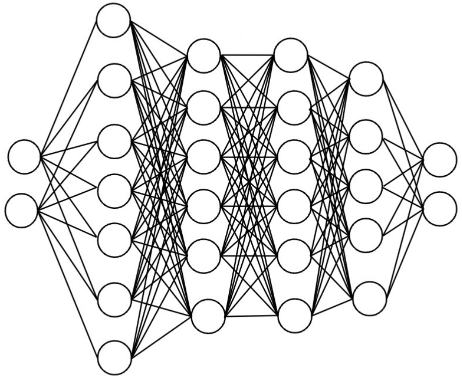
NVIDIA

Neural Network Compression

Maying Shen

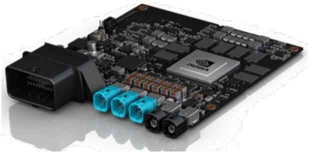
Senior Research Engineer

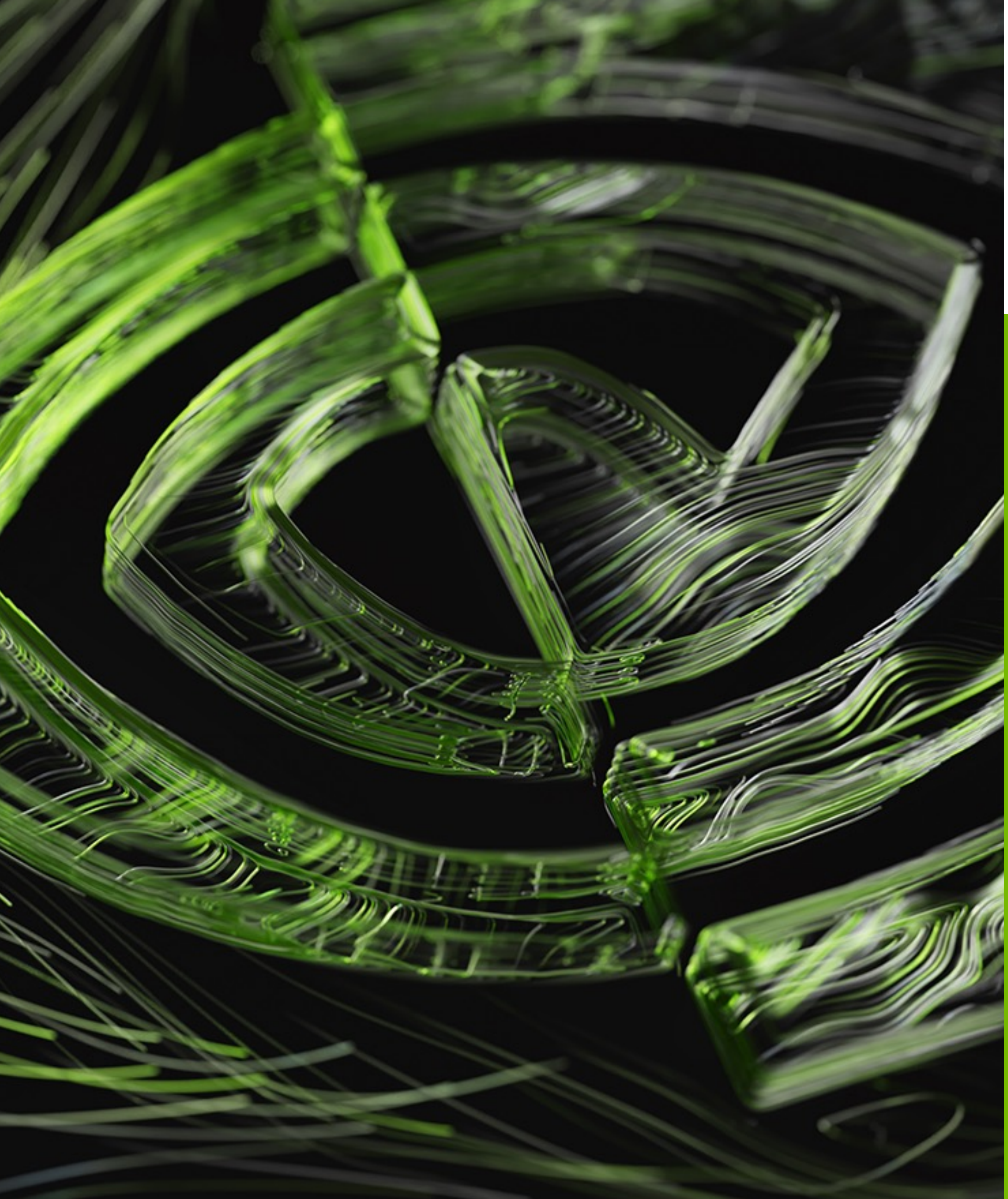
Why Compression?



Compress

Hardware to Deploy



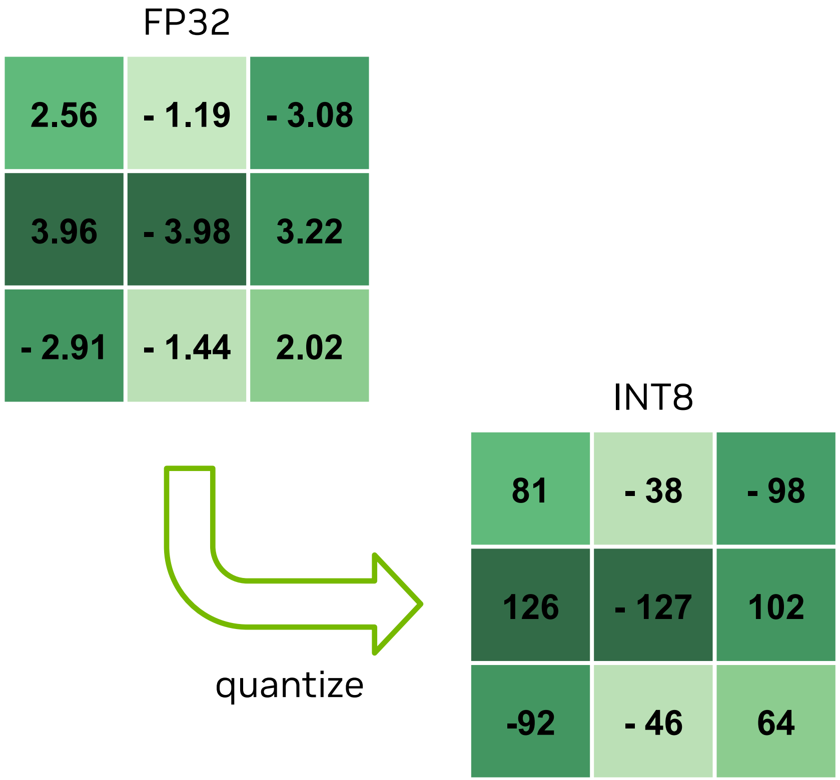


- Network Quantization
- Network Pruning
- Tools for quantization and pruning
- DLA optimization – Live Demo

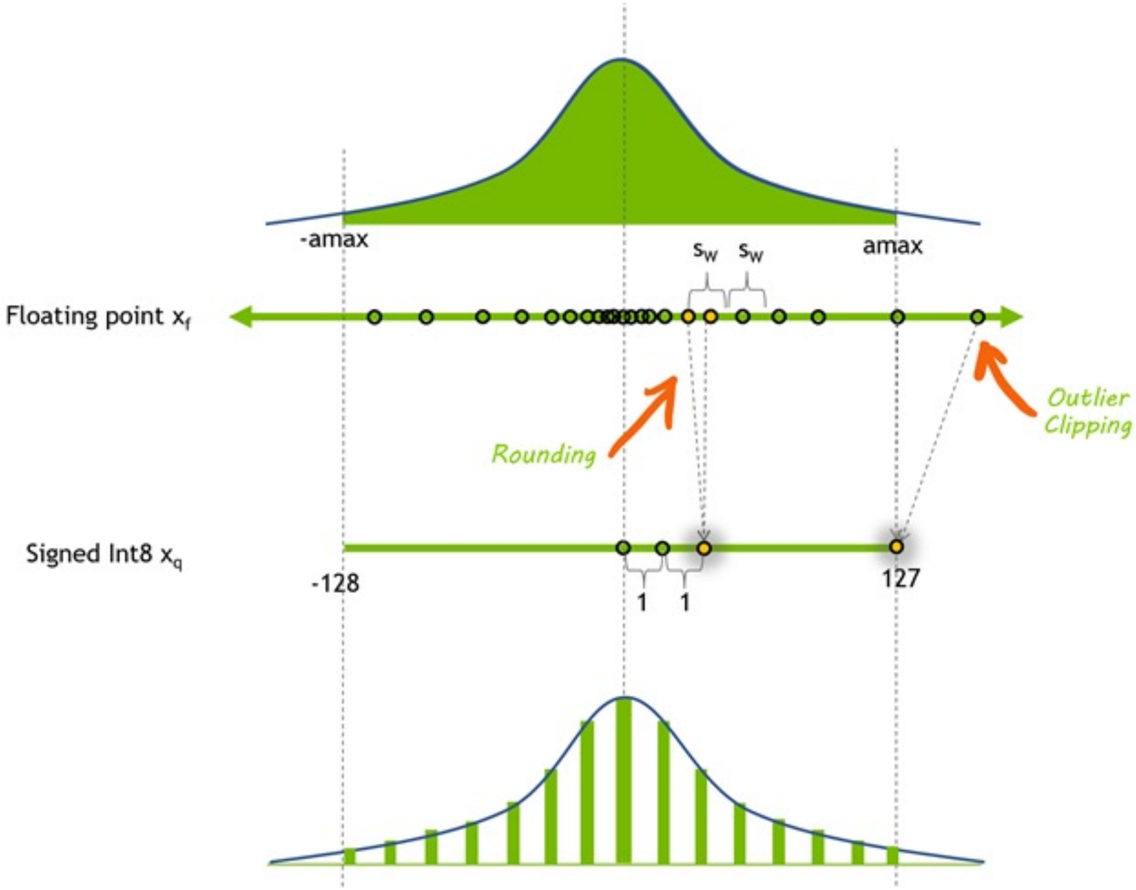
Quantization

Inference with Lower Precision

- Most models are trained in FP32 / FP16 to take advantage of a wider range of numbers.
- Reducing precision reduces compute, memory, and power.
  - NVIDIA GPUs employ faster and cheaper 8-bit Tensor Cores for computation.
  - 32-bit floats to 8-bit integers results in 4x memory reduction and about 2-4x throughput improvement.



