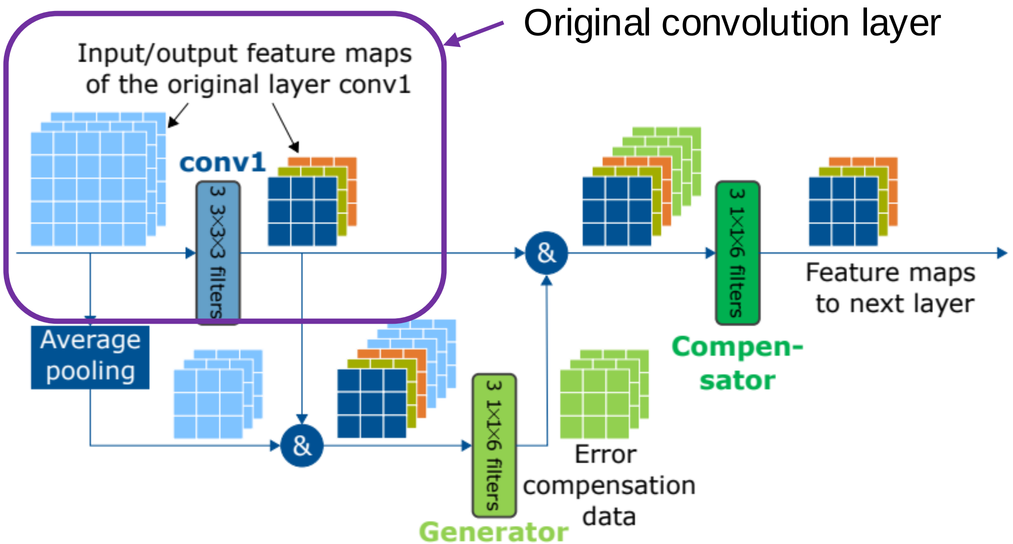
- Log-normal model of weight distribution

$$W_{variation} = W_{nominal} * e^{\theta}$$
$$\theta \sim N(0, \sigma^2)$$



Inference accuracy degradation of neural networks under variations.

Structural Error Compensation Countering Variations

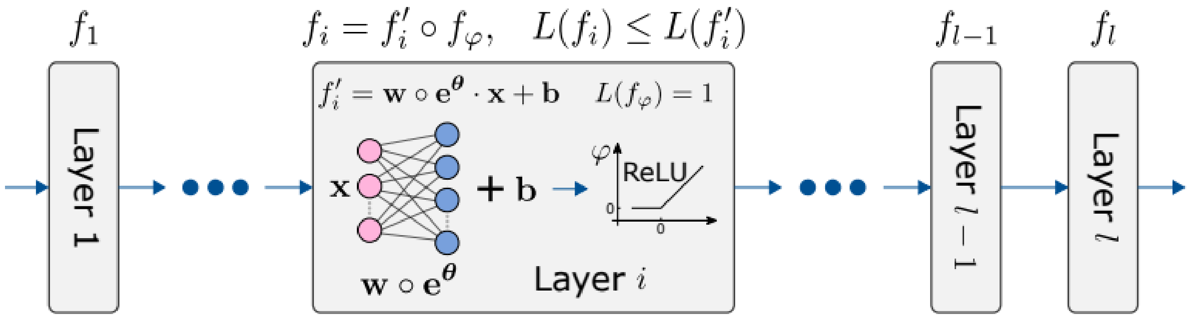


Error compensation for a convolutional layer

- Generator: generate error compensation data
- Compensator: correct the erroneous feature map with the compensation data

Amro Eldebiky, Grace Li Zhang, Georg Boecherer, Bing Li, Ulf Schlichtmann, “CorrectNet: Robustness Enhancement of Analog In-Memory Computing for Neural Networks by Error Suppression and Compensation”, Design, Automation and Test in Europe (DATE), April 2023, **Nominated as Best Paper Award**

