Resources Constrained Learning for Edge Intelligence

Van-Tam NGUYEN

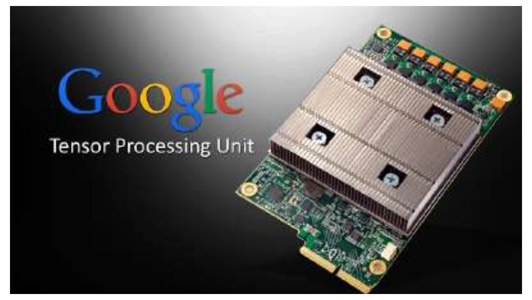
van-tam.nguyen@telecom-paris.fr

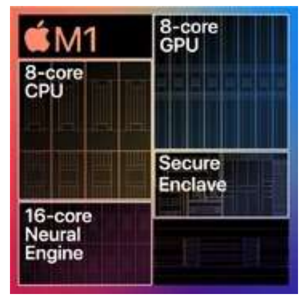
April 2nd, 2024

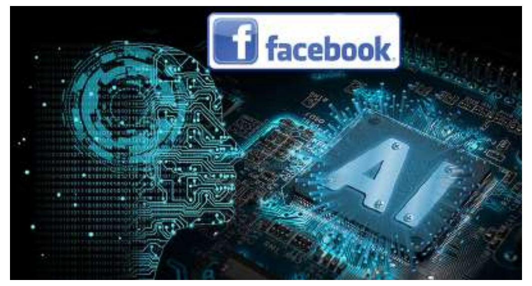
Market

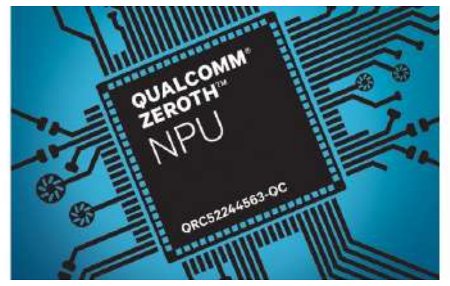
AIoT/Embedded AI market > 1/3 AI market

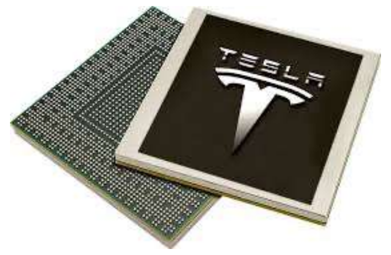
- Global AIoT Market : $144.07 Billion by 2028, growing at a CAGR of 38.1% (1)
- Global AI Market : $422.37 Billion, growing at a CAGR of 39.4% (2)











Dojo

Intel (Gaudi2)

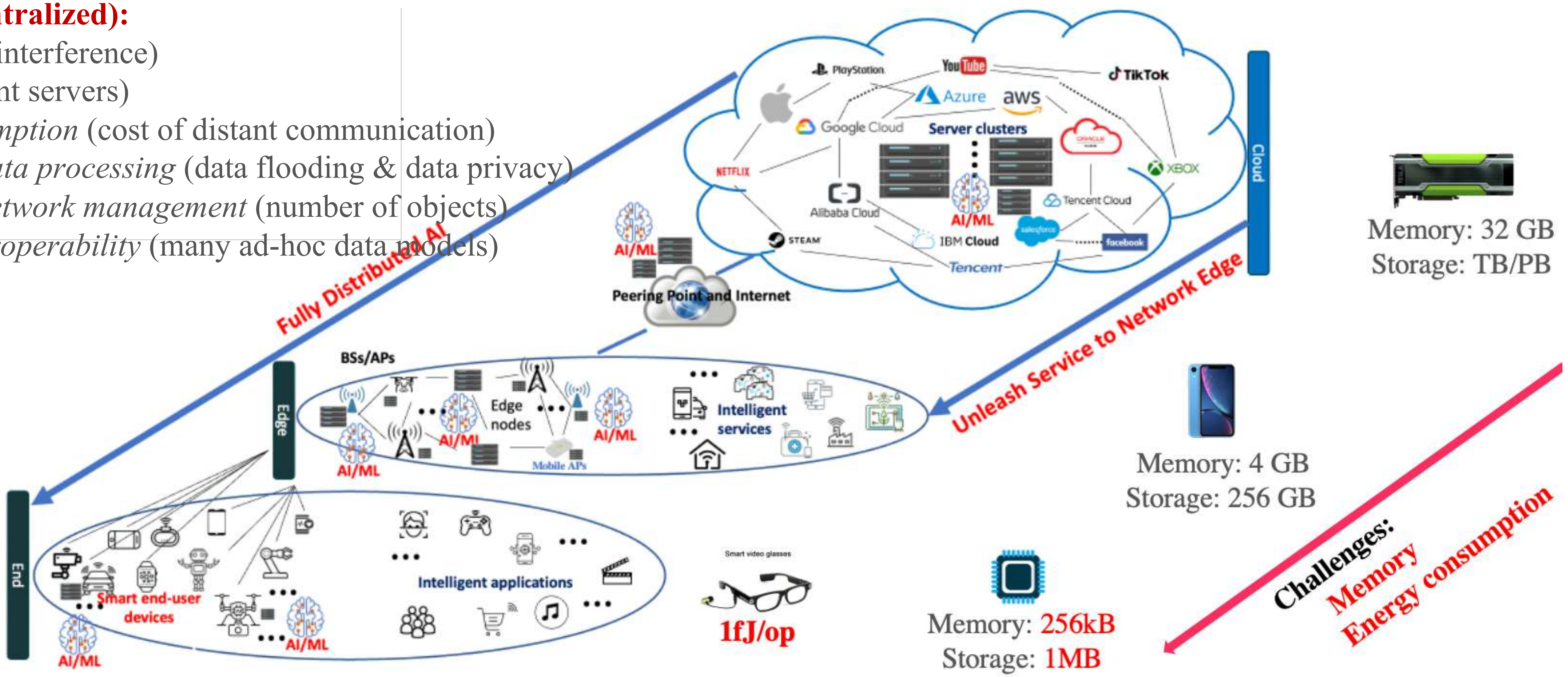
Sources :
(1) : https://www.linkedin.com/pulse/artificial-intelligence-things-aiot-market-projected-reach-sharma/
(2) : https://www.bloomberg.com/press-releases/2022-06-27/-422-37-billion-global-artificial-intelligence-ai-market-size-likely-to-grow-at-39-4-cagr-during-2022-2028-industry

AI in telecom at is all-time high

End-to-End AI for 6G

Problems in (centralized):

- *Connectivity* (interference)
- *Latency* (distant servers)
- *Energy consumption* (cost of distant communication)
- *Centralized data processing* (data flooding & data privacy)
- *Centralized network management* (number of objects)
- *Semantic interoperability* (many ad-hoc data models)



Advantages (distributed):

- *Reduced data transfer*
- *Low latency*
- *Reduced network consumption*
- *Increased security and privacy*
- *Network flexibility*
- *Increased semantic interoperability*

Is Edge AI Possible?

Training : Much larger Memory Footprint

On-Device Training Under 256KB Memory

NEURAL INFORMATION PROCESSING SYSTEMS 2022

Ji Lin[1*] **Ligeng Zhu**[1*] **Wei-Ming Chen**[1] **Wei-Chen Wang**[1] **Chuang Gan**[2] **Song Han**[1]
[1]MIT [2]MIT-IBM Watson AI Lab
https://tinyml.mit.edu/on-device-training

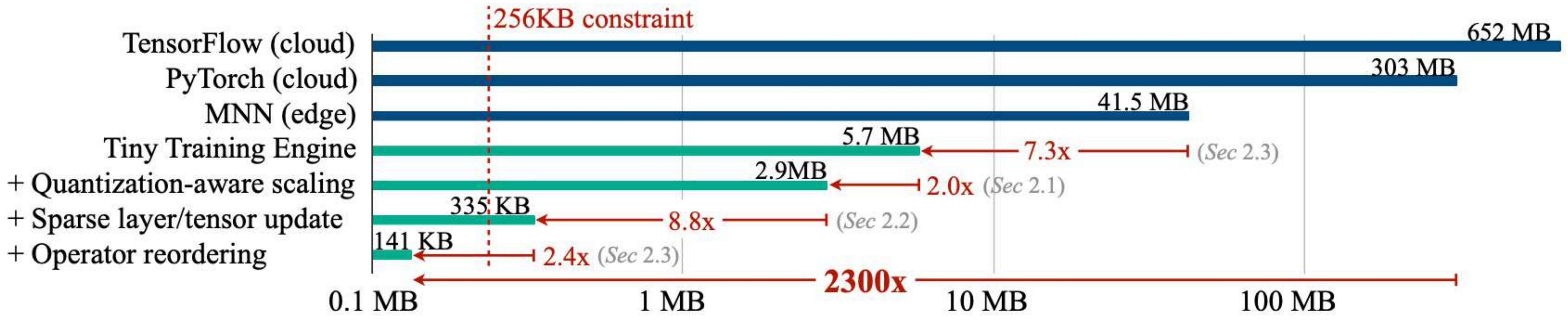
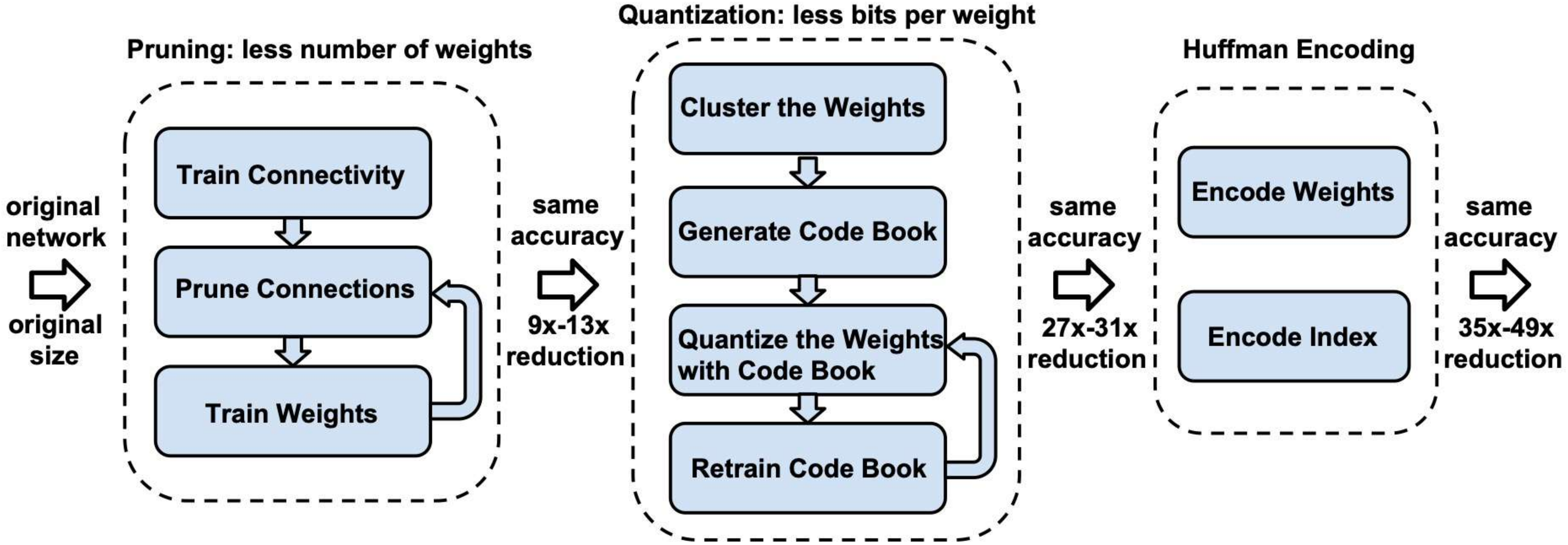


Figure 1. Algorithm and system co-design reduces the training memory from 303MB (PyTorch) to 141KB with the same transfer learning accuracy, leading to 2300× reduction. The numbers are measured with MobilenetV2-w0.35 [60], batch size 1 and resolution 128×128. It can be deployed to a microcontroller with 256KB SRAM.

AI Inference on the Edge is Possible

Deep Neural Network : Over-parameterization

- AlexNet : 240 MB
- VGG-16 : 552 MB



Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *ICML 2016*.