Source: https://developer.nvidia.com/blog/improving-int8-accuracy-using-quantization-aware-training-and-tao-toolkit/



- FP32 represent ~4M values in range [-3.4e38, 3.4e38] and about half values in [-1, 1].
- Int8 can represent only 256 values.
- We need a mapping process to convert from FP32 to INT8 representation.

$$x_q = Clip(Round(x_f/scale))$$

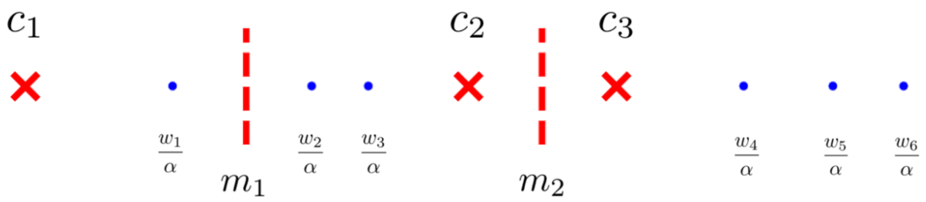
Source: https://developer.nvidia.com/blog/achieving-fp32-accuracy-for-int8-inference-using-quantization-aware-training-with-tensorrt/

From High Precision to Low Precision

From High Precision to Low Precision

$$\min_{\alpha, \mathbf{z}_1, \ldots, \mathbf{z}_N} \quad \mathrm{Loss}(\alpha, \mathbf{Z}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} (w_n - \alpha\, c_k)^2$$

$$\text{s.t.} \quad \mathbf{z}_n^T \mathbf{1} = 1, \quad \mathbf{z}_n \in \{0, 1\}^K$$



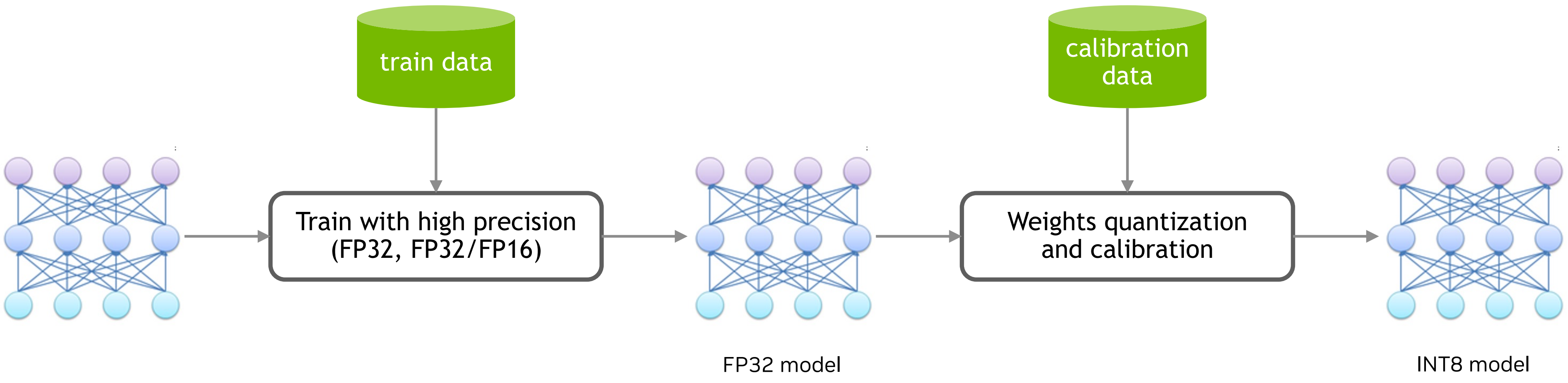
Source: Idelbayev et al. Optimal Quantization using Scaled Codebook CVPR 2021

Codebook

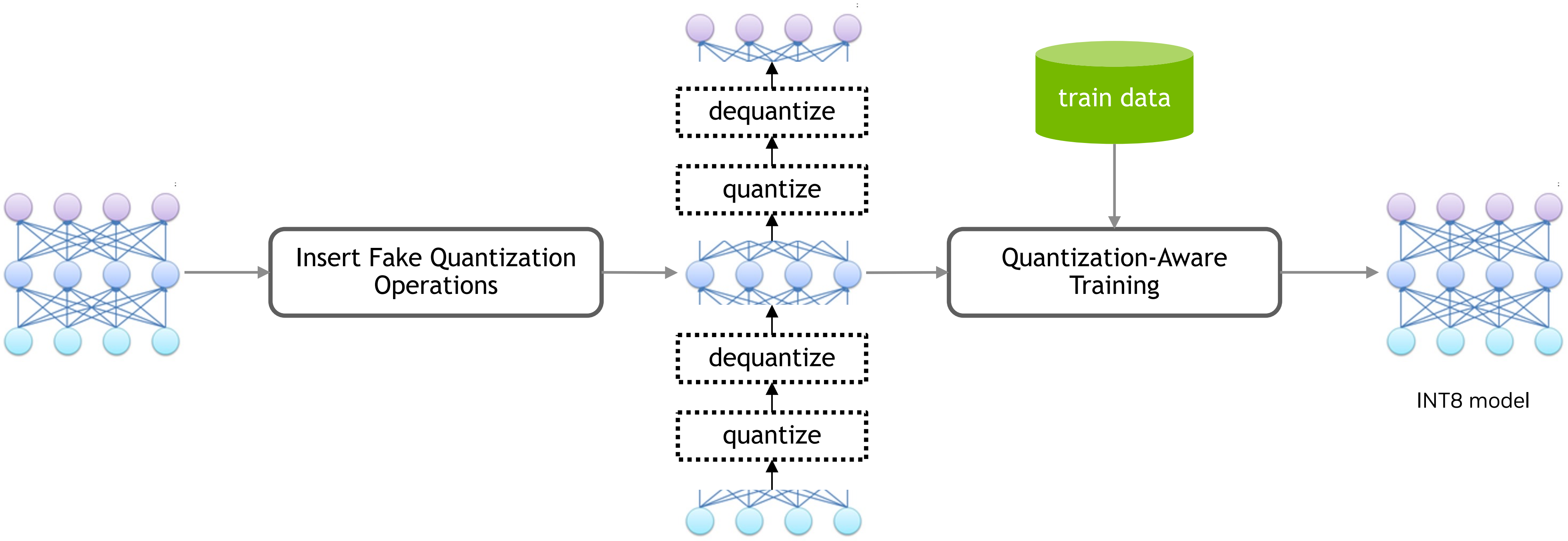
- **Symmetric INT8 quantization**: $\mathcal{C} = \{0, \pm 1, \pm 2, \ldots, \pm 127\}$
- Powers-of-two quantization: $\mathcal{C} = \{0, \pm 1, \pm 2^2, \pm 2^3, \ldots, 2^s\}$
- Log-scale codebooks: $\mathcal{C} = \{0, \pm \log 2, \pm \log 3, \pm \log 4, \ldots, \pm \log s\}$
- and many others

How to quantize a model?

Post-Training Quantization (PTQ)



Quantization-Aware-Training (QAT)



Model Quantization

Take Home Message

Post Training Quantization

- simple and fast
- no training is needed
- accuracy drop

Quantization Aware Training

- involves training pipeline
- produces higher accuracy

Pruning

- Pruning Category
- How, What, When to prune
- Application to AV

Neural Network Pruning

Pruning categorization

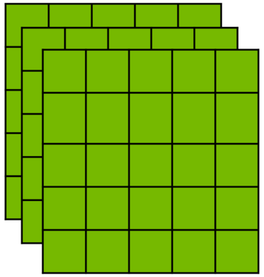
Overparameterization leads to better performance and parameter redundancy

Fine-grained (parameter) pruning

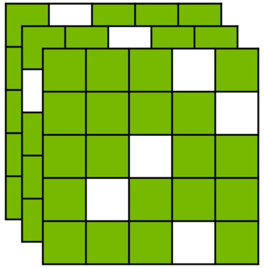
- Removes parameters from the model
- Requires specialized hardware

Coarse-grained (channel) pruning

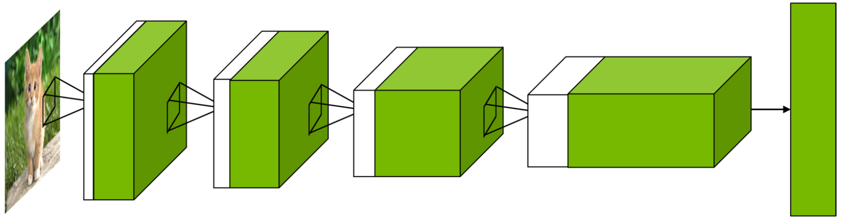
- Removes feature maps from the model
- Common GPUs can take advantage of acceleration



