Model Compression and Pruning Techniques

Fatemeh Rahimian

RISE Research Institutes of Sweden

fatemeh.rahimian@ri.se

Acknowledgement: Some slides are borrowed from the Song Han's course on TinyML.

Outline

Model compression: a brief introduction

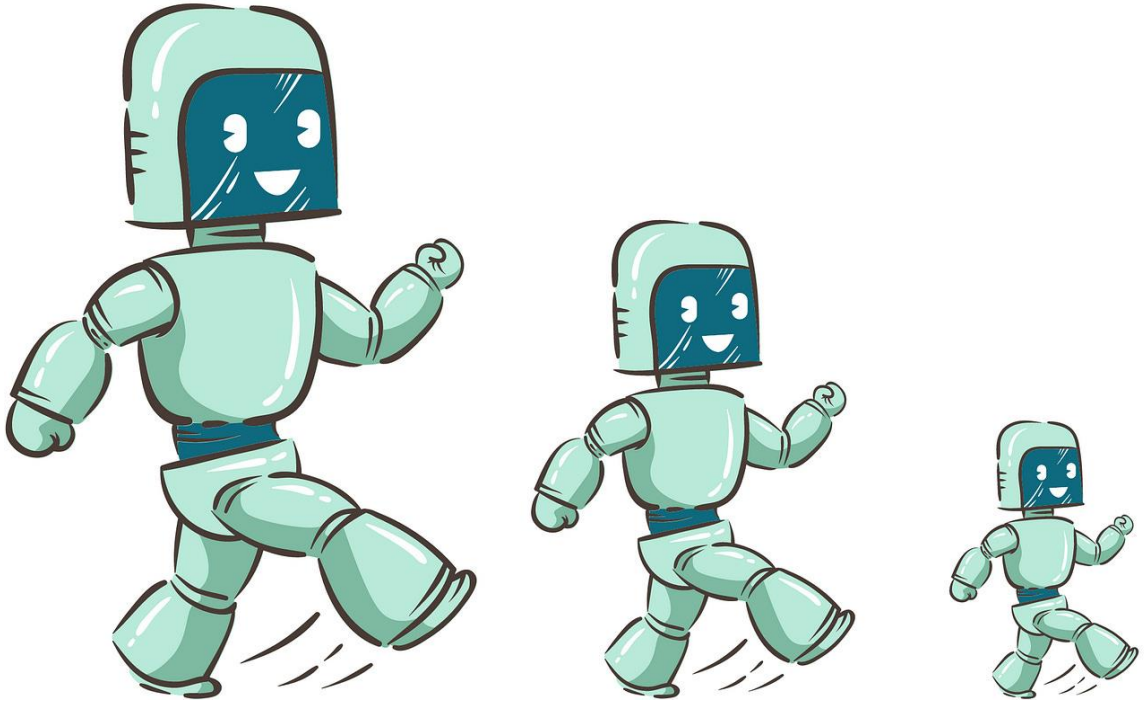
- Pruning
- Quantization
- Knowledge Distillation
- Low Rank Factorization

Pruning in more details

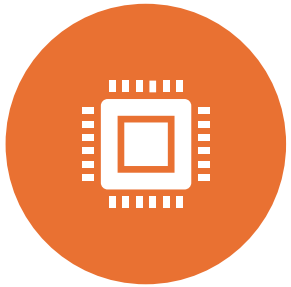
- Pruning criteria
- Pruning granularity
- Pruning ratio
- When to prune

What Is Model Compression?

- Reducing the size of a model
- without significantly compromising its predictive accuracy.

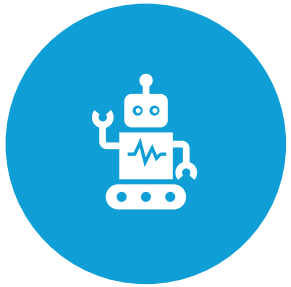


Why Model Compression?