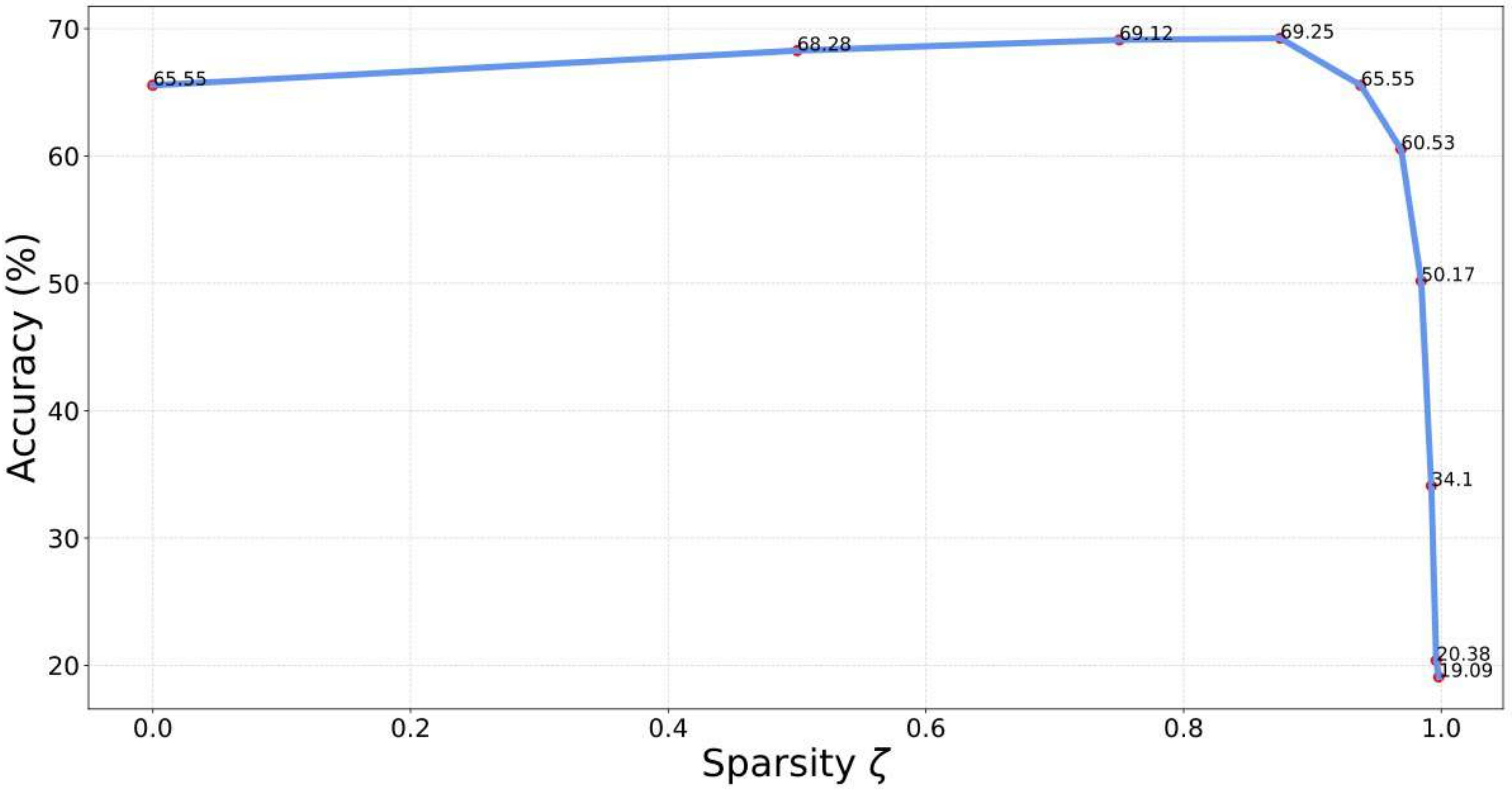
- **Deep Compression**
  - AlexNet : 6.9 MB (35x)
  - VGG-16 : 11.3 MB (49x)

Outline

Compression for Inference

Unstructured Pruning and Sparsity

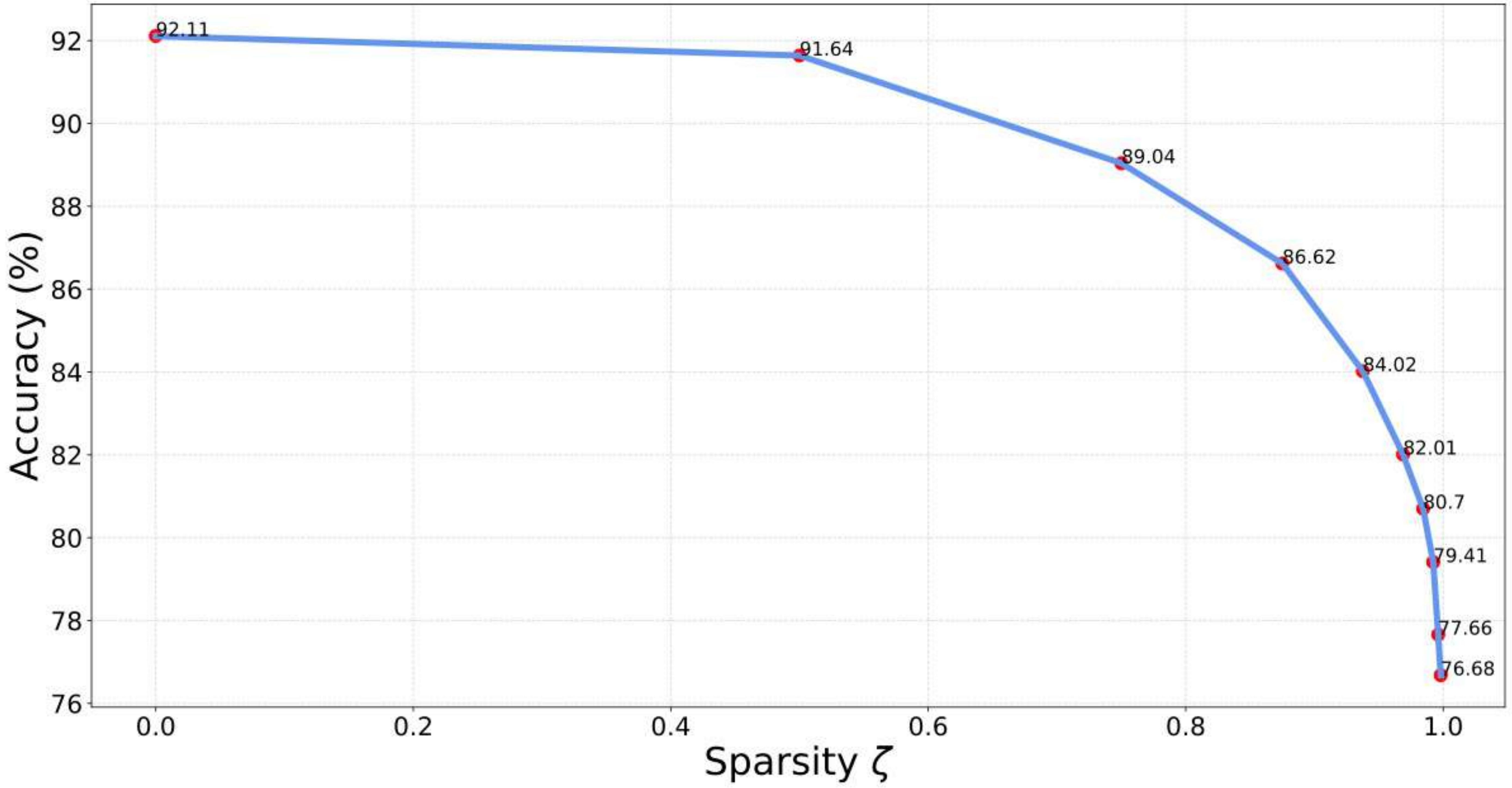
- LeNet5
- Cifar1(



Compression for Inference

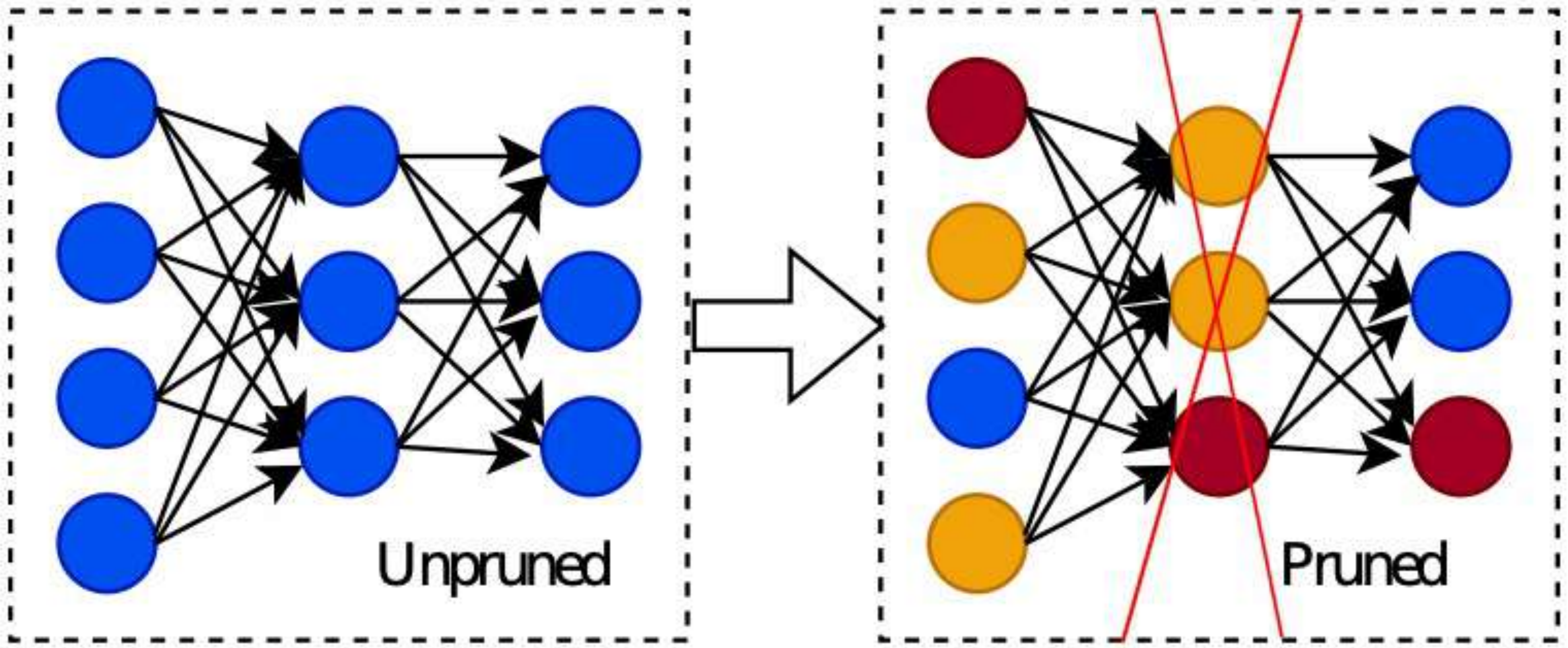
Unstructured Pruning and Sparsity

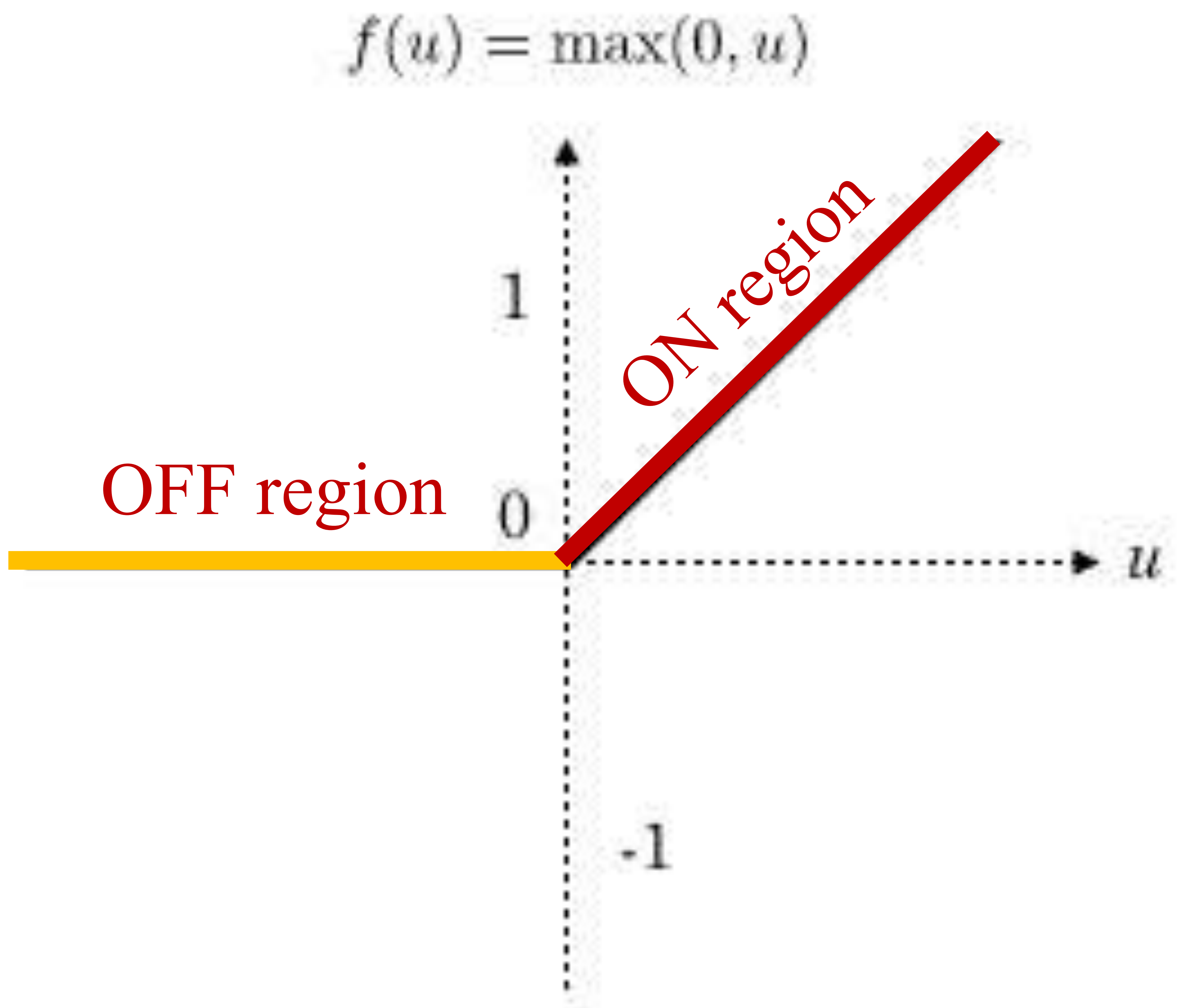
- Swin Transformer
- Cifar10



Compression for Inference

Depth Reduction ?