Dense Network



Fine Pruning



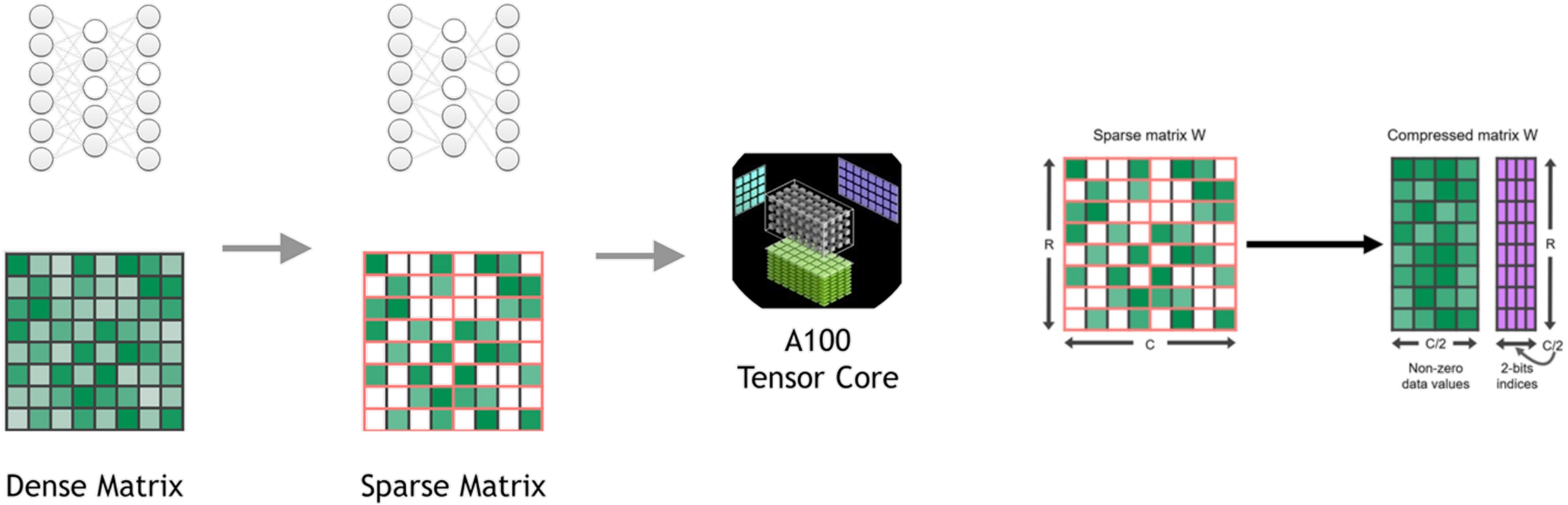
Coarse Pruning

Source: Shen et al [HALP: Hardware-aware latency pruning NeurIPS 2022.]

Fine-grained Pruning

Ampere Sparsity on Nvidia GPUs

NVIDIA Ampere architecture Sparse Tensor Cores accelerate 2:4 fine-grained sparsity.



Source: https://developer.nvidia.com/blog/accelerating-inference-with-sparsity-using-ampere-and-tensorrt/

Coarse-grained Pruning

Speedup on off-the-shelf GPUs

Element-wise operations of multiple inputs

- Layers connected to the element-wise operations need to be aligned in pruning
- Group channels with same channel index to enforce pruning together

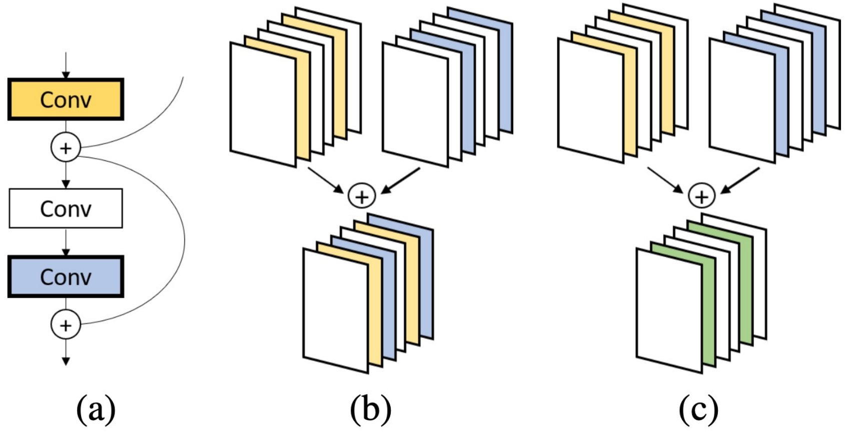


Figure 1. Pruning residual layers (a) is a challenging task. b) Applying sparsity constraints individually to each layer leads to different sparsity structures and therefore more channels need to be retained (colored blocks). c) Aligned sparsity between layers tied by skip connections would help on effective pruning. In this case, only two channels would remain after the pruning operation.

Source: [*Guo, et.al.*], Variance-Aware Cross-Layer Regularization for Pruning. CVPR W 2019

Pruning Methods

Regularization-based

- Encourage sparsity during training

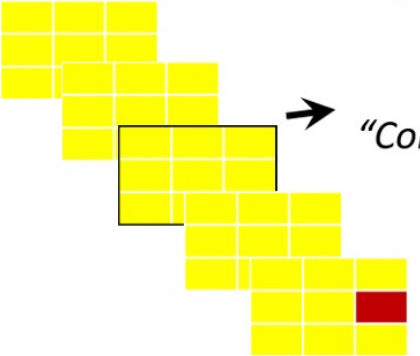
$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(\mathbf{x}_i, \Theta)) + \boxed{r(\Theta)}$$

- Compression-aware training
- Group Lasso penalty

Saliency-based

- Select sub-network via saliency
- Saliency / importance score depends on the model weights and loss

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(\mathbf{x}_i, \Theta)) + \frac{\rho}{2}||\Theta||_2^2$$



Neuron
"Convolutional kernel"

Convolutional layer
5x1x3x3

Considers each parameter independently

Weight Decay
(prevent weights with large values)

Source: [*Alvarez and Salzmann*], [Learning the number of neurons in neural nets,] NeurIPS 2016

[*Alvazed and Salzmann*], [Compression-aware training of DNN]. NeurIPS 2017

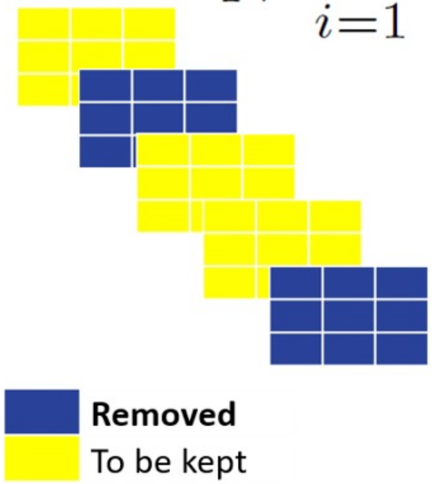
Regularization-Based Pruning

Compression-Aware Training

Regularization-Based Pruning

Compression-Aware Training

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(\mathbf{x}_i, \Theta)) + \frac{\rho}{2} ||\Theta||_2^2 + r(\Theta),$$



$$r(\Theta) = \sum_{l=1}^{L} \lambda_l \sqrt{P_l} \sum_{n=1}^{N_l} ||\theta_l^n||_2$$

Size of the group

Source: [*Alvarez and Salzmann*], Learning the number of neurons in neural nets, NeurIPS 2016

[*Alvazed and Salzmann*], Compression-aware training of DNN. NeurIPS 2017

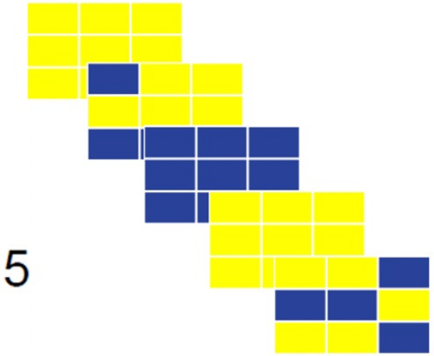
$$\min_{\Theta} \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, f(\mathbf{x}_i, \Theta)) + \frac{\rho}{2}||\Theta||_2^2 + r(\Theta) \ ,$$

$$r(\Theta) = \sum_{l=1}^{L} \left((1-\alpha)\lambda_l \sqrt{P_l} \sum_{n=1}^{N_l} ||\theta_l^n||_2 + \alpha\lambda_l ||\theta_l||_1 \right)$$

- $\alpha = 0$
- $\alpha = 0.5$

Removed

To be kept



Source: [*Alvarez and Salzmann*], Learning the number of neurons in neural nets, NeurIPS 2016

[*Alvazed and Salzmann*], Compression-aware training of DNN. NeurIPS 2017

Regularization-Based Pruning

Compression-Aware Training

Regularization-Based Pruning

Some other approaches

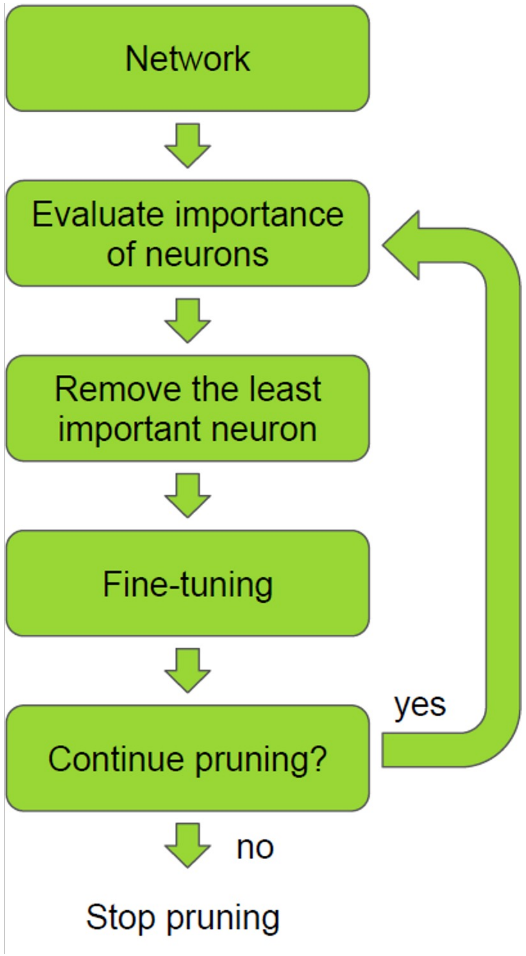
- Low-rank regularization
- L0 regularization
- Dropout learning
- Budget-aware regularization
- . . .

Saliency-Based Pruning

Sub-Network Selection

1. Estimate importance of neurons (units / channels)
2. Rank neurons
3. Remove least K important units
4. Fine tune the pruned network
5. Repeat step 1-4

- Iterative pruning typically leads to better accuracy than single-shot pruning.

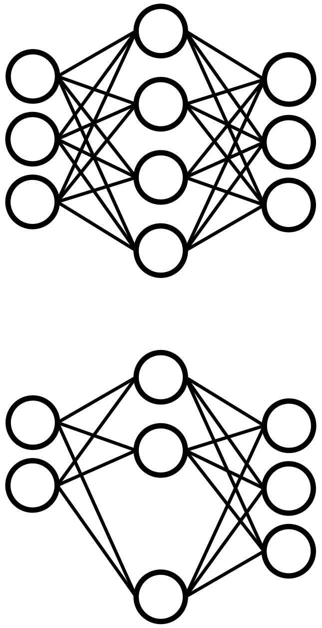


What to Prune?