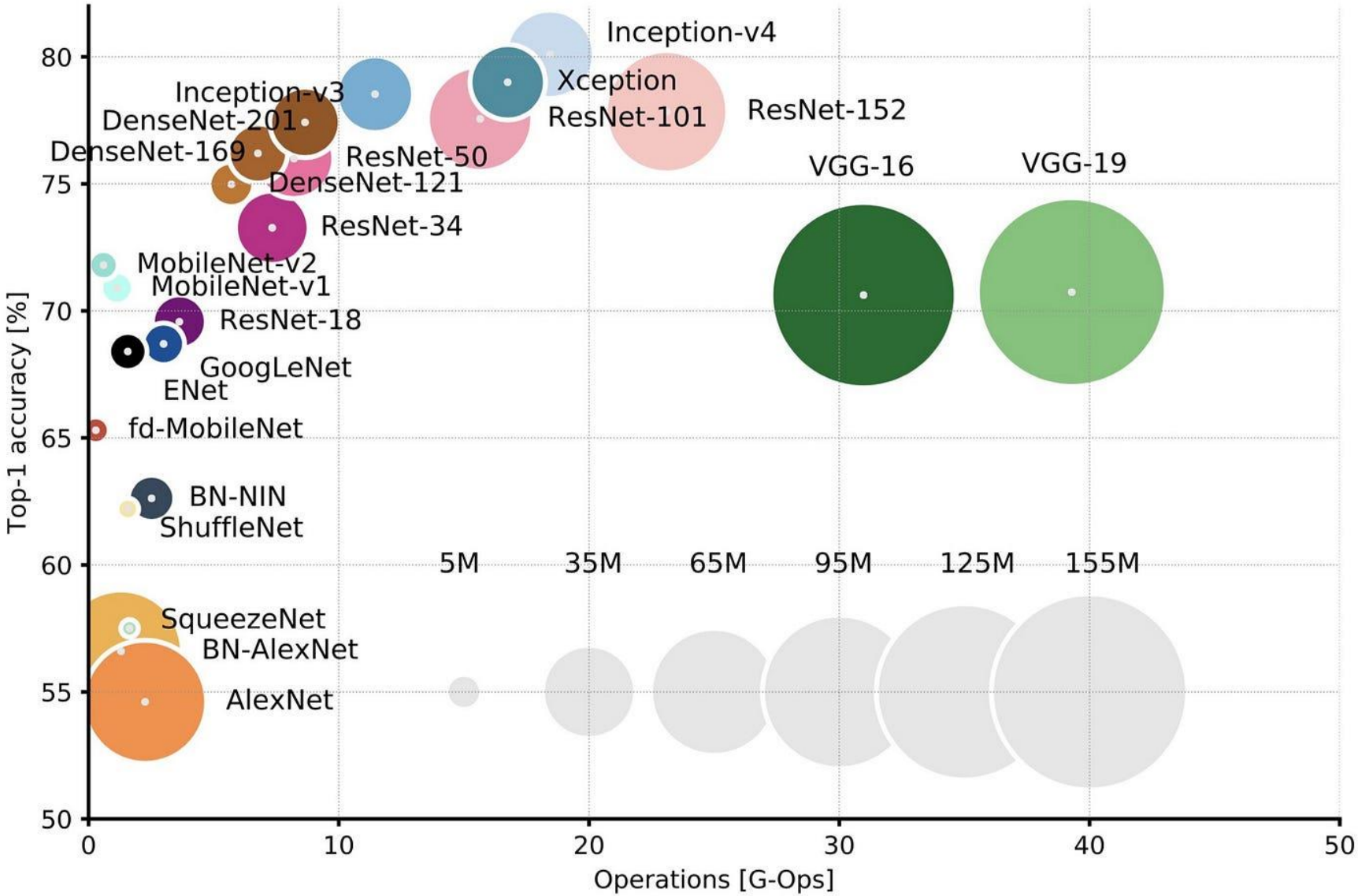
Deep learning models are compute- and memory- intensive.

Edge devices have limited power, memory, and compute.



Model compression enables AI on-device

Applications: AIoT, wearables, autonomous drones.