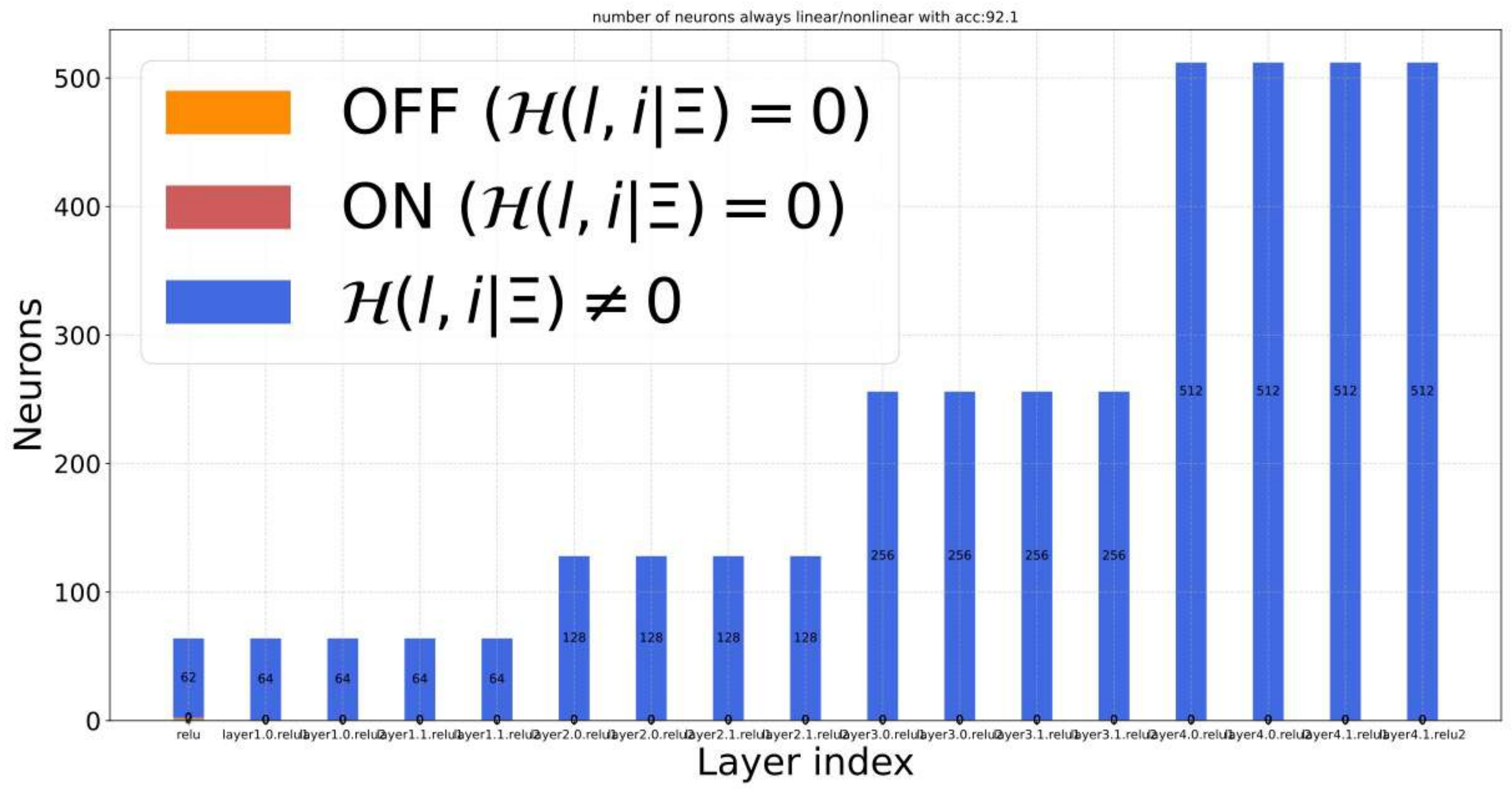


$$\boldsymbol{y}_{l,i}^{\xi} = \mathrm{ReLU}\left(\boldsymbol{z}_{l,i}^{\xi}\right)$$

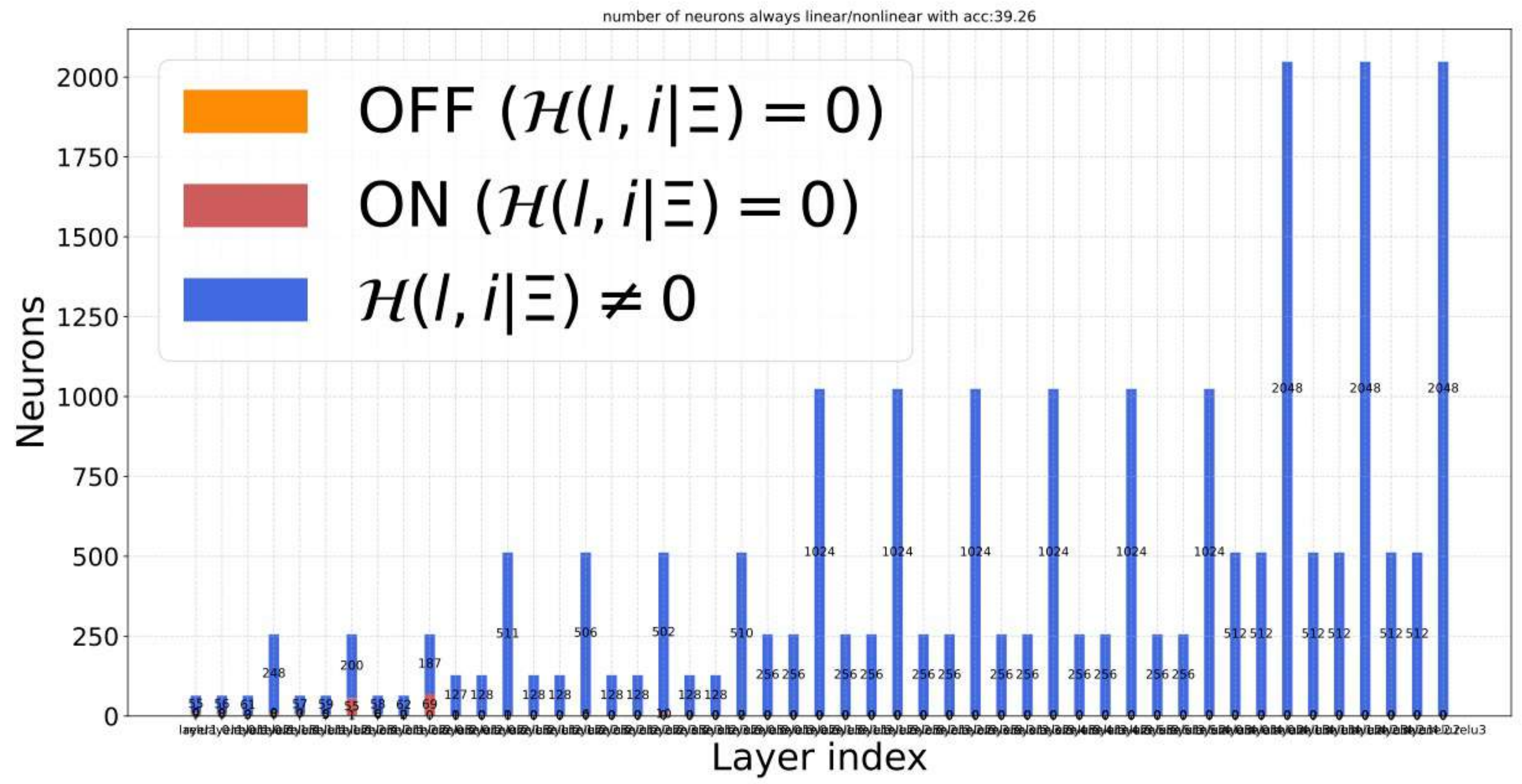
Compression for Inference

Depth Reduction ?