Pruning Criterion

Prune weights / neurons that induce the least output perturbation.

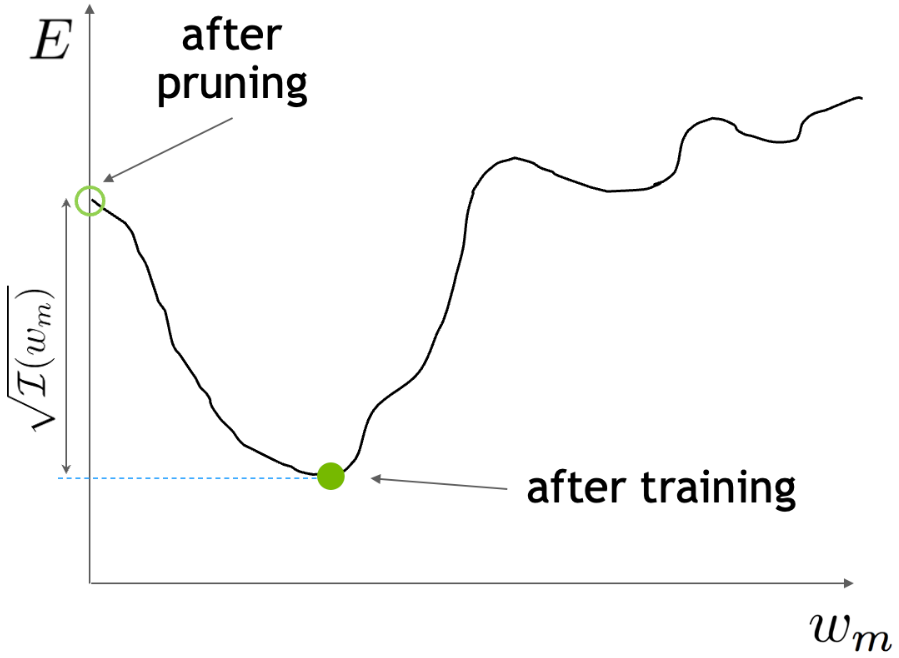
- Magnitude-based
  - Learning both weights and connections for efficient neural networks
  - Lottery ticket hypothesis
- Gradient-based
  - Optimal Brain Damage, Optimal Brain Surgeon
  - Skeletonization
  - SNIP
  - Taylor Importance



$$E(\mathcal{D}, \mathbf{W})$$

$$\Updownarrow \Delta E$$

$$E(\mathcal{D}, \mathbf{W} | w_m = 0)$$



$$\mathcal{I}(w_m) = \Big(E(\mathcal{D}, \mathbf{W}) - E(\mathcal{D}, \mathbf{W} | w_m = 0)\Big)^2$$

Taylor expansion:

$$E(\mathcal{D}, \mathbf{W} | w_m = 0) = E(\mathcal{D}, \mathbf{W}) - g_m w_m + \frac{1}{2} w_m \mathbf{H}_m \mathbf{W} + R_3$$

First order Taylor expansion in cost by removing a weight

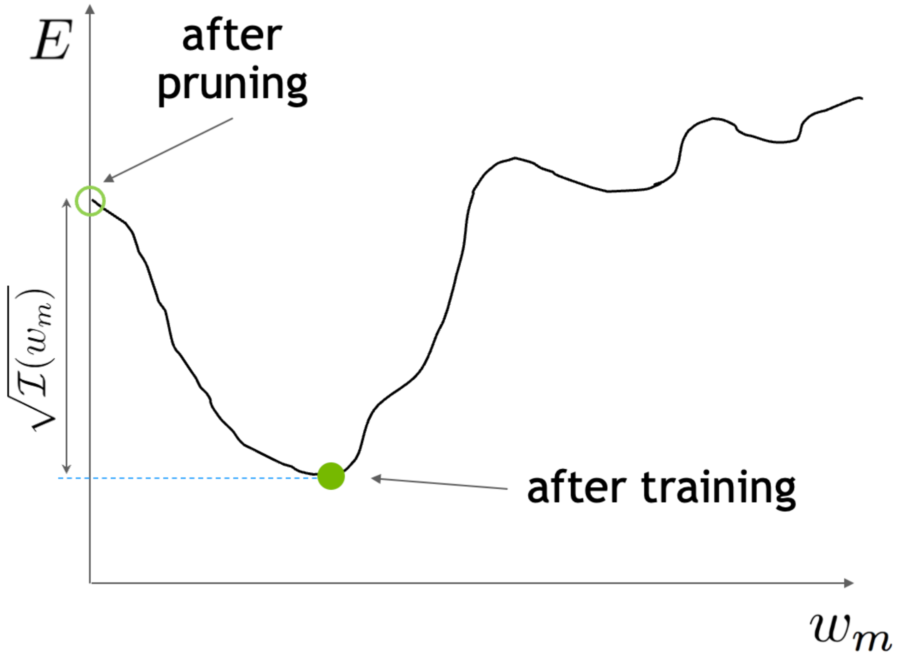
$$\mathcal{I}^{(1)}_m(\mathbf{W}) = \Big(g_m w_m\Big)^2$$

Source: [*Molchanov, et.al.*], Pruning Convolution Neural Networks for Resource Efficient Inference. ICLR 2017

[*Molchanov, et.al.*], Importance Estimation for Neural Network Pruning. CVPR 2019

Approximate the Importance

Via Taylor expansion



$$\mathcal{I}(w_m) = \Big(E(\mathcal{D}, \mathbf{W}) - E(\mathcal{D}, \mathbf{W}|w_m = 0)\Big)^2$$

Taylor expansion:

$$E(\mathcal{D}, \mathbf{W}|w_m = 0) = E(\mathcal{D}, \mathbf{W}) - g_m w_m + \frac{1}{2} w_m \mathbf{H}_m \mathbf{W} + R_3$$

Second order Taylor expansion will lead to Optimal Brain Surgery (1980s):

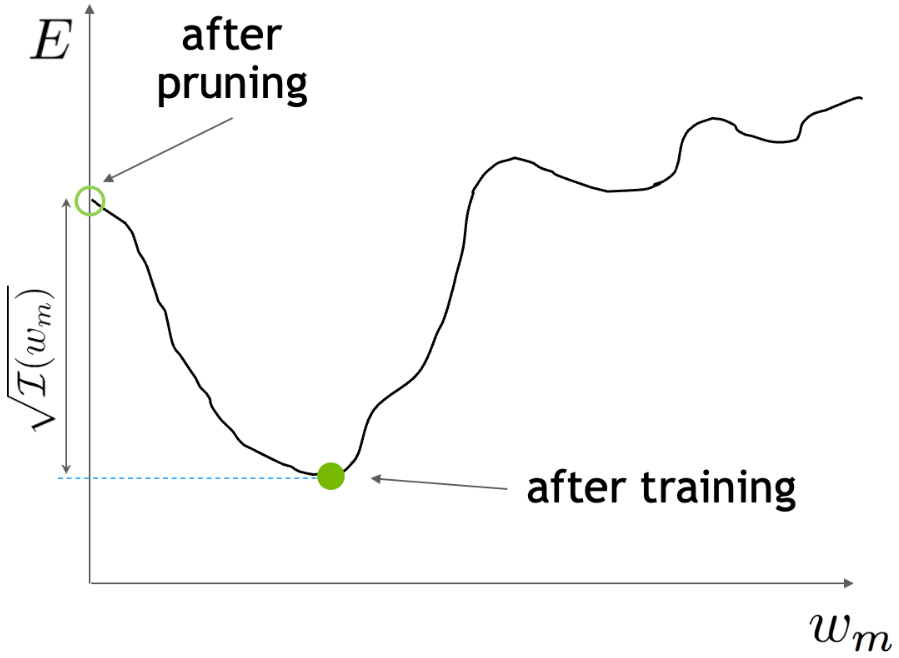
$$\boldsymbol{\delta_m} = -\frac{w_m}{[\mathbf{H}^{-1}]_{mm}} \cdot \mathbf{H}^{-1}_{:,m} \text{ incurring error } \varepsilon_m = \frac{w_m^2}{2[\mathbf{H}^{-1}]_{mm}}.$$

Source: [*Molchanov, et.al.*], Pruning Convolution Neural Networks for Resource Efficient Inference. ICLR 2017

[*Molchanov, et.al.*], Importance Estimation for Neural Network Pruning. CVPR 2019

Approximate the Importance

Via Taylor expansion



The joint importance of a structural set of parameters can be approximated by

$$\mathcal{I}_{\mathcal{S}}^{(1)}(\mathbf{W}) \triangleq \left(\sum_{s\in\mathcal{S}} g_s w_s\right)^2$$

With the existence of Batch Normalization layer, the importance can be captured on BN layer

$$\mathcal{I}_m^{(1)} = \left(\gamma_m \frac{\partial E}{\partial \gamma_m} + \beta_m \frac{\partial E}{\partial \beta_m}\right)^2$$

Source: [*Molchanov, et.al.*], Pruning Convolution Neural Networks for Resource Efficient Inference. ICLR 2017

[*Molchanov, et.al.*], Importance Estimation for Neural Network Pruning. CVPR 2019

Approximate the Importance

Via Taylor expansion

Approximate the Importance

Via Taylor expansion

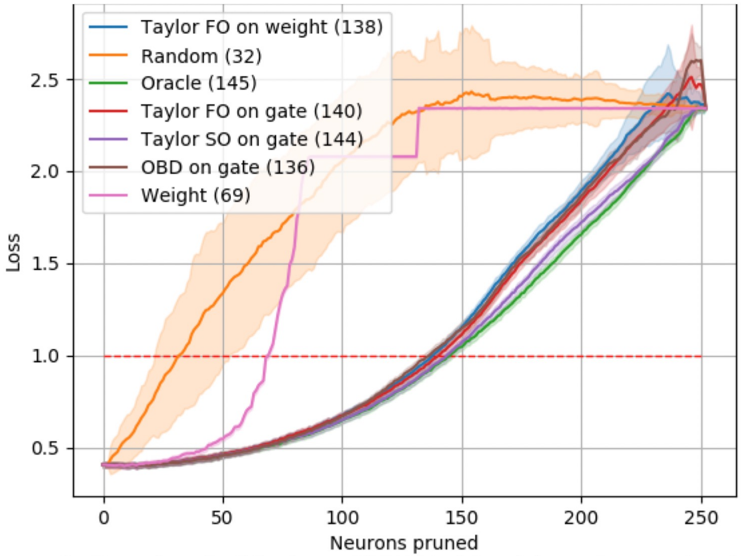


Figure 3: Pruning LeNet3 on CIFAR-10 with various criteria. Network remains fixed and is not fine-tuned. Results are averaged over 50 seeds with mean and standard deviation. The number of pruned neurons when the loss reaches 1.0 is shown in parentheses.