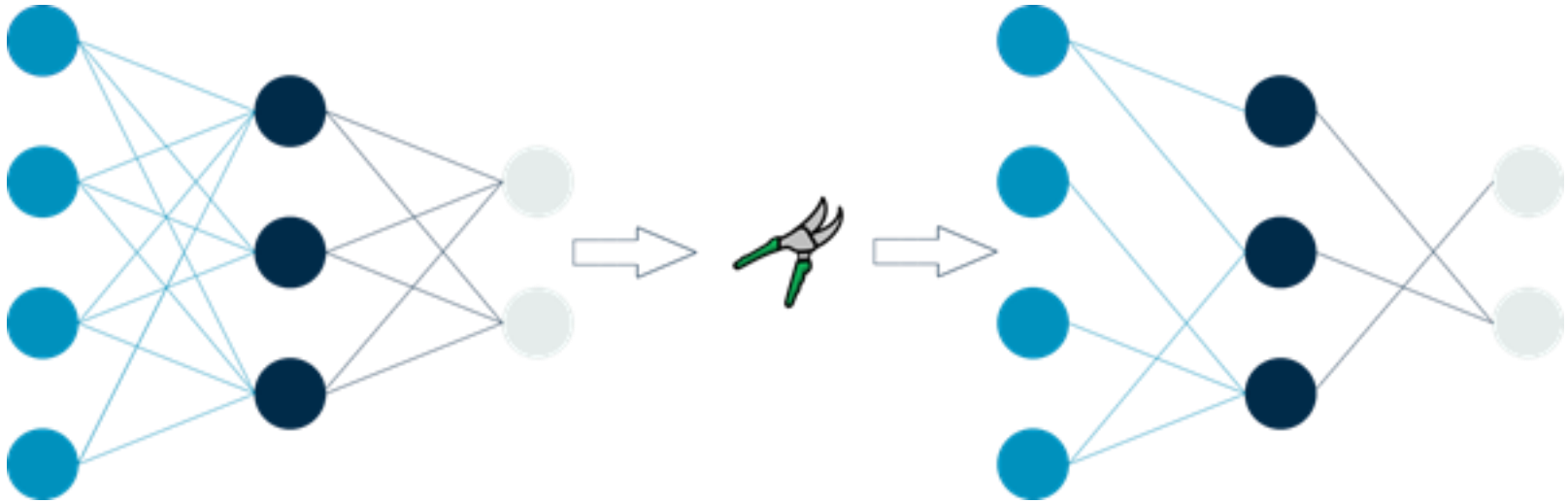

Inception-v4
Inception-v3
Xception
ResNet-101
ResNet-152
DenseNet-201
DenseNet-169
ResNet-50
DenseNet-121
VGG-16
VGG-19
ResNet-34
MobileNet-v2
MobileNet-v1
ResNet-18
GoogLeNet
ENet
fd-MobileNet
BN-NIN
ShuffleNet
5M
35M
65M
95M
125M
155M
SqueezeNet
BN-AlexNet
AlexNet
Top-1 accuracy [%]
80
75
70
65
60
55
50
0
10
20
30
40
50
Operations [G-Ops]