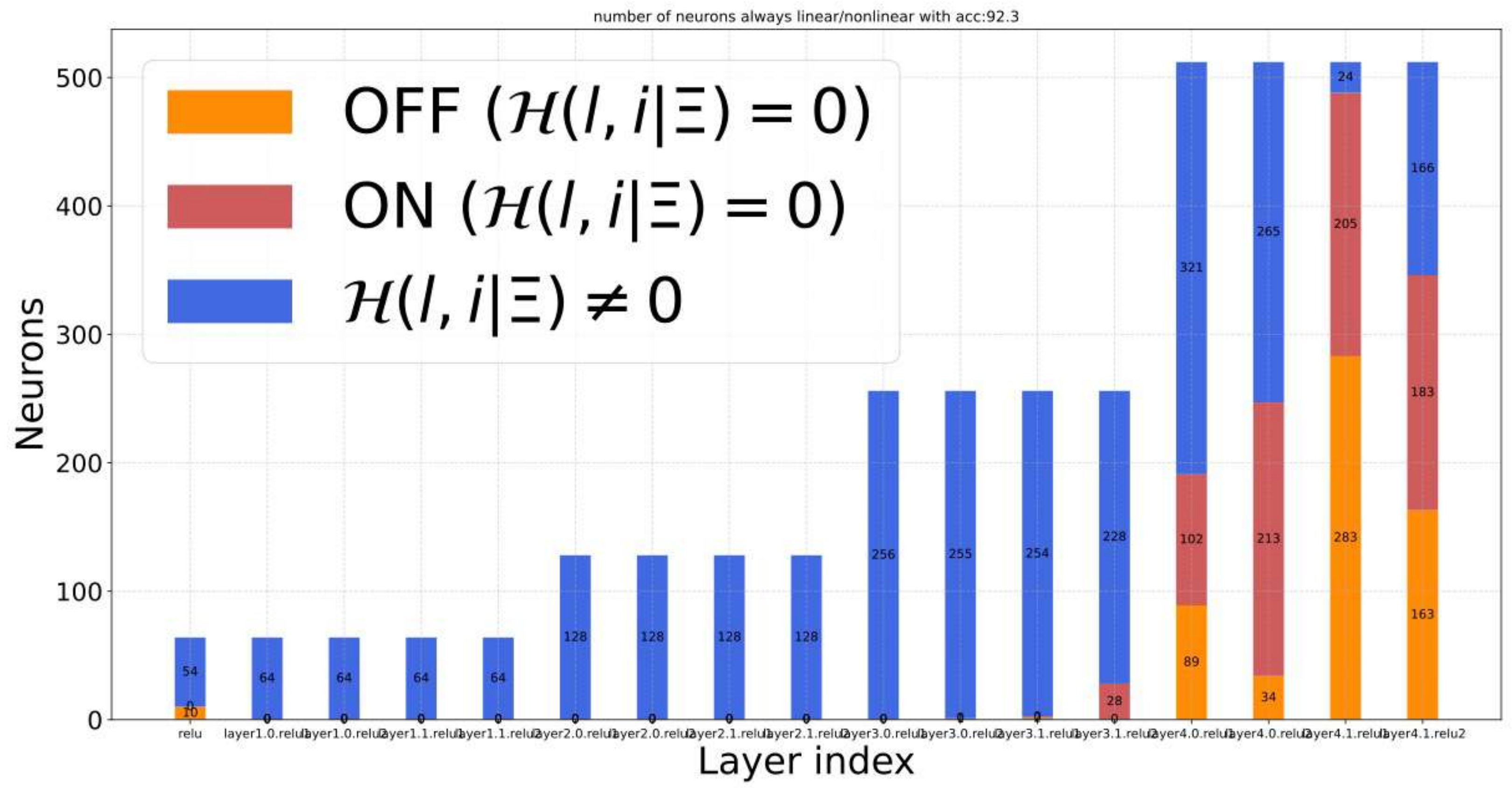
Resnet18-CIFAR10



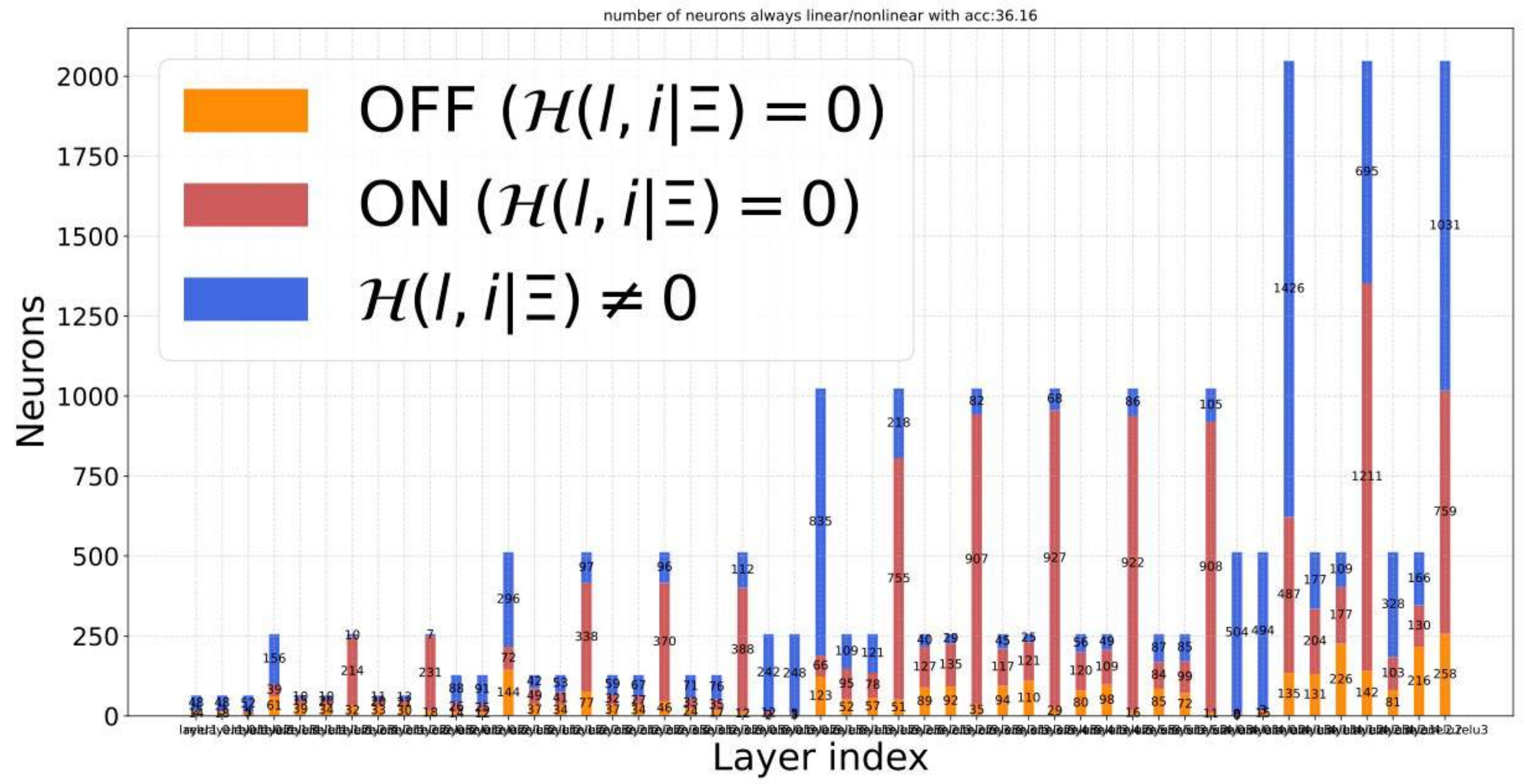
Resnet50-TinyImageNet



Pruning rate: 0.9921875

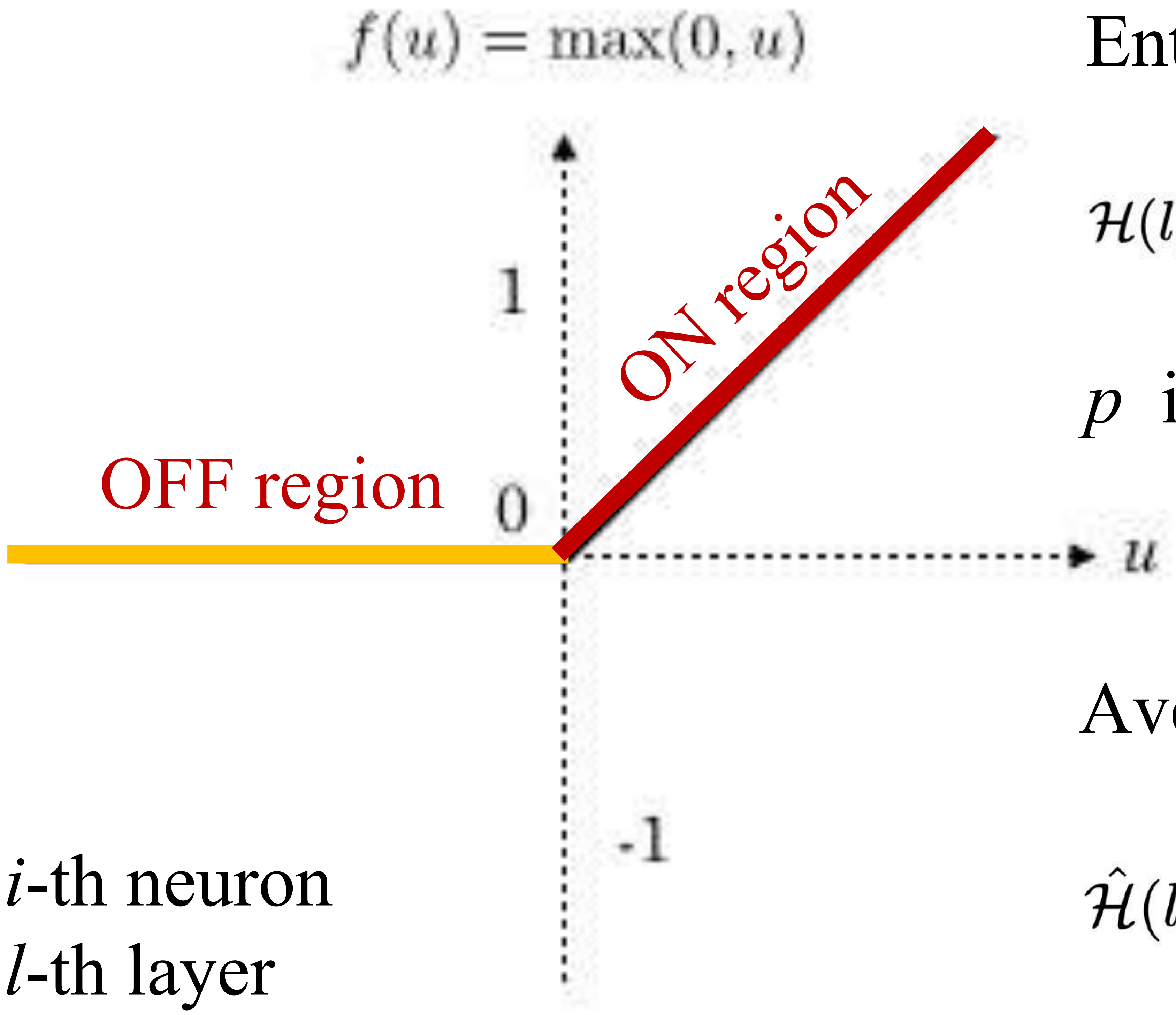


Pruning rate: 0.998046875



Compression for Inference

Entropy Guided Pruning



i-th neuron
l-th layer
ξ-th sample in dataset Ξ

$$\boldsymbol{y}_{l,i}^{\xi} = \mathrm{ReLU}\left(\boldsymbol{z}_{l,i}^{\xi}\right)$$

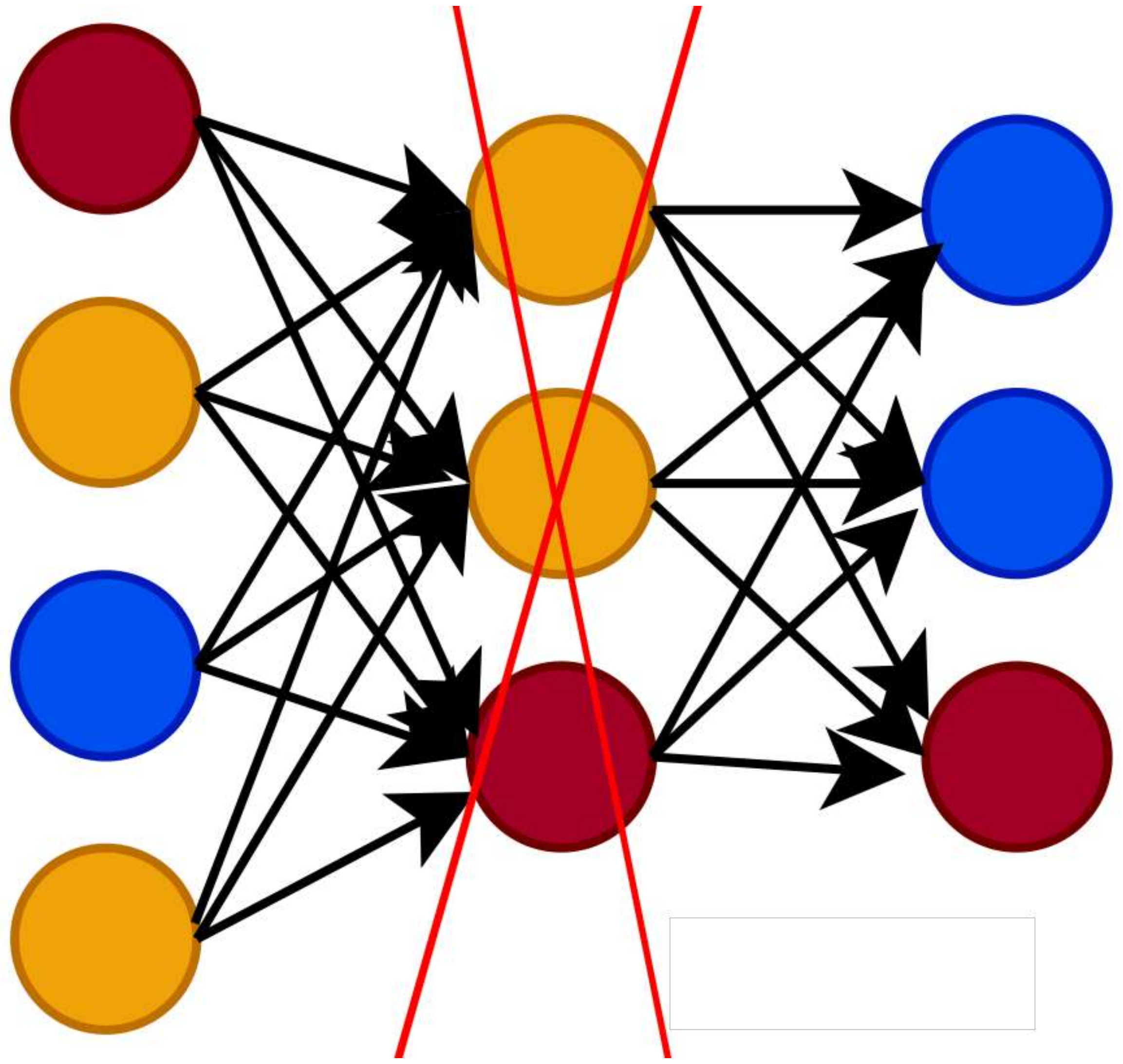
Entropy of the *i*-th neuron at the *l*-th layer:

$$\mathcal{H}(l,i|\Xi) = -\frac{1}{\log(2)}\left\{p^{\mathrm{ON}}(l,i|\Xi)\log\left[p^{\mathrm{ON}}(l,i|\Xi)\right] + p^{\mathrm{OFF}}(l,i|\Xi)\log\left[p^{\mathrm{OFF}}(l,i|\Xi)\right]\right\}$$

p is the frequency of the *i*-th neuron to be in the ON/OFF state

Average entropy for *l*-th layer

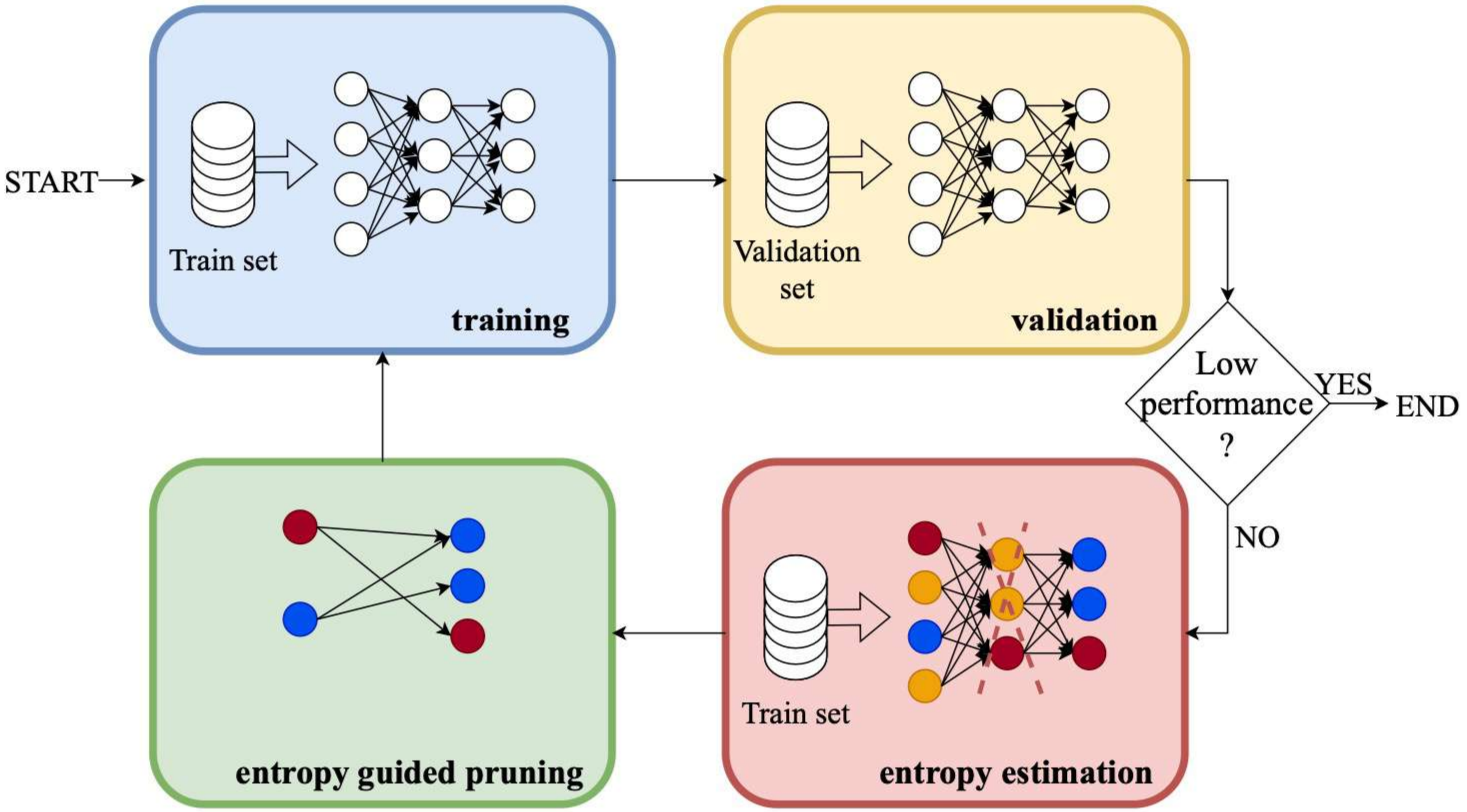
$$\hat{\mathcal{H}}(l|\Xi) = \frac{1}{N_l}\sum_i \mathcal{H}(l,i|\Xi)$$



Entropy = 0 => PRUNED

Compression for Inference

Entropy Guided Pruning



- Iterative pruning based on the entropy of different layers.
- More pruning should be applied to layers with **lower entropy** (more likely to reach zero entropy).
- To minimize the impact on model performance, more pruning should be done on **smaller magnitude weights**.