Error Suppression



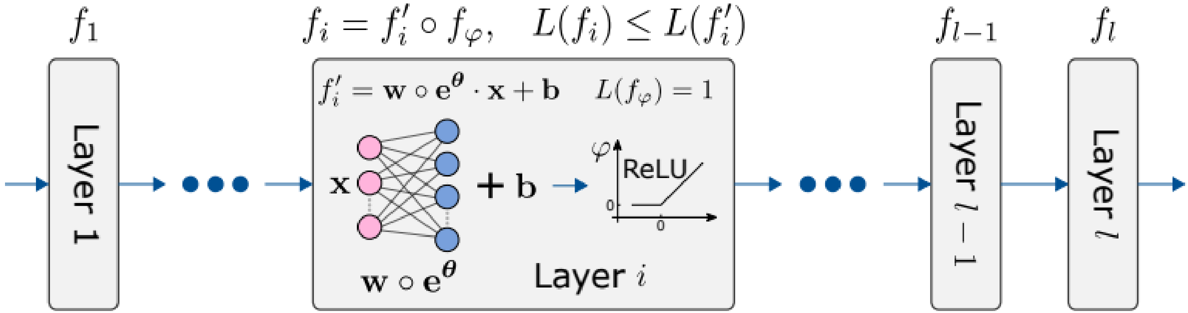
- Uncertainty propagation suppression in a layer with Lipschitz Constant Regularization, θ~N(0,σ)

$$\left|\left(\boldsymbol{w} \circ \boldsymbol{e}^{\boldsymbol{\theta}} \cdot \boldsymbol{x_1} + \boldsymbol{b}\right) - \left(\boldsymbol{w} \circ \boldsymbol{e}^{\boldsymbol{\theta}} \cdot \boldsymbol{x_2} + \boldsymbol{b}\right)\right|_p \le k|\boldsymbol{x_1} - \boldsymbol{x_2}|_p \text{ , } \boldsymbol{k<1}$$

$$sup \frac{\left|\boldsymbol{w} \circ \boldsymbol{e}^{\boldsymbol{\theta}} \cdot (\boldsymbol{x_1} - \boldsymbol{x_2})\right|_p}{|\boldsymbol{x_1} - \boldsymbol{x_2}|_p} = \left\|\boldsymbol{w} \circ \boldsymbol{e}^{\boldsymbol{\theta}}\right\|_p \le k$$

Bound e^θ by $\mu_{e^\theta} + 3\sigma_{e^\theta}$: $\quad \|\boldsymbol{w}\|_p \le \frac{k}{\mu_{e^\theta}+3\sigma_{e^\theta}} = \lambda$

Error Suppression



- Using the 2-norm (spectral norm) → the maximum singular value of the matrix
- Singular values of W are the square roots of the eigenvalues of W^TW

$$L = L_{CE} + \beta * \sum_{w_l \in \boldsymbol{W}} \left\| \boldsymbol{w}_{\boldsymbol{l}}^{\boldsymbol{T}} \boldsymbol{w}_{\boldsymbol{l}} - \lambda^2 \boldsymbol{I} \right\|^2$$

$$\lambda = \frac{k}{\mu_{e^\theta} + 3\sigma_{e^\theta}} = \frac{k}{e^{\frac{\sigma^2}{2}} + 3\sqrt{\left(e^{\sigma^2} - 1\right) e^{(\sigma^2)}}}$$

Experimental Results

Network/ Dataset	Original model Accuracy		CorrectNet Accuracy	CorrectNet Overhead	
	σ= 0	σ= 0.5	σ= 0.5	Weight	#Layers
VGG16/ Cifar100	70.52%	1.69%	67.01%	1.03%	4
VGG16/ Cifar10	93.2%	16.01%	91.29%	0.58%	3
LeNet/ Cifar10	80.89%	25.29%	74.9%	3.47%	1
LeNet/ MNIST	98.79%	84.58%	97.74%	5%	2

Thank you for your attention!