Model Compression Techniques

- Pruning
- Quantization
- Konwledge distillation
- Low rank factorization

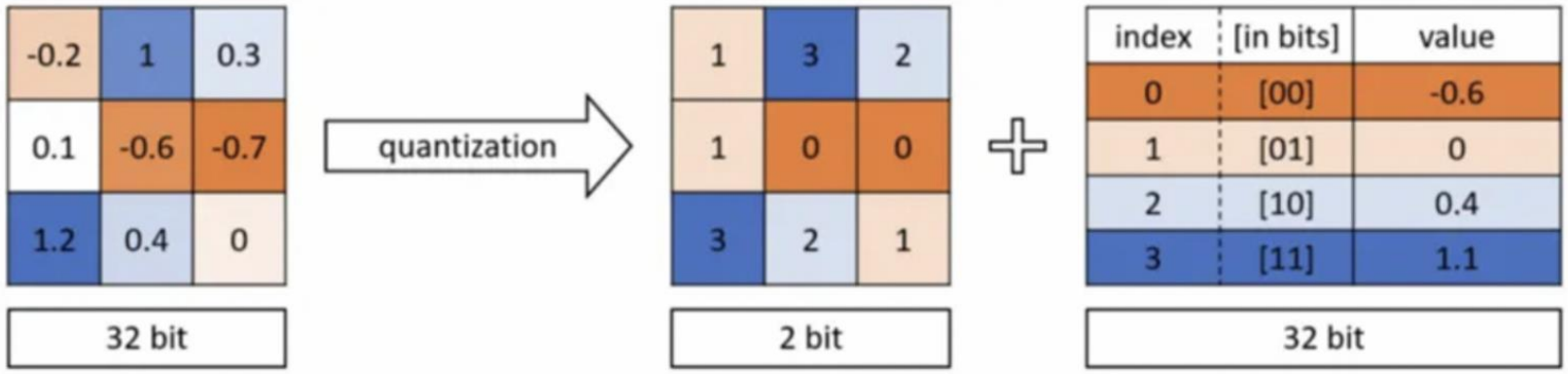
What Is Pruning?

- Systematically removing less significant parameters of a NN



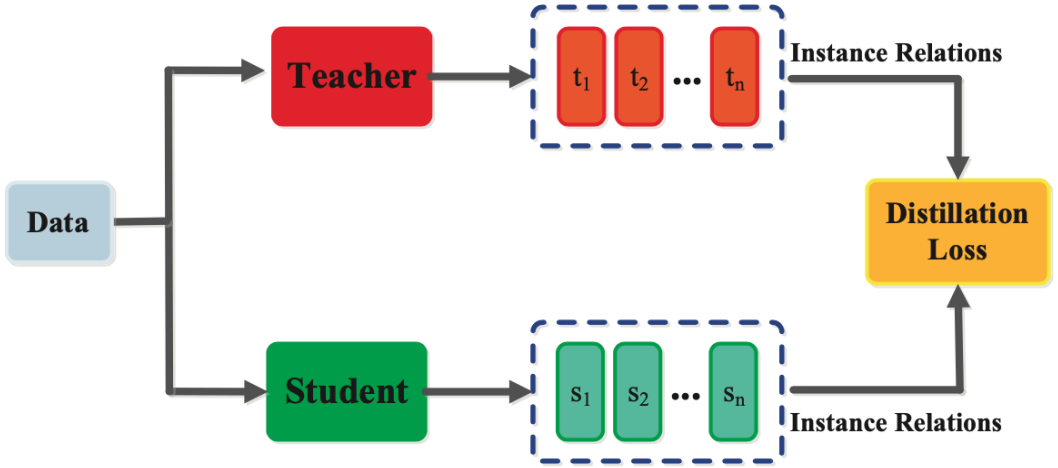
What Is Quantization?

- Reduce the **total number of bits** required to represent each parameter
  - usually converting floating-point numbers into integers