- Gradient-based techniques are superior to weight magnitude-based pruning.
- Taylor-importance-based pruning can reach Oracle pruning performance.
- Second order pruning (Hessian) is not necessary for importance estimation.

Source: [*Molchanov, et.al.*], Importance Estimation for Neural Network Pruning. CVPR 2019

Compact Models

The goal of pruning

storage

memory

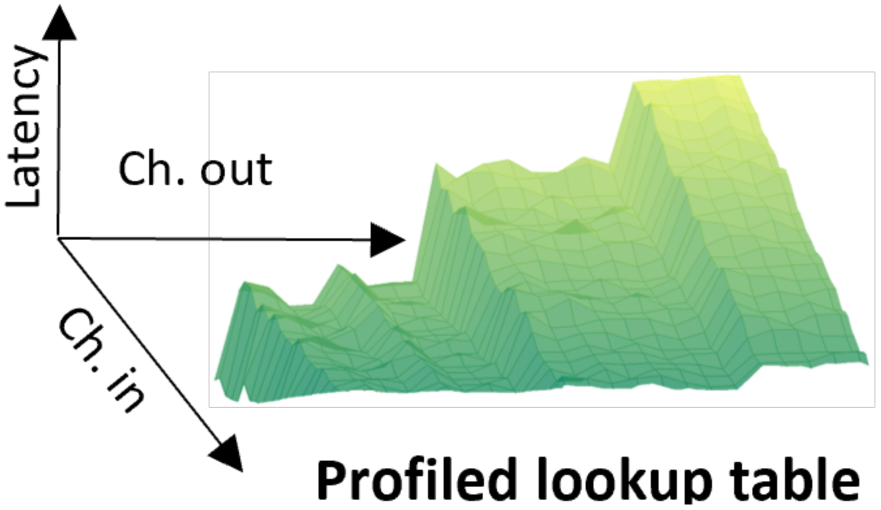
compute

latency

Neural Network Pruning for Latency Reduction

Maximize compute capacity and reduce inference latency.

- FLOPs and parameters are not linearly correlated with latency.
- Algorithms typically target generic platforms rather than specific hardware where models will be deployed.

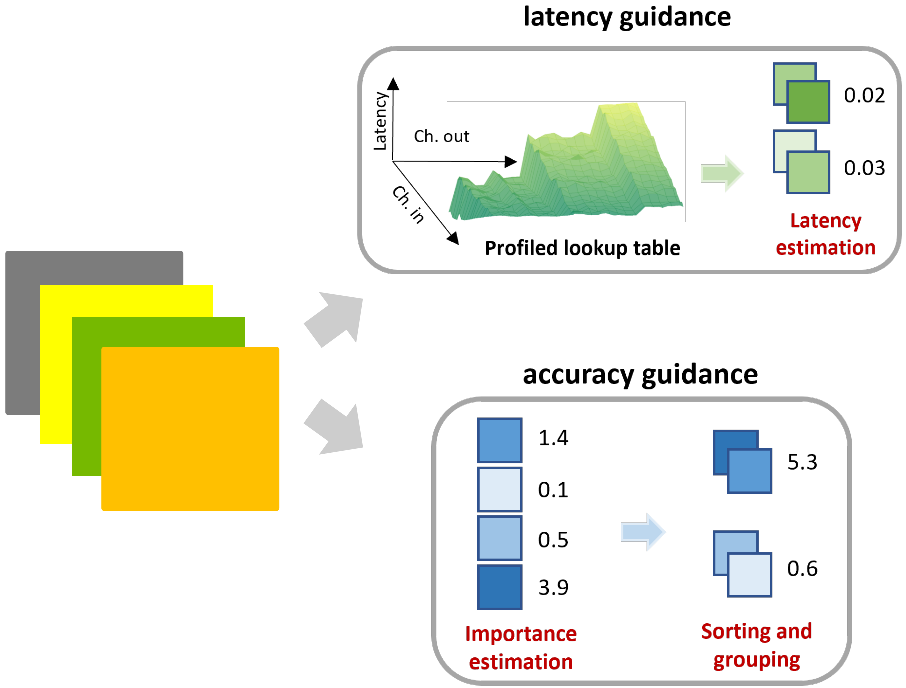


Source: [*Shen et al.*] Structural Pruning via Latency-Saliency Knapsack. NeurIPS 2022

$$\arg\min_{\hat{\mathbf{W}}} \mathcal{L}(\hat{\mathbf{W}}, \mathcal{D}) \quad \text{s.t.} \quad \Phi\left(f(\hat{\mathbf{W}}, x_i)\right) \le C$$

$$\arg\max_{p_1,\cdots,p_l} \sum_{l=1}^{L} I_l(p_l), \text{ s.t. } \sum_{l=1}^{L} T_l(p_{l-1}, p_l) \le C, \ \forall l \ \ 0 \le p_l \le N_l$$

$$\boxed{\max \sum_{l=1}^{L}\sum_{j=1}^{p_l} \mathcal{I}_l^j, \quad \text{s.t.} \quad \sum_{l=1}^{L}\sum_{j=1}^{p_l} c_l^j \le C,} \quad 0\le p_l \le N_l, \quad \mathcal{I}_l^1 \ge \mathcal{I}_l^2 \ge \dots \mathcal{I}_l^{N_l}$$



Source: [*Shen et al.*] Structural Pruning via Latency-Saliency Knapsack. NeurIPS 2022

HALP: Hardware-Aware Latency Pruning

Latency Constrained Pruning

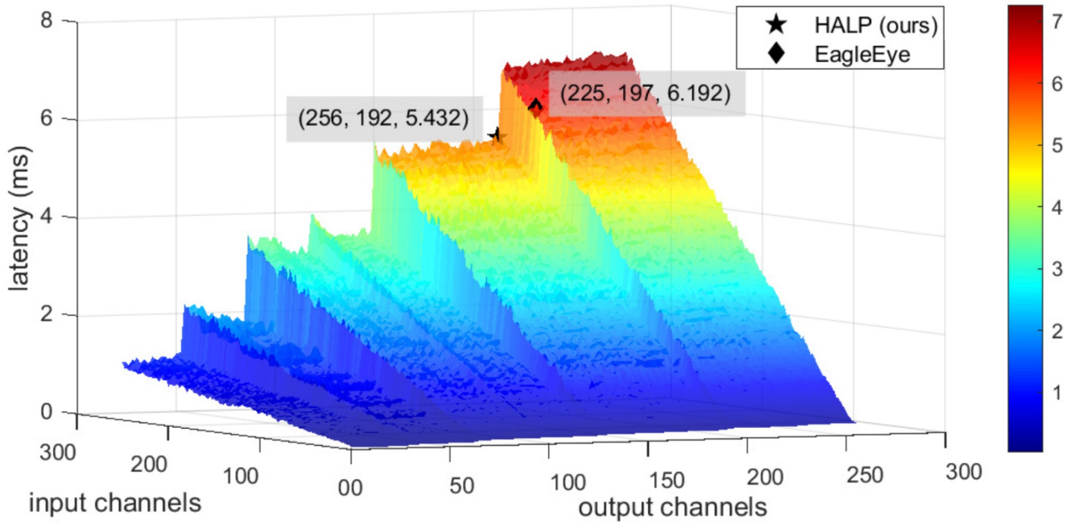
Layer Latency after Pruning

Latency Comparison with / without Latency-Aware

	Latency-Aware	Generic Pruning
C_in	256	225
C_out	192	197
# param	442K	242K
# FLOP	86.7M	78.2M
Latency (ms)	**5.43**	6.19

* Conv(kernel=3) with 28x28 input

While optimizing for latency, the layer gets more parameters remaining but lower latency.

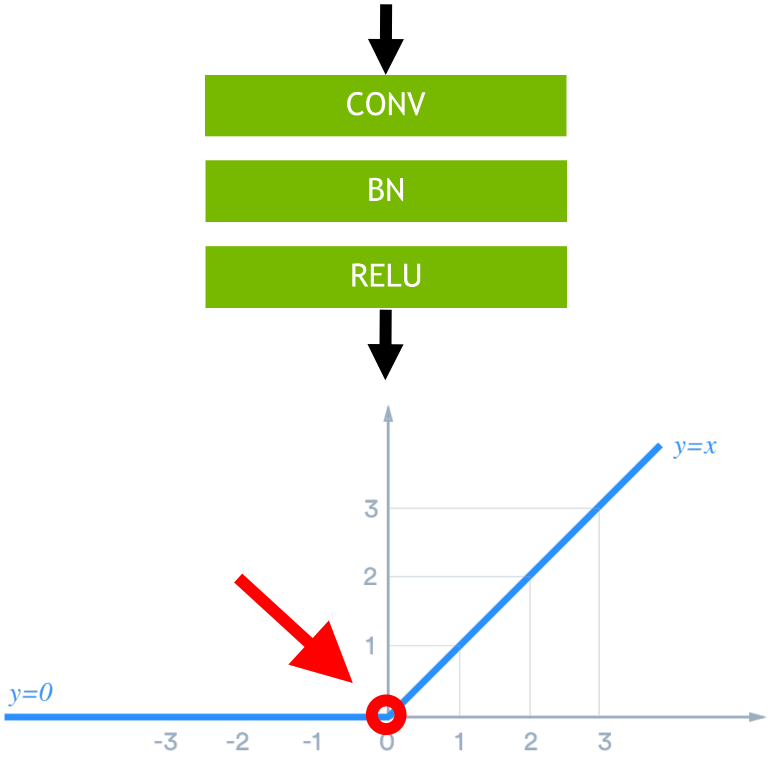


Source: [*Shen et al.*] Structural Pruning via Latency-Saliency Knapsack. NeurIPS 2022

Aggressive pruning tends to lead large accuracy drop.