Compression for Inference

Entropy Guided Pruning

Pruning irrelevance meter :

$$\mathcal{I}_l = \underbrace{\hat{\mathcal{H}}(l|\Xi)}_{\text{Layer's Entropy}} \cdot \frac{1}{\underbrace{||\theta_l||_0}_{\text{Cardinality of the non-zero weights in the } l\text{-th layer (Magnitude)}}} \sum_i |\theta_{l,i}|$$

The larger this value is, the least we are interested in removing parameters from this layer

Pruning relevance:

$$\mathcal{R}_l = \begin{cases} \frac{\sum_j \mathcal{I}_j}{\mathcal{I}_l} & \mathcal{I}_l \neq 0 \\ 0 & \mathcal{I}_l = 0 \end{cases}$$

Amount of parameters to be removed at the *l*-th layer:

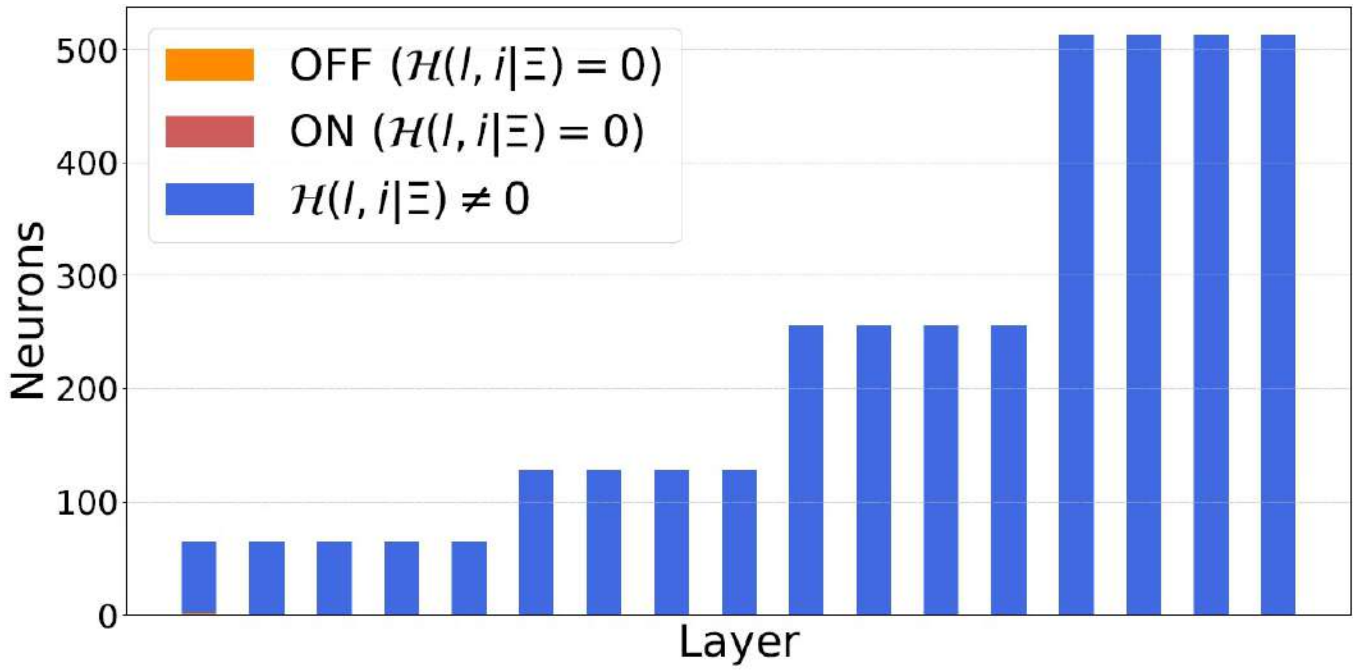
$$||\theta_l||_0^{\text{pruned}} = ||\theta||_0^{\text{pruned}} \cdot \frac{\exp[\mathcal{R}_l - \max_k(\mathcal{R}_k)]}{\sum_j \exp[\mathcal{R}_j - \max_k(\mathcal{R}_k)]}$$

Compression for Inference

Experiments

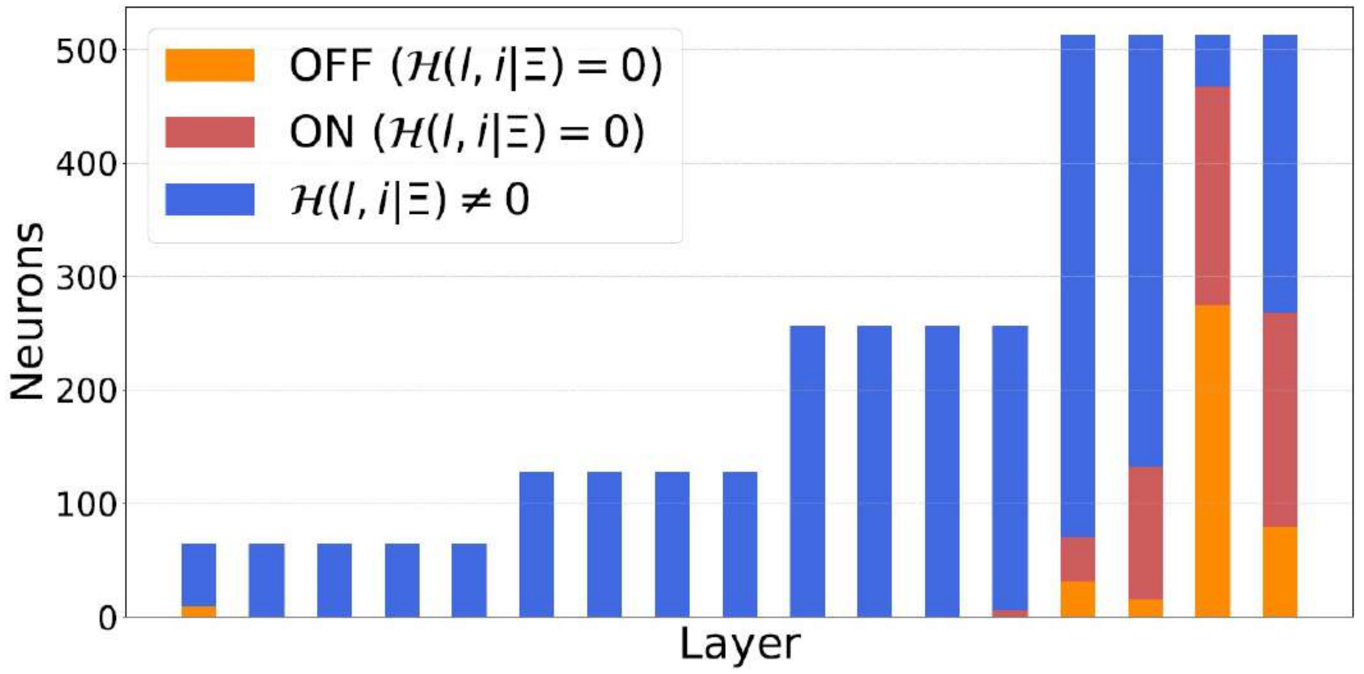
ResNet-18 trained on CIFAR-10

Sparsity 0%



Test accuracy: 92.10%

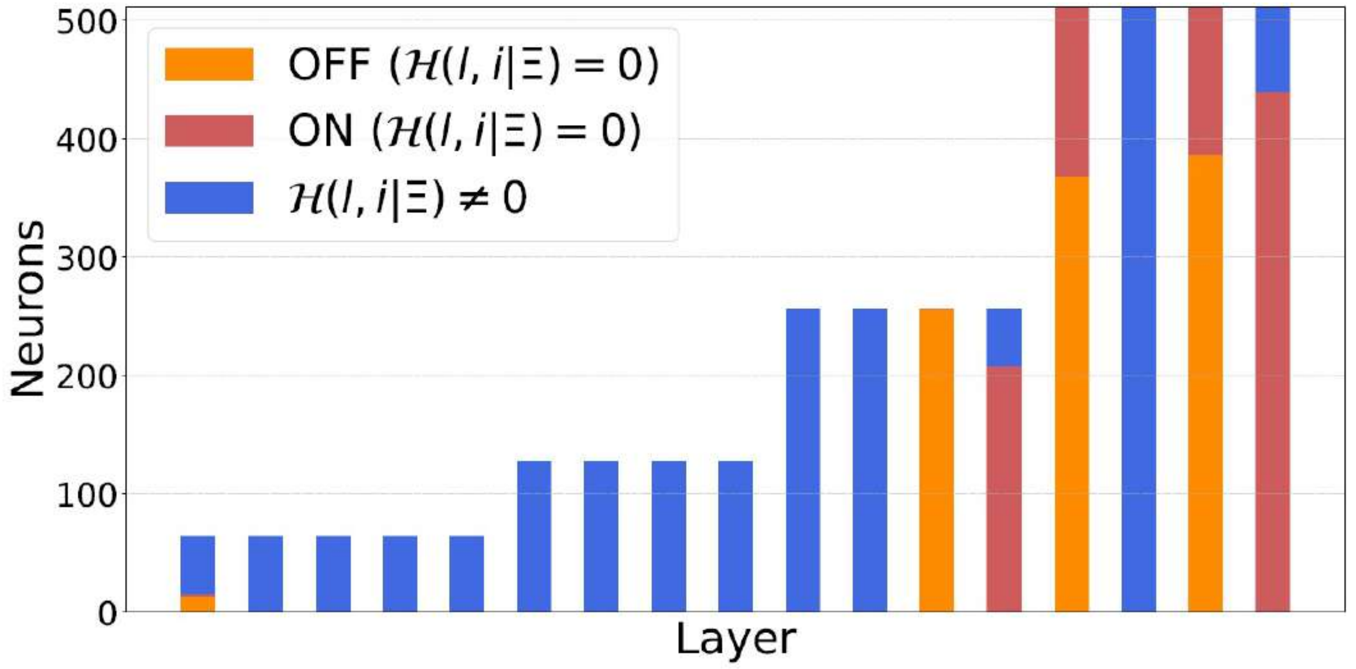
Sparsity 98.4%



Unstructured Pruning

Test accuracy: 92.73%

Sparsity 98.4%



Entropy Guided Pruning

Test accuracy: 93.12%

Compression for Inference

Experiments

ResNet-18 on CIFAR-10
Swin-Treansformer on Tiny-ImageNet

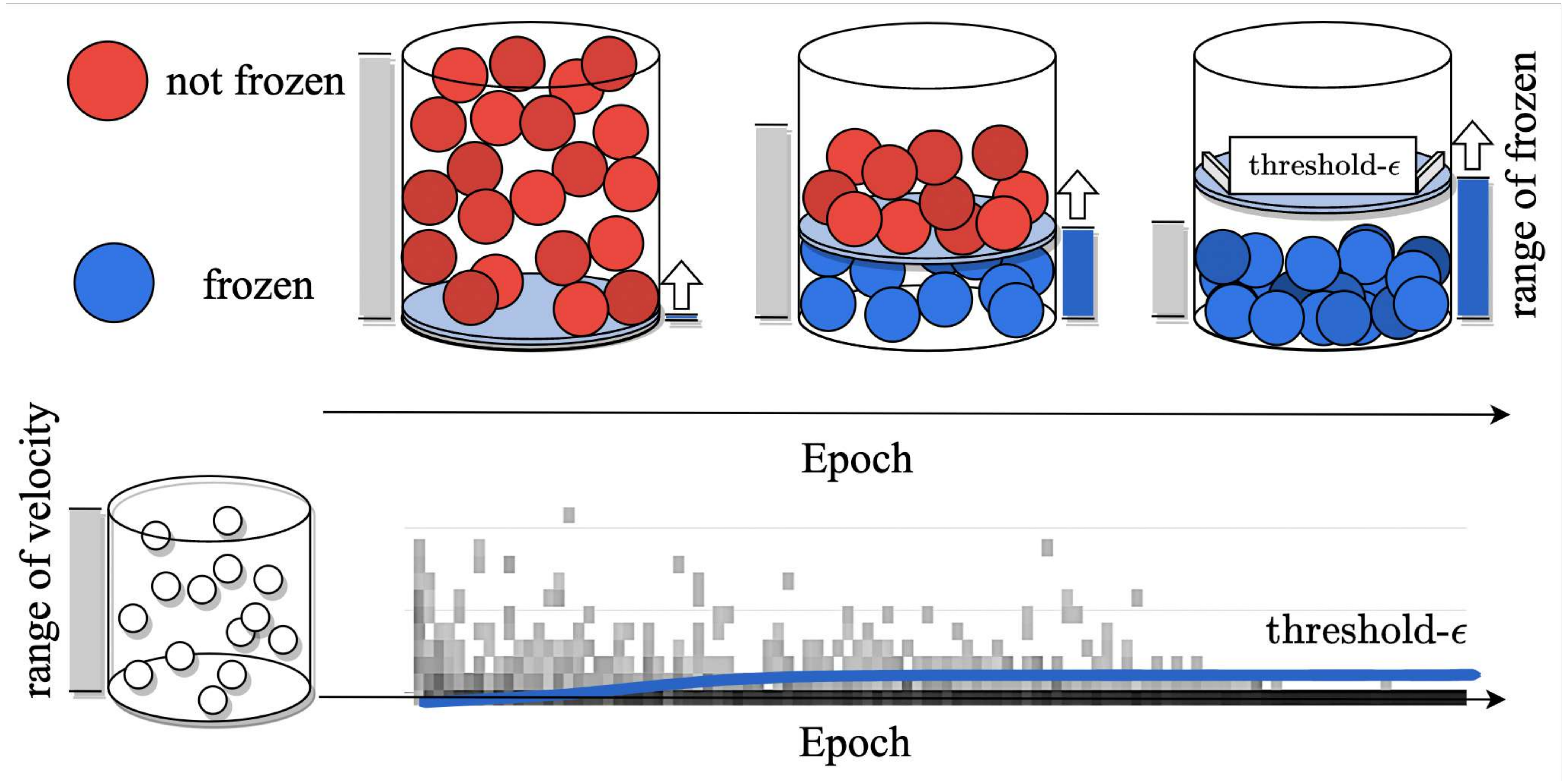
Notice:
Non-linear activation in ResNet-18: ReLU,
Non-linear activation in Swin-T: GELU