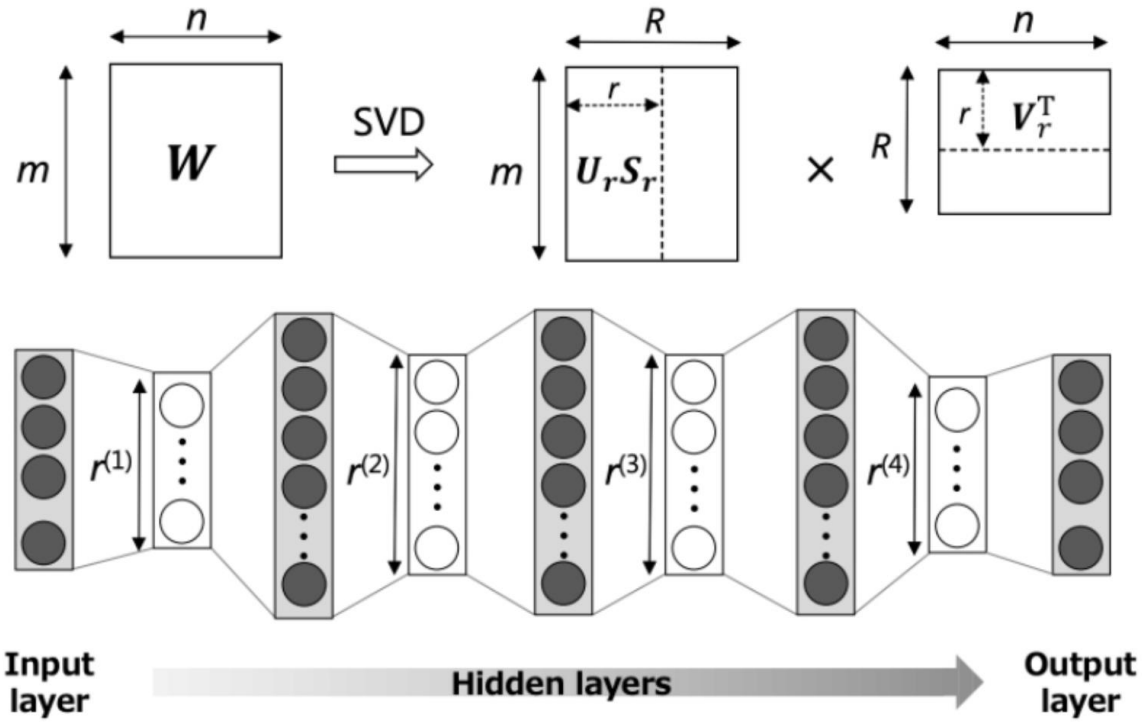
What Is Knowledge Distillation?

- Transferring experience from a largescale model (teacher) to a smaller-scale model (student)
  - extracting the informative aspects of a large model's behaviour
  - instilling this knowledge into a smaller model

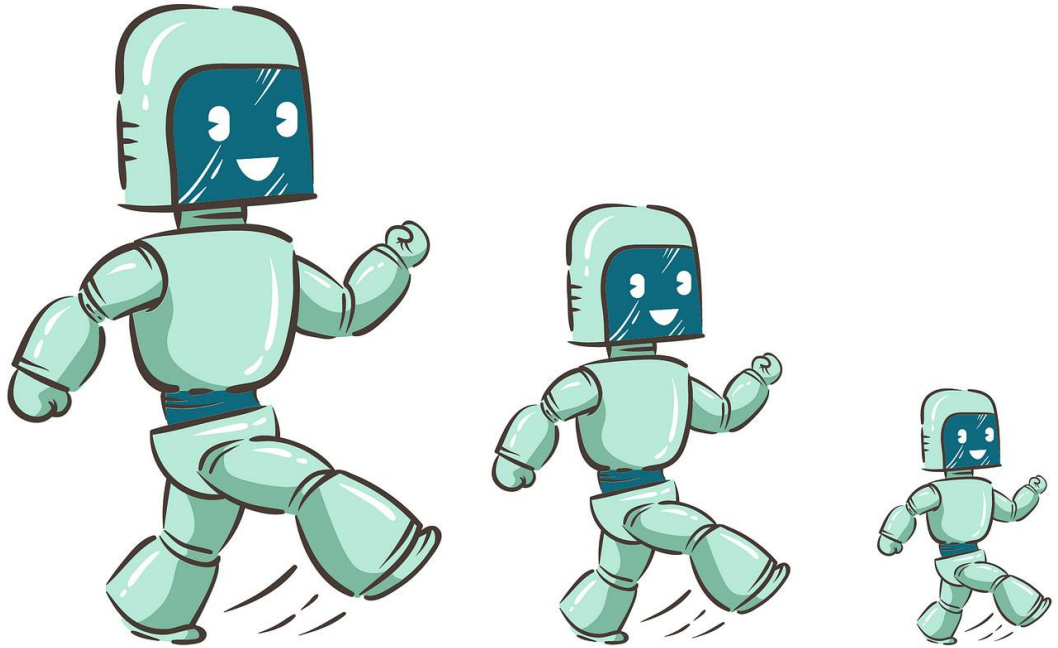


What Is Low-rank Factorization?