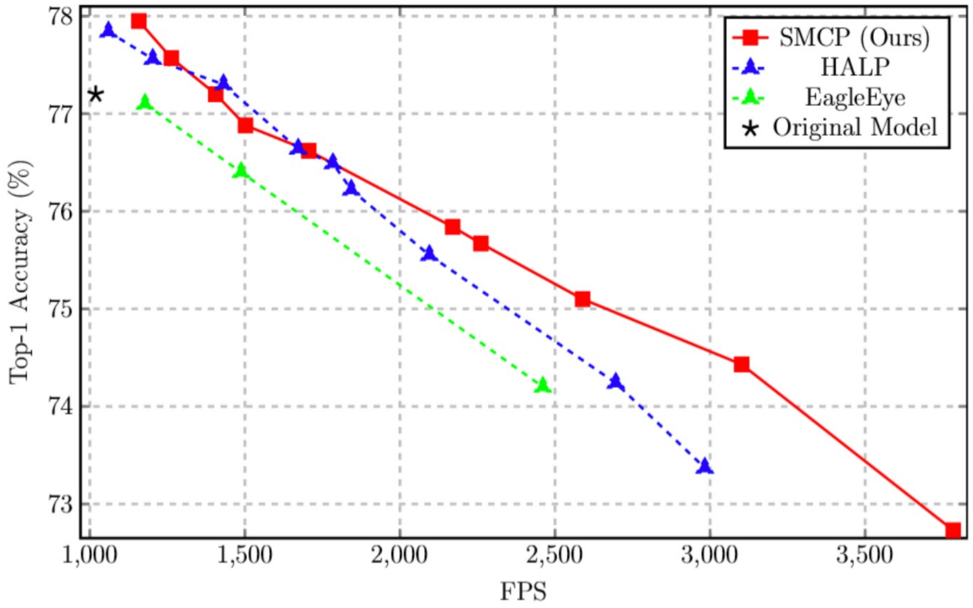
Soft pruning helps preserve the capacity to explore the optimality.

- Allow channels to be restored or grow back to undo poor early decisions.
- Apply mask to weights instead of permanently zeroing weights.
- Switching to input channel pruning makes “pruned” neuron weights receive gradients continuously.



Source: Humble et al. Soft Masking for Cost-constraint channel pruning. ECCV 2022

Soft-Masking Channel Pruning



- Better latency-accuracy trade-off when targeting specific hardware and latency constraints
- Algorithmic improvements yield better performance when targeting large pruning ratios

Source: Humble et al. Soft Masking for Cost-constraint channel pruning. ECCV 2022

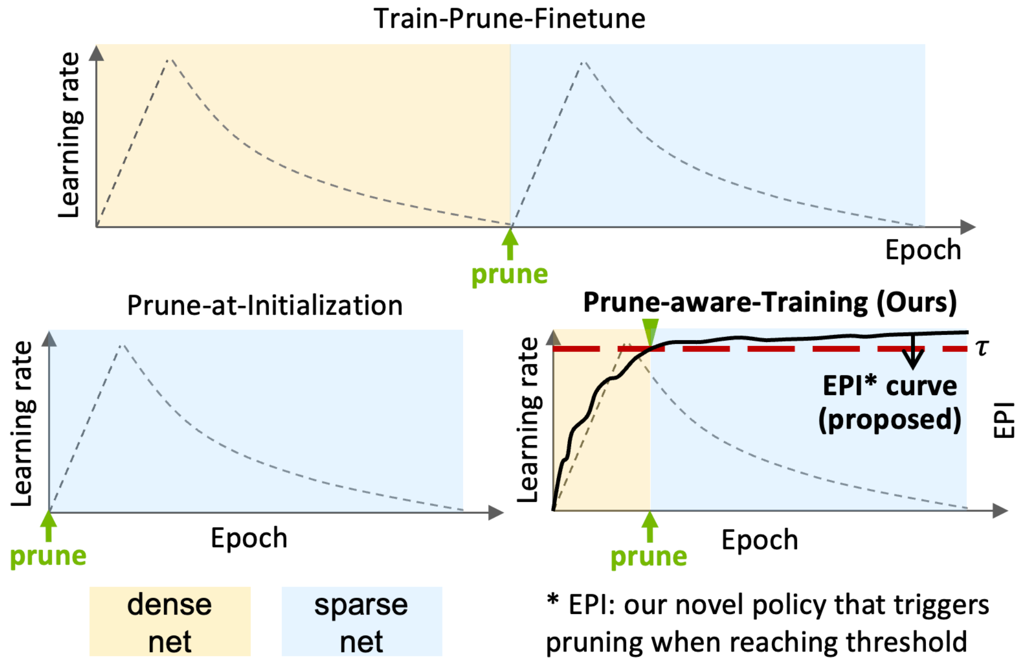
Latency-Aware Pruning Results

Can training efficiency also benefit from pruning?

When to Prune?

Training Efficiency

- Train-Prune-Finetune
  - Dense training is needed thus almost doubling the training cost.
  - Achieves high accuracy.
- Prune-at-Initialization
  - Requires least training cost.
  - Suffers noticeable accuracy loss.
- Prune-aware-Training
  - Push towards an early pruning.
  - Seeks a better trade-off between the accuracy and training efficiency

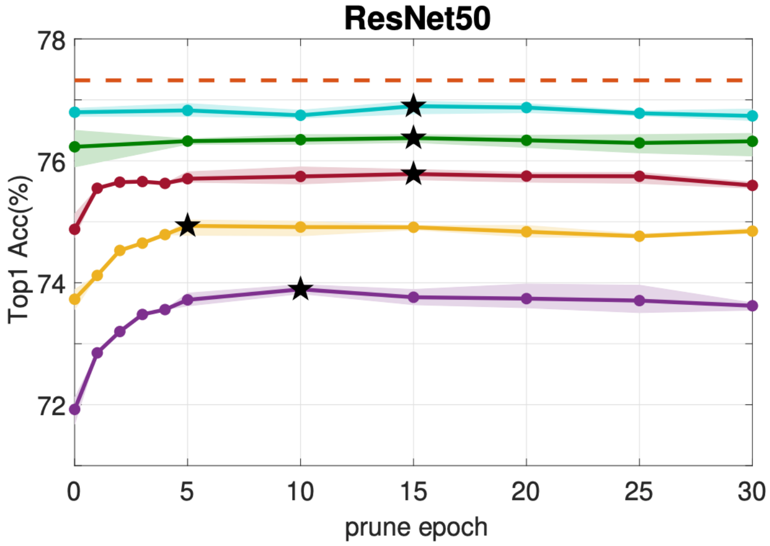


Source: [*Shen, Molchanov, Yin, Alvarez*], When to Prune? A Policy towards Early Structured Pruning. CVPR 2022

Towards Early Pruning for Sub-Network Selection

Early pruning indicator

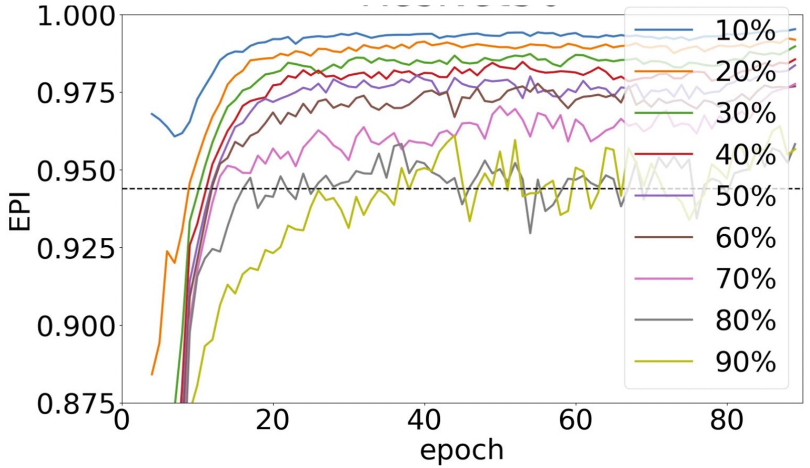
- Early stage of training imposes a rapid motion in learning, and shows to be critical to accuracy
- A stable dominant sub-network formed by the top-k most important neurons quickly emerges



Dominant sub-network stability measurement

$$\Psi(\mathcal{N}_1, \mathcal{N}_2) = 1 - \frac{1}{L}\sum_{l=1}^{L} \frac{\left|n_{(1,l)} - n_{(2,l)}\right|}{n_{(1,l)} + n_{(2,l)}}$$

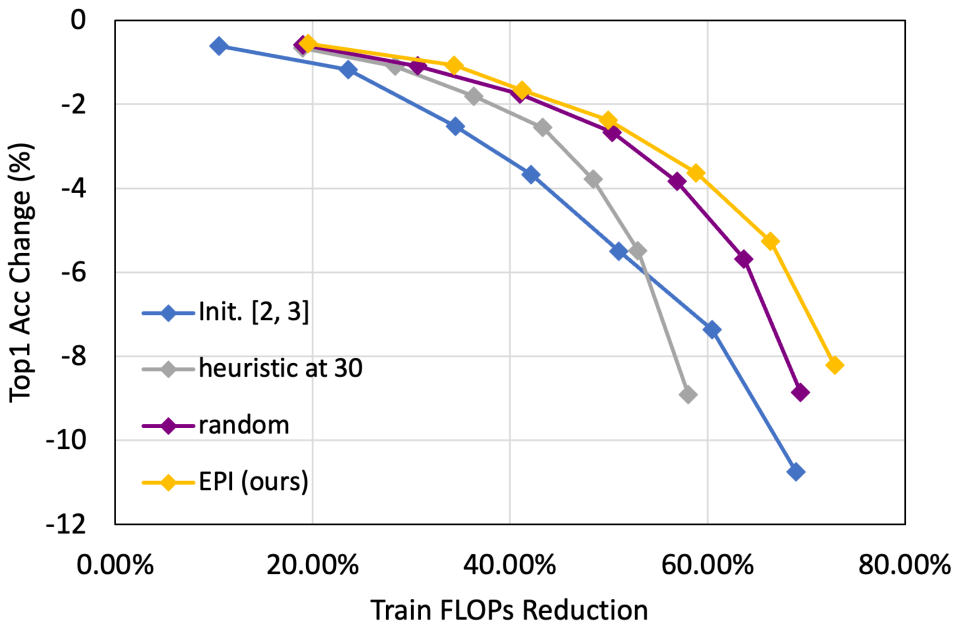
$$EPI_t = \frac{1}{r}\sum_{j=1}^{r} \Psi(\mathcal{N}_t, \mathcal{N}_{t-j})$$