Model	Sparisty	EGP	Lay. rem.	TOP-1
Resnet-18 on CIFAR-10	0.0		0/17	92.10
	50.0		0/17	92.56
		✓	1/17	92.36
	75.0		0/17	93.00
		✓	1/17	92.81
	93.8		0/17	93.03
		✓	**3/17**	92.93
	98.4		0/17	92.73
		✓	**3/17**	**93.12**
Swin-T on Tiny-Inet	0.0		0/12	**75.38**
	50.0		0/12	74.06
		✓	1/12	71.48
	75.0		0/12	72.02
		✓	2/12	70.28
	93.8		0/12	67.58
		✓	3/12	66.58
	98.4		0/12	63.46
		✓	**6/12**	62.30

Z. Liao, V. Quétu, V. T. Nguyen and E. Tartaglione, " Can Unstructured Pruning Reduce the Depth in Deep Neural Networks?," in *Workshop on Resource Efficient Deep Learning for Computer Vision, ICCV Workshop* 2023.

Save Computation at Training

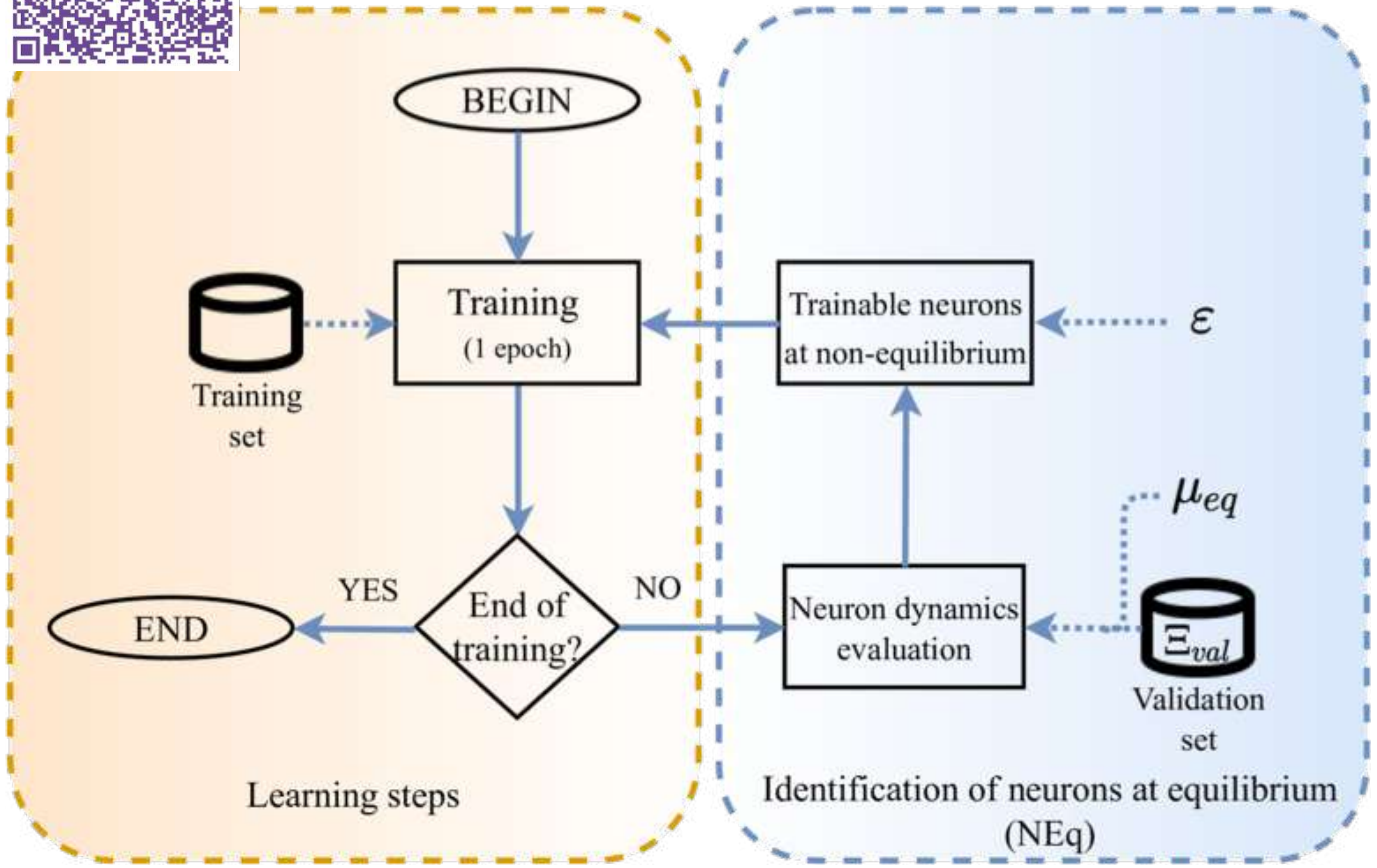
SCoTTi: Save Computation at TrainingTime

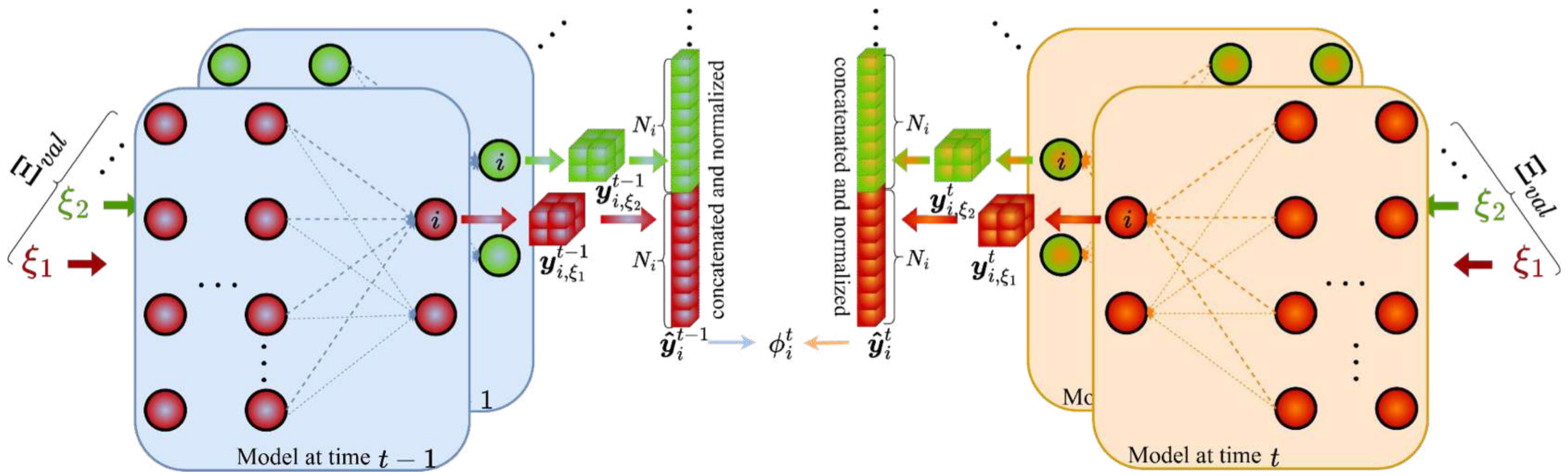


Save Computation at Training

SCoTTi: Save Computation at TrainingTime

To update or not to update? Neurons at Equilibrium in deep models (2022)





The cosine similarity between the outputs of the same neuron in two consecutive epochs :

$$\phi_i^t = \sum_{\xi \in \Xi_{val}} \sum_{n=1}^{N_i} \hat{y}_{i,n,\xi}^t \cdot \hat{y}_{i,n,\xi}^{t-1}.$$

$$\Delta\phi_i^t = \phi_i^t - \phi_i^{t-1}.$$

The velocity: $v_{\Delta\phi_i}^t = \Delta\phi_i^t - \mu_{eq} v_{\Delta\phi_i}^{t-1}$ The condition for a frozen state : $\left| v_{\Delta\phi}^t \right| < \varepsilon$

Save Computation at Training

SCoTTi: Save Computation at TrainingTime

GD: The Ultimate Optimizer (2022)

- Work started from Almeida et al. (1999) and Baydin et al. (2018).
- Rule for hyper-gradients (eg. dynamic learning rate).
- Proposed a software implementation.

$$w_{i+1} = w_i - \alpha \frac{\partial f(w_i)}{\partial w_i}$$

$$w_{i+1} = w_i - \alpha_{i+1} \frac{\partial f(w_i)}{\partial w_i}$$

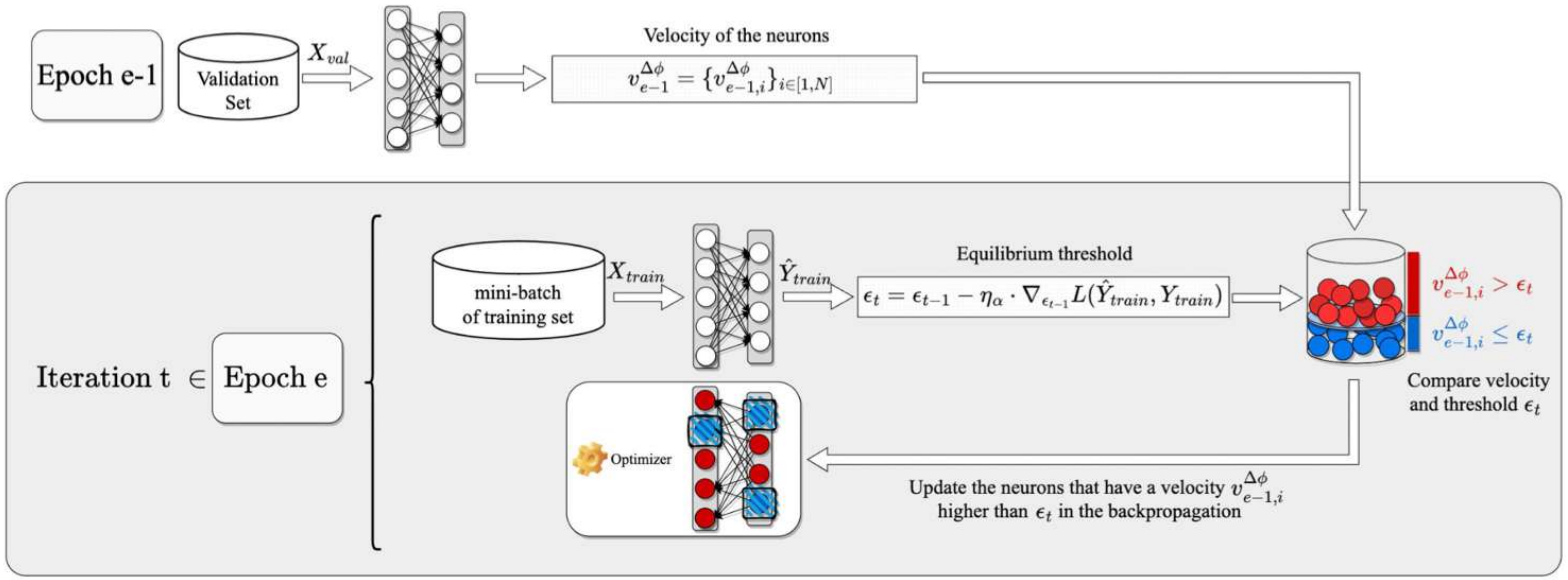
$$\alpha_{i+1} = \alpha_i - \kappa \frac{\partial f(w_i)}{\partial \alpha_i}$$

If α is too small, the optimizer runs very slowly, whereas if α is too large, the optimizer fails to converge

$$\frac{\partial f(w_i)}{\partial \alpha_i} = \frac{\partial f(w_i)}{\partial w_i} \cdot \frac{\partial w_i}{\partial \alpha_i} = \frac{\partial f(w_i)}{\partial w_i} \cdot \frac{\partial \left(w_{i-1} - \alpha_i \frac{\partial f(w_{i-1})}{\partial w_{i-1}} \right)}{\partial \alpha_i} = \frac{\partial f(w_i)}{\partial w_i} \cdot \left(- \frac{\partial f(w_{i-1})}{\partial w_{i-1}} \right)$$

Save Computation at Training

SCoTTi: Save Computation at TrainingTime



- Together with the learning rate, we also learn the threshold ε which varies at the iteration scale.
- The velocity is however evaluated once after every epoch, on the validation set.
- As the learning rate is also optimized, the lower it becomes the higher ε becomes, as the neurons will converge faster to their final state.