Decomposing large, dense weight matrices into smaller, lower-rank matrices.



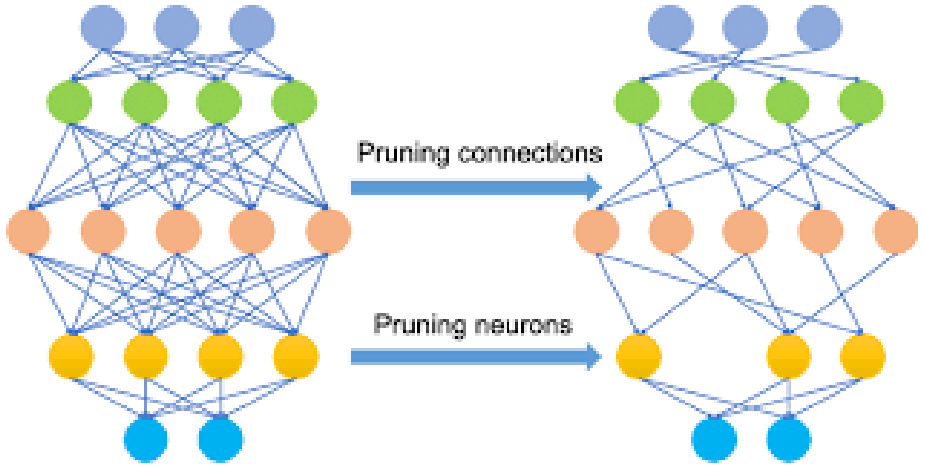
Pruning
in more details



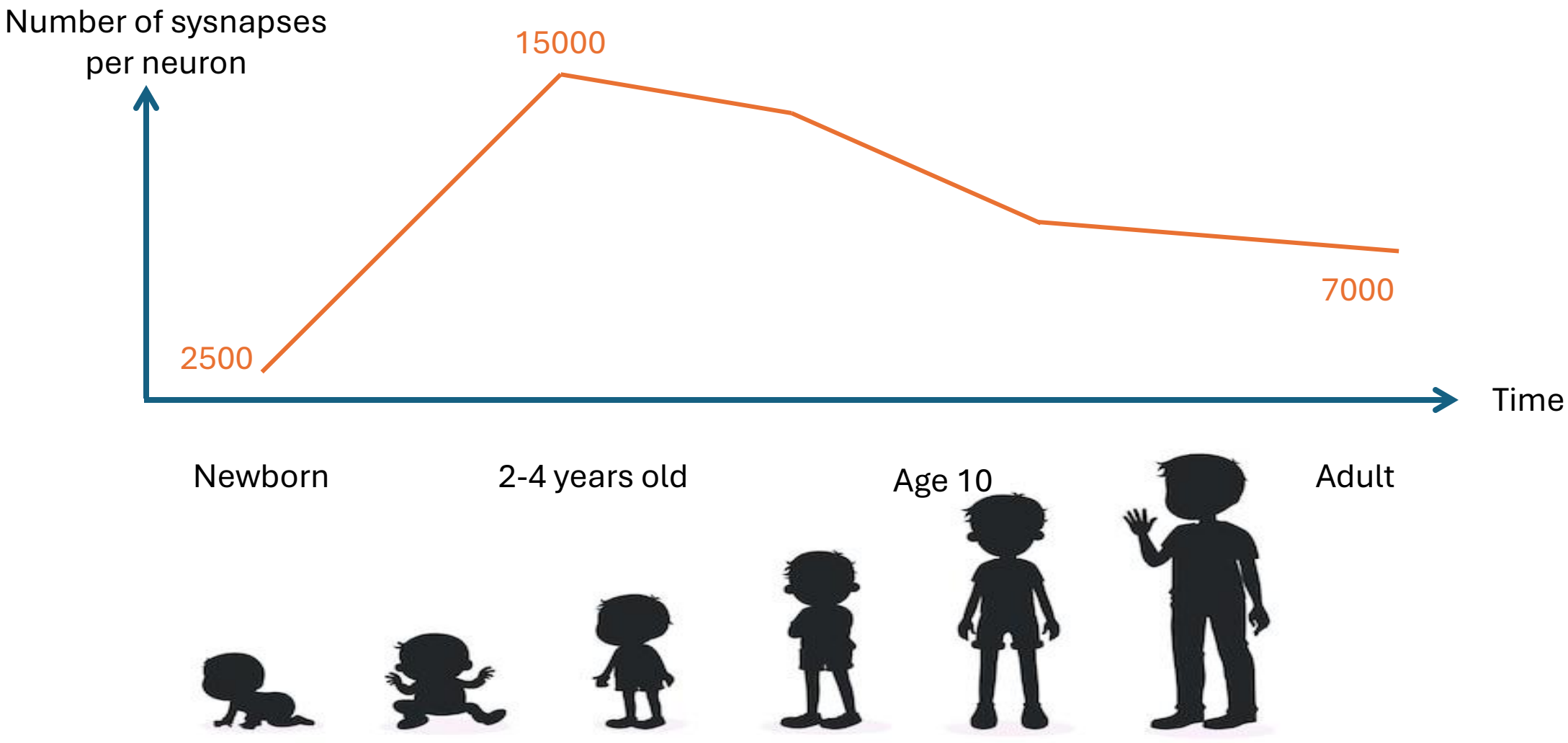
What Is Pruning?