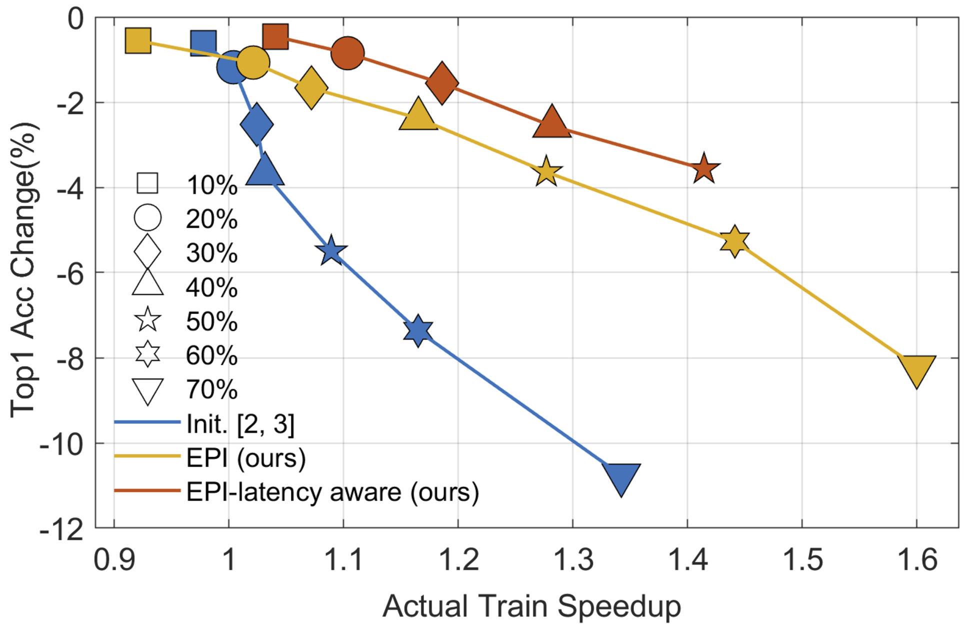


Source: [*Shen, Molchanov, Yin, Alvarez*], When to Prune? A Policy towards Early Structured Pruning. CVPR 2022

Training Efficiency

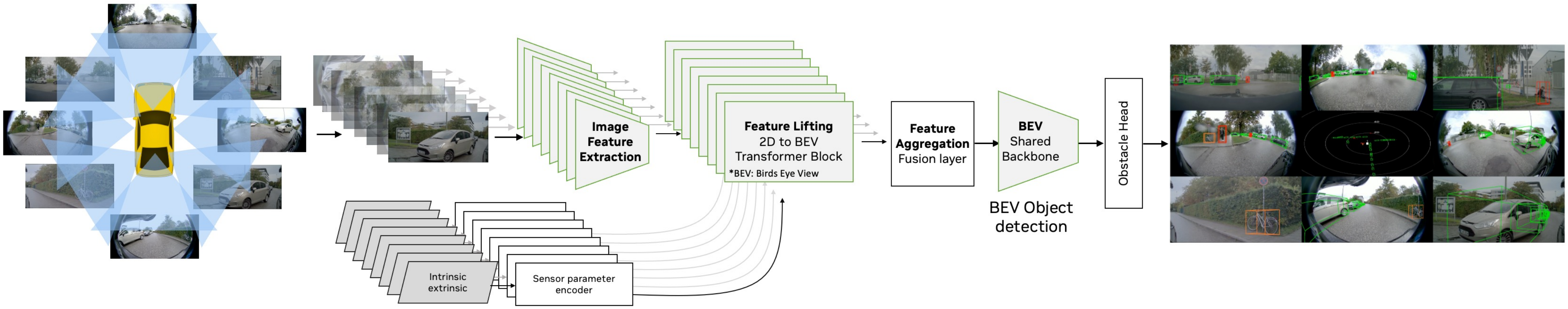
Early pruning indicator





Source: [*Shen, Molchanov, Yin, Alvarez*], When to Prune? A Policy towards Early Structured Pruning. CVPR 2022

Pruning for Detection



Source: [*Shen et al.*] [Hardware-Aware Latency Pruning for Real-Time 3D Object Detection](). IV 2023

Model Compression for Autonomous Vehicles

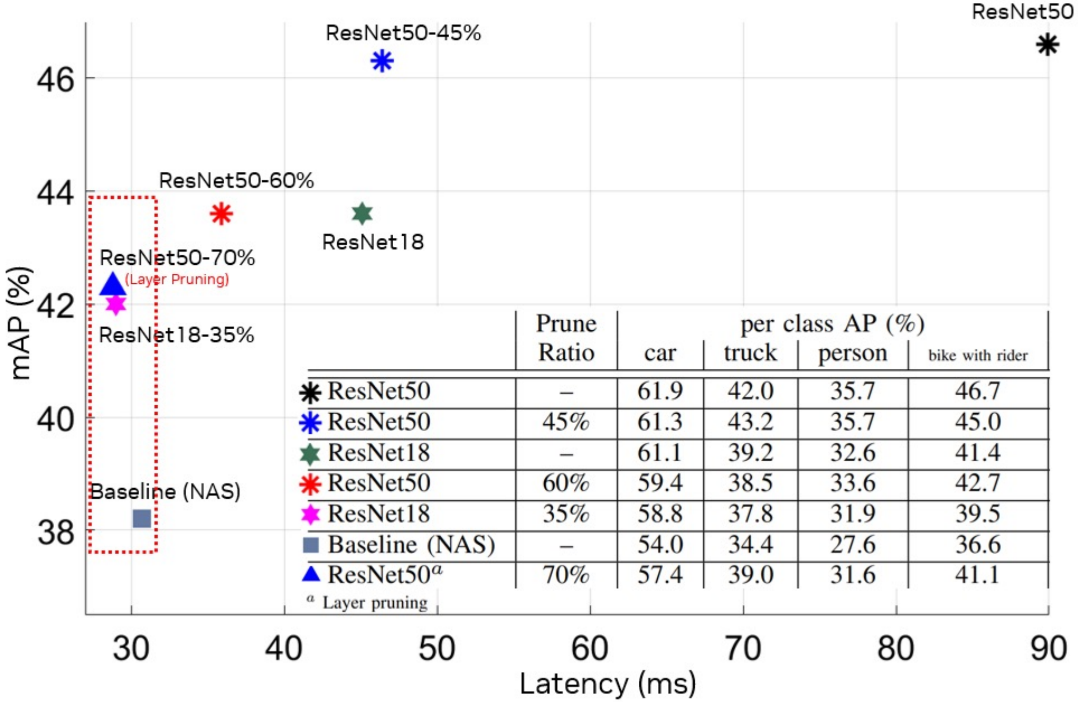
Train Large, then Compress

Model Compression for Autonomous Vehicles

Train Large, then Compress

Start on a larger model and then apply aggressive optimization

- Maximize the accuracy by training a larger model
- Compress the model aggressively to meet resource constraints
- With deep models, layer pruning should be considered to benefit latency



	Prune Ratio	car	per class AP (%) truck	person	bike with rider
ResNet50	–	61.9	42.0	35.7	46.7
ResNet50	45%	61.3	43.2	35.7	45.0
ResNet18	–	61.1	39.2	32.6	41.4
ResNet50	60%	59.4	38.5	33.6	42.7
ResNet18	35%	58.8	37.8	31.9	39.5
Baseline (NAS)	–	54.0	34.4	27.6	36.6
ResNet50[a]	70%	57.4	39.0	31.6	41.1

[a] Layer pruning

Source: [*Shen et al.*] Hardware-Aware Latency Pruning for Real-Time 3D Object Detection. IV 2023

Distant Objects

Far away Object

Close Objects

Train compact model

Train large model + Prune

Model Compression for Autonomous Vehicle

Train Large, then Compress

Full-Stack, GPU-Based Acceleration of Deep Learning. June 2023.