Z. Li, E. Tartaglione and V. T. Nguyen "SCoTTi: Save Computation at Training Time with an adaptive framework," in *Workshop on Resource Efficient Deep Learning for Computer Vision, ICCV Workshop* 2023.

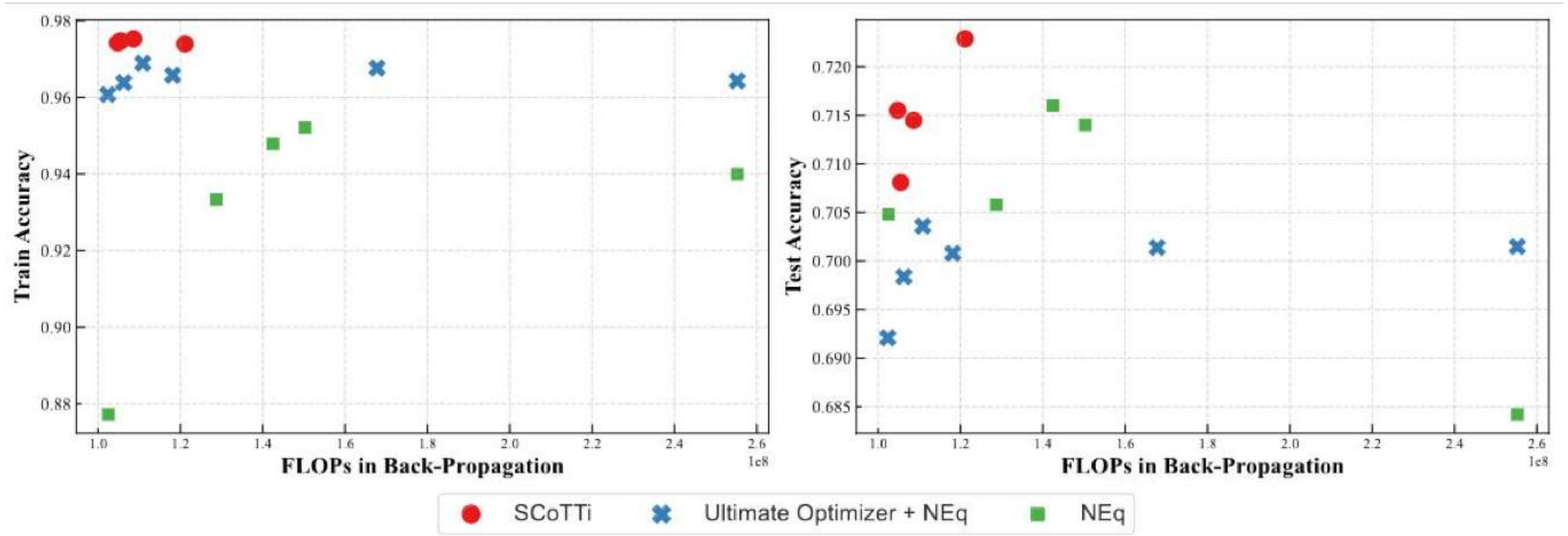
Save Computation at Training

Experiments

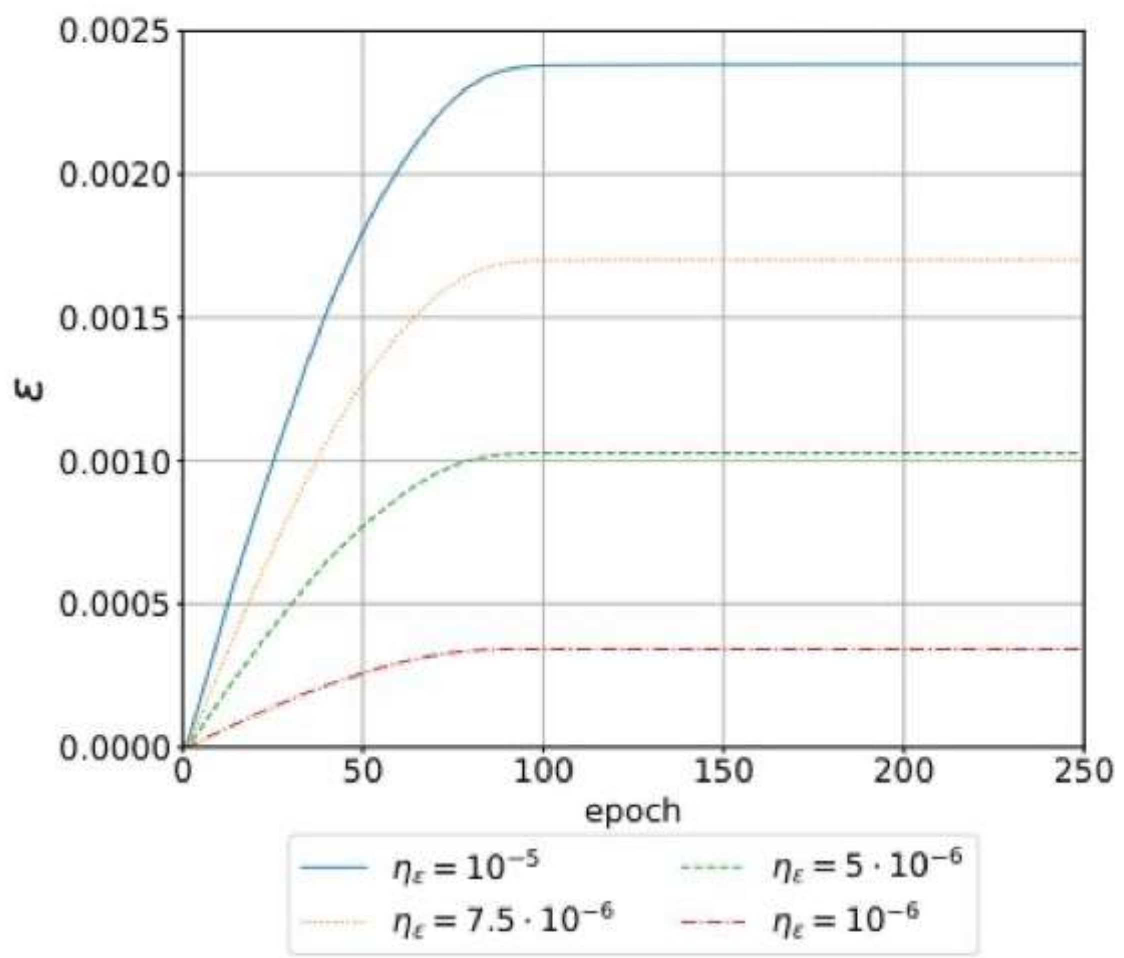
Dataset	Architecture	Optimization Approach			FLOPs saved	Top-1
		NEq [16]	Ultimate optim. [15]	Learned ϵ		
CIFAR-10 [39]	VGG-16 [40]				00.00%	88.54%
		✓			37.41%	89.86%
			✓		00.00%	**92.76%**
		✓	✓		30.98%	92.70%
		✓	✓	✓	**43.66%**	92.58%
	Swin-T* [41]				00.00%	91.59%
		✓			39.66%	90.96%
			✓		00.00%	91.65%
		✓	✓		48.84%	91.74%
		✓	✓	✓	**58.76%**	**91.77%**
CIFAR-100 [39]	ResNet-32 [42]				00.00%	68.42%
		✓			38.80%	69.97%
			✓		00.00%	70.06%
		✓	✓		59.89%	69.24%
		✓	✓	✓	**60.59%**	**70.43%**
	ResNet-56 [42]				00.00%	69.69%
		✓			41.12%	71.40%
			✓		00.00%	70.15%
		✓	✓		56.58%	70.36%
		✓	✓	✓	**58.97%**	**71.55%**
Clipart [43]	ResNet-18 [42]				00.00%	73.21%
		✓			38.06%	72.19%
			✓		00.00%	73.01%
		✓	✓		49.33%	72.60%
		✓	✓	✓	**53.86%**	**73.21%**
Painting [43]	ResNet-18 [42]				00.00%	64.51%
		✓			27.94%	62.14%
			✓		00.00%	60.82%
		✓	✓		**77.34%**	63.46%
		✓	✓	✓	76.92%	**65.44%**
Tiny ImageNet [43]	MobileNet-v2* [44]				00.00%	55.69%
		✓			53.28%	56.40%
			✓		00.00%	60.02%
		✓	✓		80.83%	60.53%
		✓	✓	✓	**86.44%**	**60.68%**

Save Computation at Training

Experiments



- Ablation study on CIFAR-100, with a ResNet-56, show the effectiveness of SCoTTi compared to NEq and Ultimate Optimizer.
- In some cases the performance also improves.



Toward AI Training on the Edge

On-Device Training

On-Device Training Under 256KB Memory (2022)

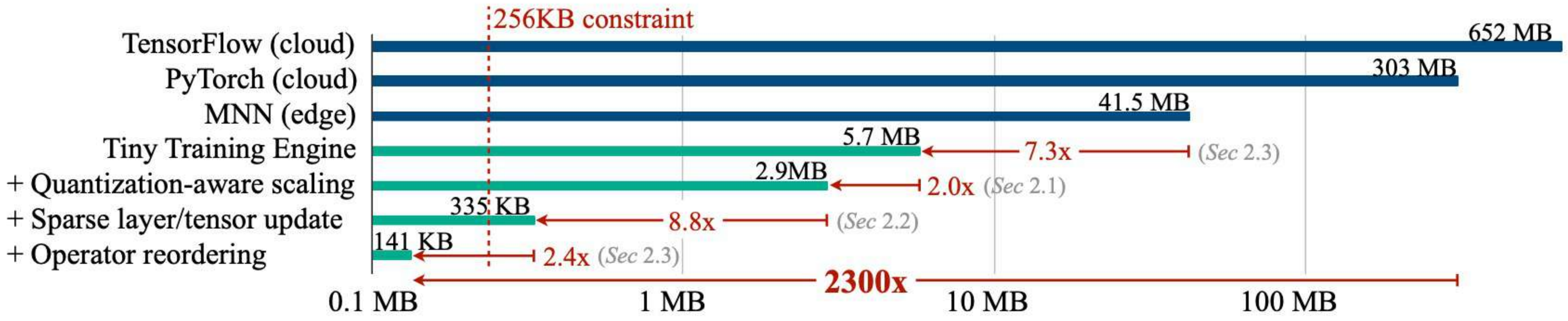


Figure 1. Algorithm and system co-design reduces the training memory from 303MB (PyTorch) to 141KB with the same transfer learning accuracy, leading to 2300× reduction. The numbers are measured with MobilenetV2-w0.35 [60], batch size 1 and resolution 128×128. It can be deployed to a microcontroller with 256KB SRAM.

TinyEngine: A Memory-Efficient Inference Library

- Eliminating runtime overheads,
- Memory scheduling
- In-place depth-wise convolution

Sparse Layer/tensor update

- Static sparse update schemes thanks to an offline evolutionary search

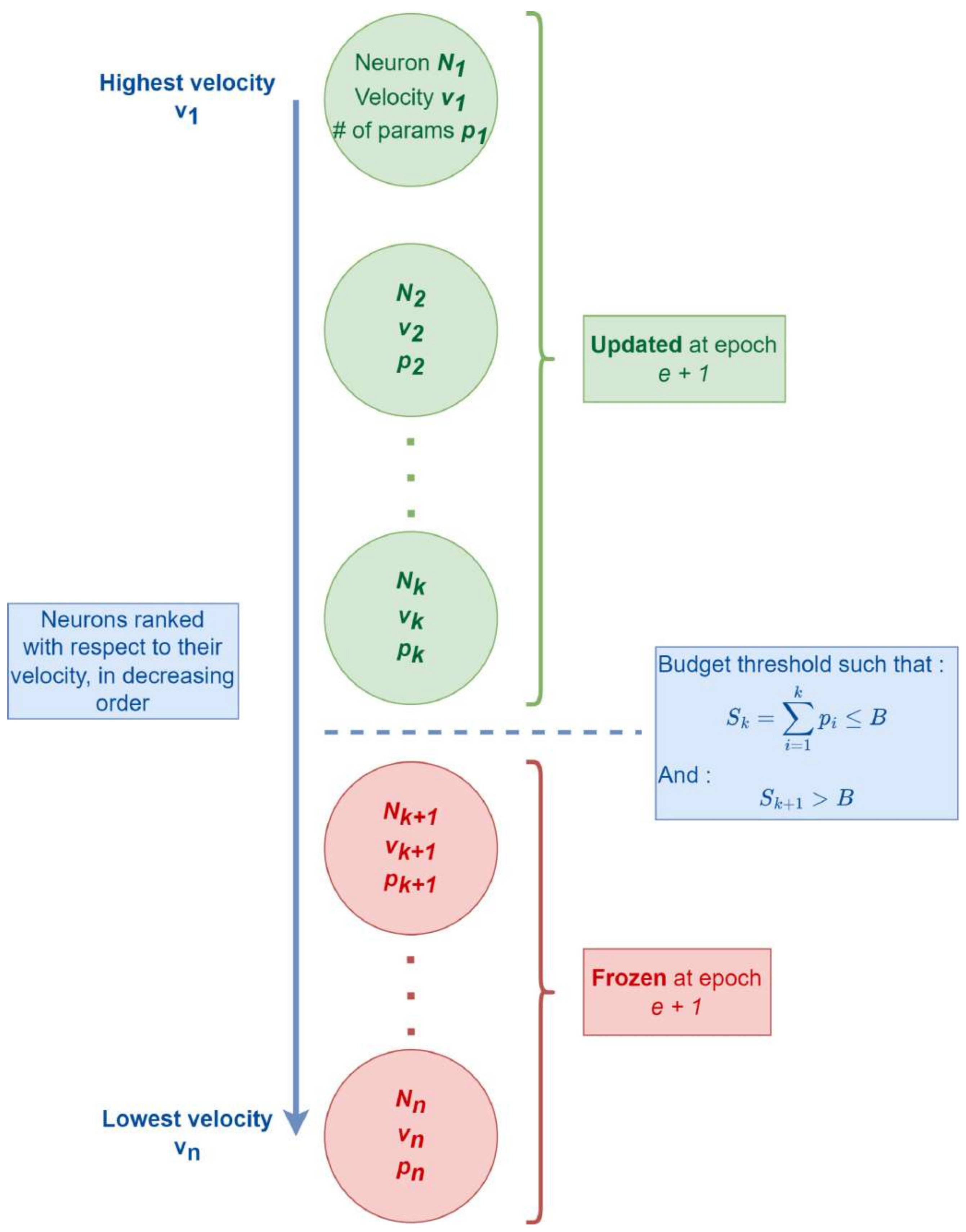
Toward AI Training on the Edge

On-Device Training

Velocity-Based Neuron Selection

1. Neuron outputs at each epoch on a fixed validation subset to determine their "*velocity*".
2. NEq consists in the **progressive freezing of neurons** as their velocity falls below a threshold ε.
3. We **update the k fastest neurons while staying within a specified parameter budget**.

=> This approach maintains high test accuracies and prevents update costs from exceeding a predetermined memory limit.

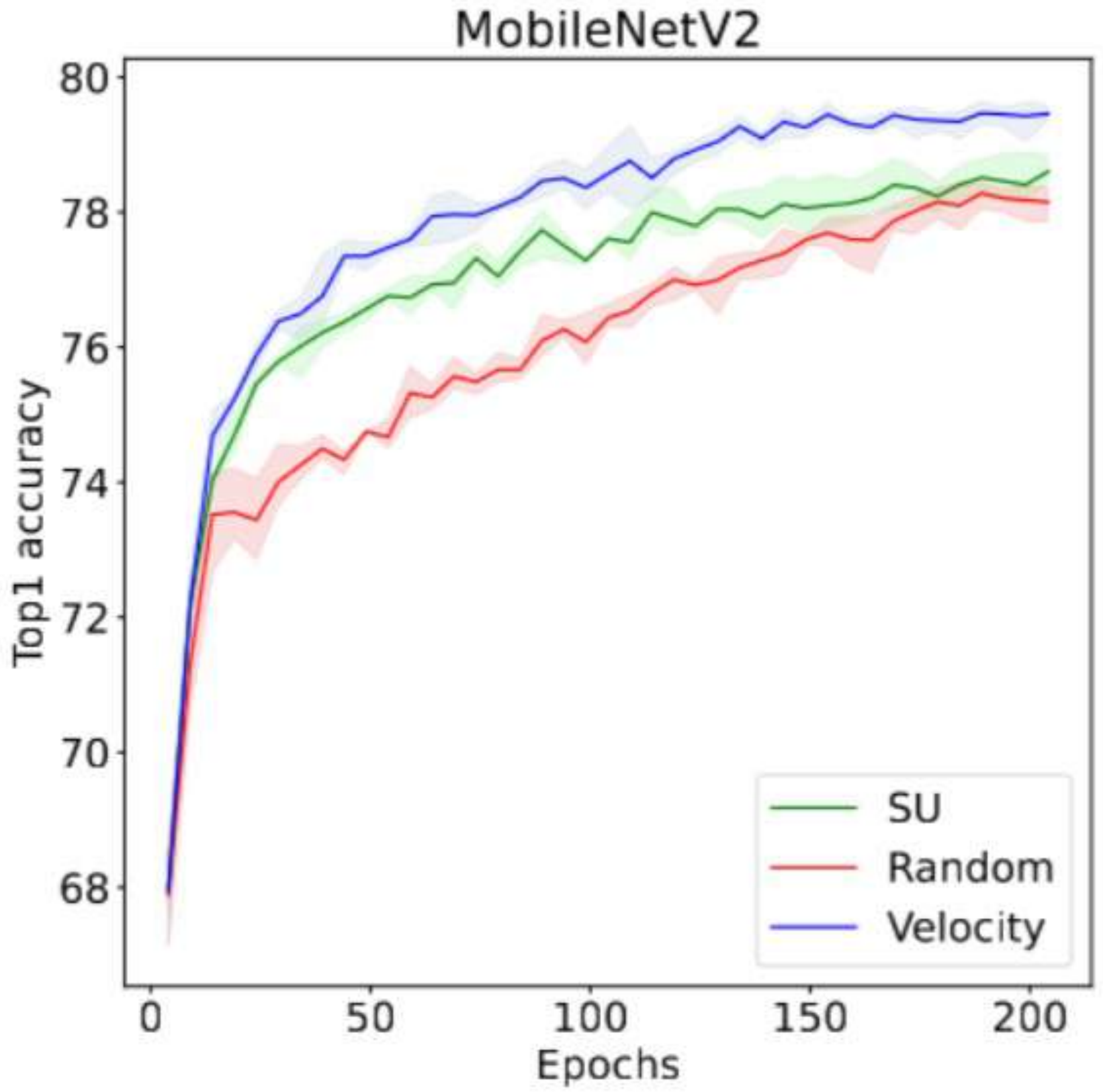


Selection of neurons to update given a network of $\boldsymbol{n}$ neurons and a budget $\boldsymbol{B}$

Toward AI Training on the Edge

On-Device Training

Velocity-Based Neuron Selection



MobileNetV2 testing accuracy through training on C100, with only 8.8% of the network's parameters updated

% of network updated	$\mathcal{B}^w_{\text{max}}$	Method	Cifar 10	Cifar 100	VWW
8.8	192 311	Sparse Update	95.13±0.21	78.60±0.22	90.66±0.29
		Velocity	95.25±0.29	79.46±0.12	91.40±0.16
		Random	94.41±0.13	78.15±0.26	90.29±0.05
21.2	464 639	Sparse Update	95.30±0.10	78.84±0.20	91.29±0.18
		Velocity	95.36±0.07	79.67±0.28	91.48±0.39
		Random	94.61±0.16	78.28±0.31	90.51±0.25
30.8	675 540	Sparse Update	95.16±0.29	78.62±0.18	91.46±0.21
		Velocity	95.49±0.16	79.43±0.19	91.57±0.20
		Random	94.57±0.20	78.58±0.11	90.58±0.10

Pretrained MobileNetV2 final top1 test accuracies (for the first epoch the neurons to update are given by the associated SU scheme)

Toward AI Training on the Edge

On-Device Training

Velocity-Based Neuron Selection

% of network updated	$\mathcal{B}^w_{\max}$	Method	Flowers	Food	Pets	CUB
8.8	192 311	Sparse Update	93.77±0.38	77.81±0.26	85.82±0.22	67.82±0.29
		Velocity	93.03±0.47	79.16±0.16	85.50±0.17	67.52±0.05
		Random	92.19±0.17	77.78±0.00	85.50±0.28	65.56±0.45
21.2	464 639	Sparse Update	94.28±0.36	78.26±0.07	84.63±0.15	68.04±0.28
		Velocity	93.34±0.08	79.63±0.17	84.91±0.82	68.23±0.61
		Random	92.43±0.10	78.26±0.07	84.73±0.29	66.30±0.13
30.8	675 540	Sparse Update	94.22±0.14	78.03±0.08	84.38±0.28	67.59±0.22
		Velocity	93.77±0.26	79.56±0.24	84.44±0.50	68.26±0.36
		Random	92.83±0.03	79.00±0.11	84.39±0.42	66.17±0.42

Pretrained MobileNetV2 final top1 test accuracies
(for the first epoch the neurons to update are
given by the associated SU scheme)

Toward AI Training on the Edge

On-Device Training

Final top1 test accuracies (for the first epoch the neurons to update are randomly selected)

Model	$\mathcal{B}^{w}_{\max}$	Method	Cifar 10	Cifar 100	VWW	Flowers	Food	Pets	CUB
MbV2	192 311	SU	94.88±0.12	78.15±0.13	90.75±0.17	92.70±0.06	75.02±0.23	**86.93±0.22**	66.48±0.41
		Velocity	95.35±0.35	79.41±0.21	90.95±0.16	92.98±0.29	79.18±0.07	85.56±0.47	67.92±0.27
		Random	94.46±0.16	78.03±0.21	90.20±0.13	92.11±0.15	77.57±0.16	85.16±0.07	65.96±0.06
	464 639	SU	95.00±0.08	78.69±0.19	90.80±0.24	92.86±0.33	76.50±0.23	86.64±0.27	67.81±0.23
		Velocity	95.49±0.02	79.52±0.11	91.41±0.19	93.32±0.15	79.67±0.19	84.36±0.76	68.19±0.24
		Random	94.51±0.05	78.49±0.19	90.55±0.24	92.64±0.32	78.48±0.17	84.92±0.21	66.18±0.71
	675 540	SU	95.18±0.17	79.03±0.30	91.03±0.16	93.08±0.09	77.19±0.07	86.42±0.45	67.72±0.32
		Velocity	95.57±0.11	79.21±0.37	91.72±0.15	93.33±0.31	79.68±0.16	84.37±0.28	68.19±0.24
		Random	94.57±0.09	78.41±0.29	90.35±0.01	92.88±0.21	78.90±0.06	84.39±0.41	66.36±0.20
	2 189 760	Baseline	**95.93±0.14**	**79.83±0.29**	**91.80±0.03**	**94.02±0.03**	**80.63±0.10**	82.82±0.18	**69.24±0.34**
Resnet18	980 715	Velocity	95.51±0.10	78.77±0.41	88.78±0.51	90.78±0.24	75.09±0.13	**82.82±0.30**	63.64±0.35
		Random	95.20±0.20	77.98±0.38	88.33±0.45	89.39±0.47	74.57±0.15	79.49±0.51	60.93±0.54
	2 369 480	Velocity	95.36±0.15	**79.12±0.12**	89.16±0.31	**91.02±0.17**	75.72±0.23	82.01±0.60	**63.84±0.39**
		Random	95.68±0.10	78.28±0.22	89.21±0.23	89.53±0.17	75.17±0.12	79.40±0.34	61.42±0.32
	3 444 987	Velocity	95.58±0.21	78.95±0.13	89.17±0.33	90.76±0.15	75.83±0.12	81.45±0.63	63.59±0.03
		Random	95.80±0.09	78.52±0.18	88.92±0.19	89.66±0.30	75.28±0.12	79.28±0.34	61.45±0.32
	11 166 912	Baseline	**96.2±0.13**	78.86±0.08	**89.78±0.24**	90.14±0.26	**76.32±0.08**	79.76±0.63	60.97±0.52
Resnet50	2 059 888	Velocity	97.10±0.07	**82.94±0.35**	93.04±0.15	93.65±0.08	81.10±0.05	**90.11±0.25**	**73.73±0.52**
		Random	96.80±0.06	81.46±0.11	92.13±0.33	94.04±0.18	80.68±0.18	88.92±0.18	72.79±0.15
	4 976 842	Velocity	97.12±0.09	82.79±0.40	**93.37±0.14**	94.84±0.08	81.62±0.17	89.42±0.25	73.55±0.18
		Random	96.97±0.08	82.04±0.11	92.59±0.20	94.23±0.22	81.52±0.11	88.46±0.07	72.40±0.60
	7 235 830	Velocity	97.07±0.08	82.45±0.18	93.21±0.14	**95.03±0.18**	81.76±0.06	88.97±0.60	73.54±0.42
		Random	97.01±0.01	82.27±0.29	92.59±0.21	94.54±0.15	81.88±0.29	88.18±0.63	72.40±0.35
	23 454 912	Baseline	**97.30±0.04**	82.63±0.25	92.91±0.22	94.87±0.20	**82.49±0.15**	87.29±0.34	72.93±0.41

Conclusions and Perspectives

- AI compression for inference are possible because of overparameterization of the DNN, and depth reduction is proposed with Entropy Guided Pruning
- Saving computation at training is necessary because of the enormous computational resources required for backpropagation.
- Training on the Edge : it is not necessary to update all the weights all the time

- Hardware Aware Model Compression (**power in data movement, not computation**)
- Hardware Aware Neural Architecture Search (**real life application => no pretrained model**)
- Optimal Gradient Pruning ()
- Training on the Edge : Taking into account activation in constrained memory budget

Acknowledgements

Zhu Liao, Ziyu Li, Ael Quelennec, Pavlo Mozharovskyi and Enzo Tartaglione

The research leading to these results has received funding from the projects titled "PC2-NF-NAI" and "PC6-FITNESS" in the frame of the program "PEPR 5G et Reseaux du futur", and the European project "ENFIELD".