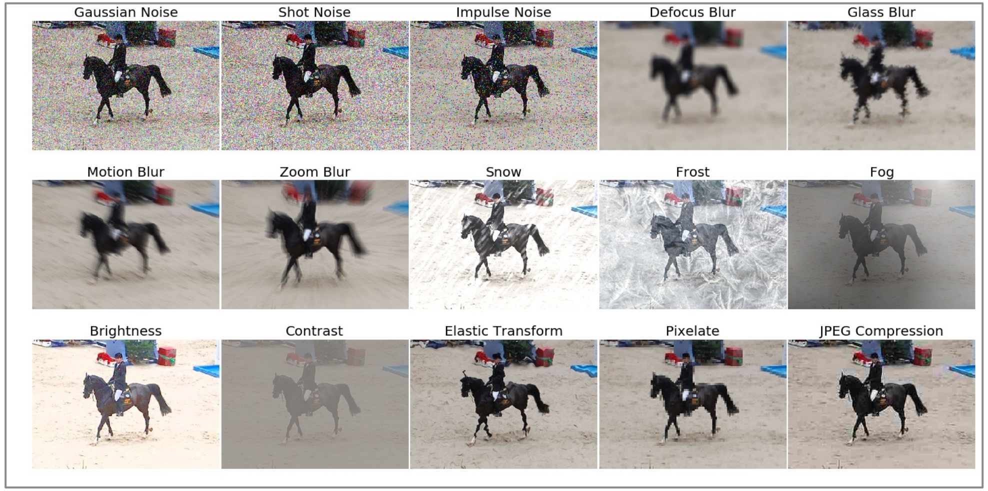
Robust Pruning

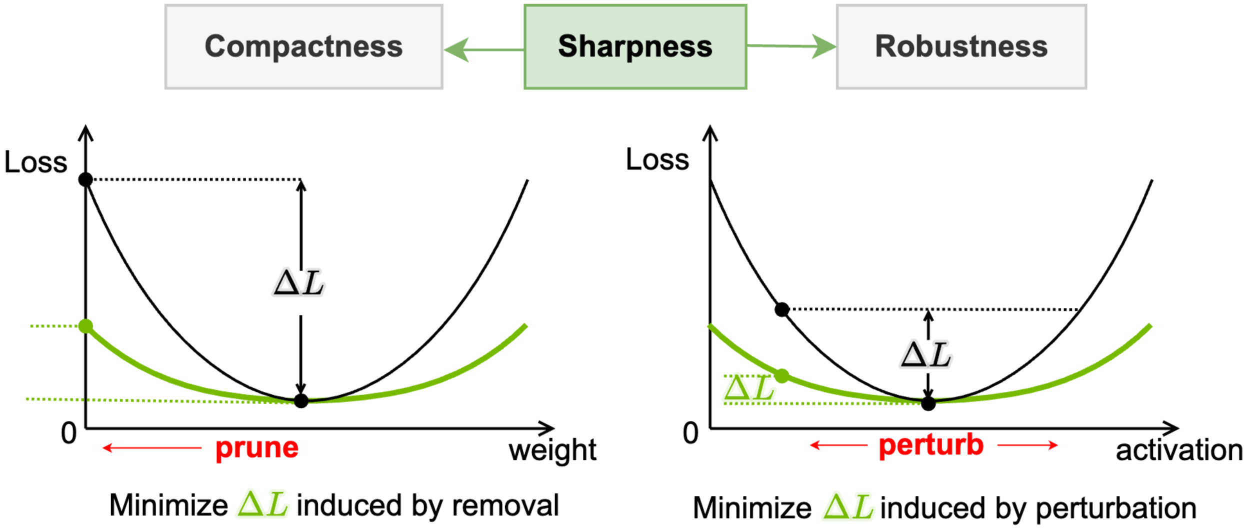
Robustness to corruptions and perturbations

Natural image corruption

- No corruptions are known during training or finetuning
- Can be seen as small deviations from the training distribution
- Pruning reduces out-of-distribution (OOD) generalization performance

How can we jointly address robustness and compression?





- Before pruning, encourage neurons to be pruned to lie within a flat minima
- After pruning, promote robustness to reduce the loss sensitivity to perturbation

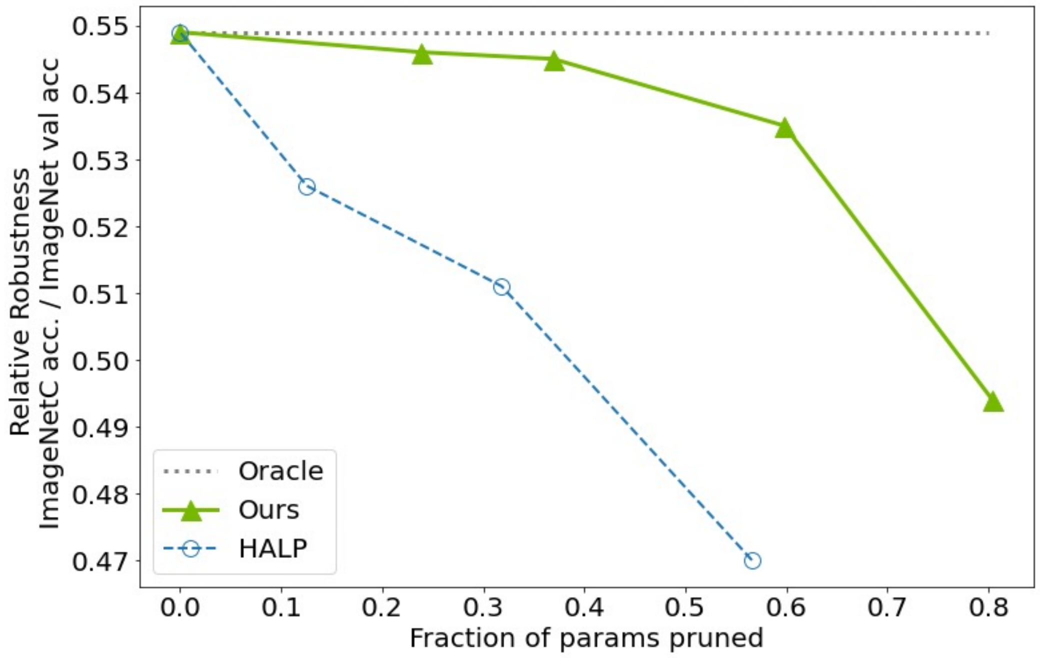
Bair et al. Robust Pruning Work in progress

Improve the Robustness

Encourage Sparsity and Robustness Simultaneously

Improve the Robustness

Encourage Sparsity and Robustness Simultaneously



Robustness of pruned models degrades drastically.

By encouraging robustness, we can reduce the performance degradation and improve robustness.

Bair et al. Robust Pruning Work in progress

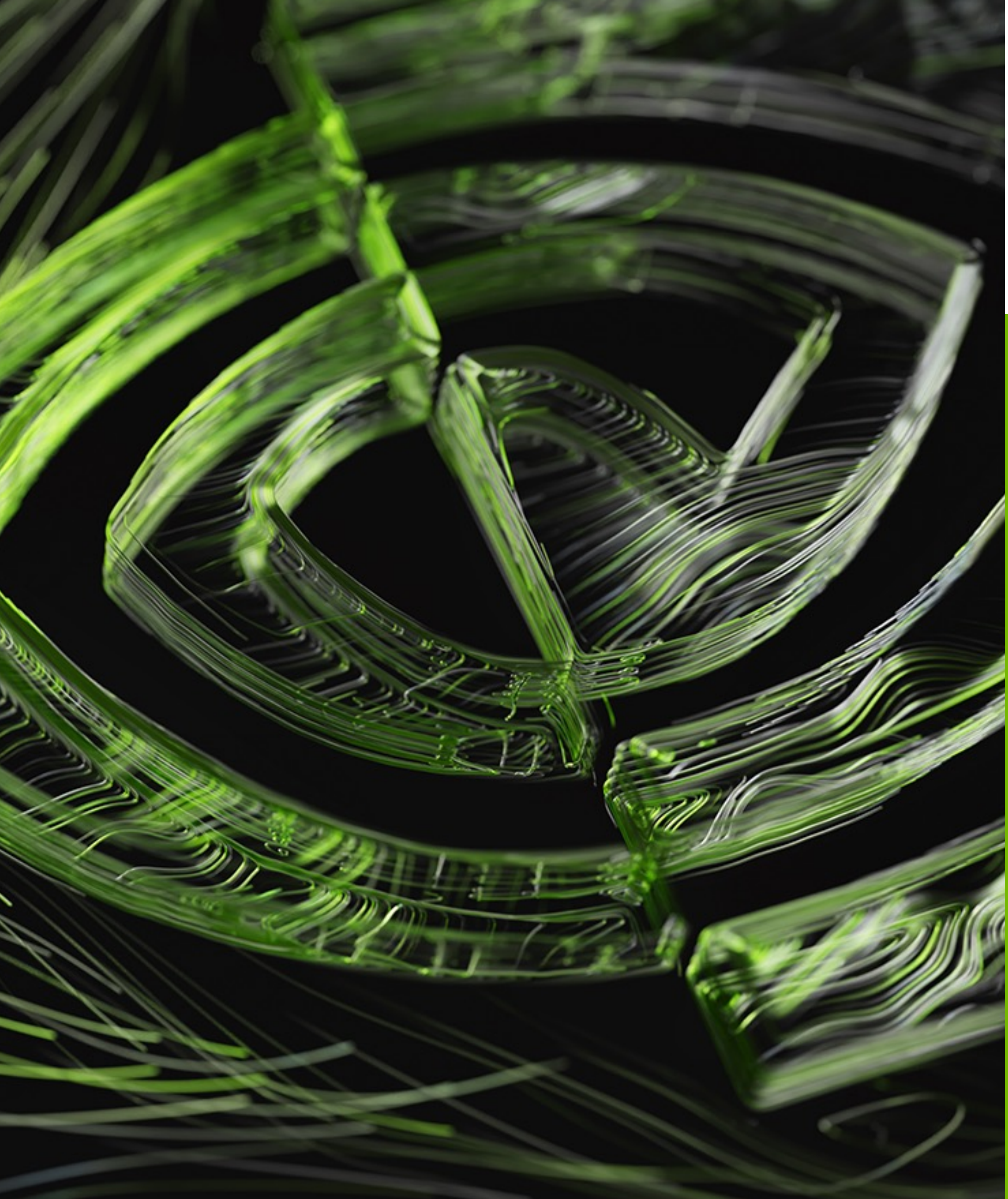
Tools for Network Quantization and Pruning

Toolkits

- Quantization
  - PyTorch Quantization
  - NVIDIA TF-QAT Toolkit

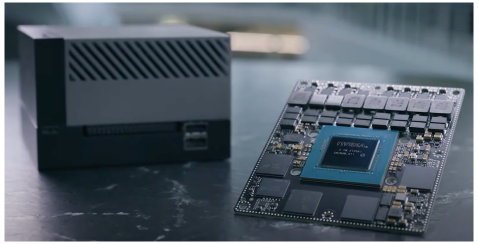
- Pruning
  - PyTorch Pruning
  - NVIDIA ASP (Automatic SParsity) for 2:4 ampere sparsity
  - Taylor Pruning
  - HALP, SMCP

DLA Optimization - Demo



DLA Optimization - Demo

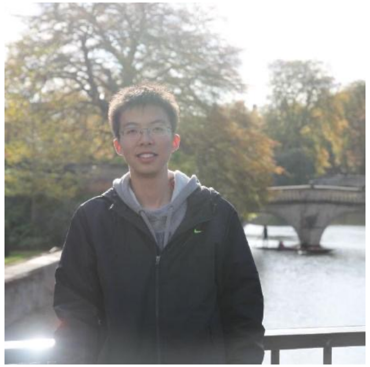
NVIDIA Jetson AGX Orin



- GPU + DLA

* DLA: fixed-function accelerator engine targeted for DL operations

Special thanks to Lei Mao for the demo!



Thanks!

Maying Shen
mshen@nvidia.com