- Also referred to as “Network Sparsification”

- Selective removal of network parameters (weights and neurons)
  - those that contribute the least to the network’s output



Pruning happens in human brain



Neural Network Pruning

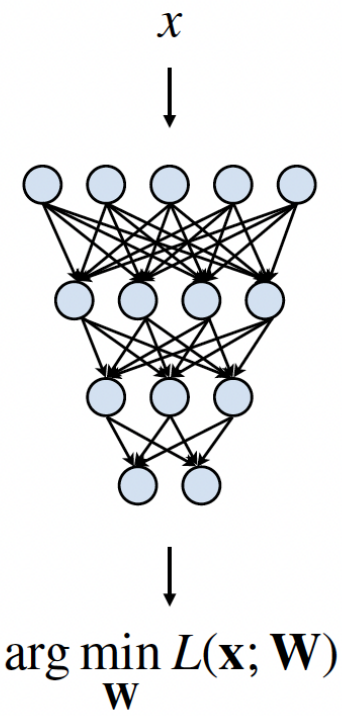
- In general, we could formulate the pruning as follows:

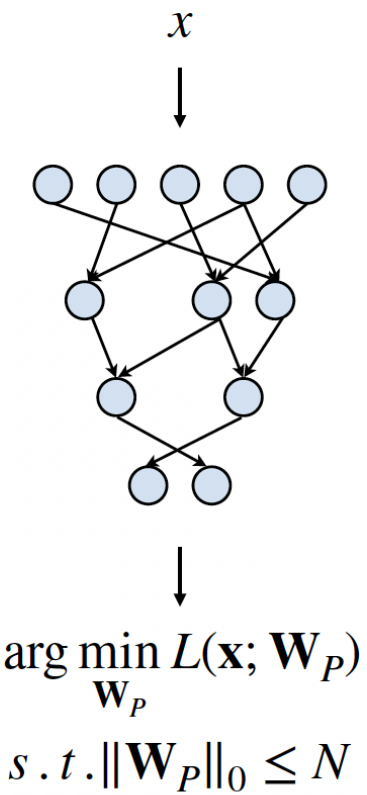
$$\arg\min_{\mathbf{W}_P} L(\mathbf{x}; \mathbf{W}_P)$$

subject to

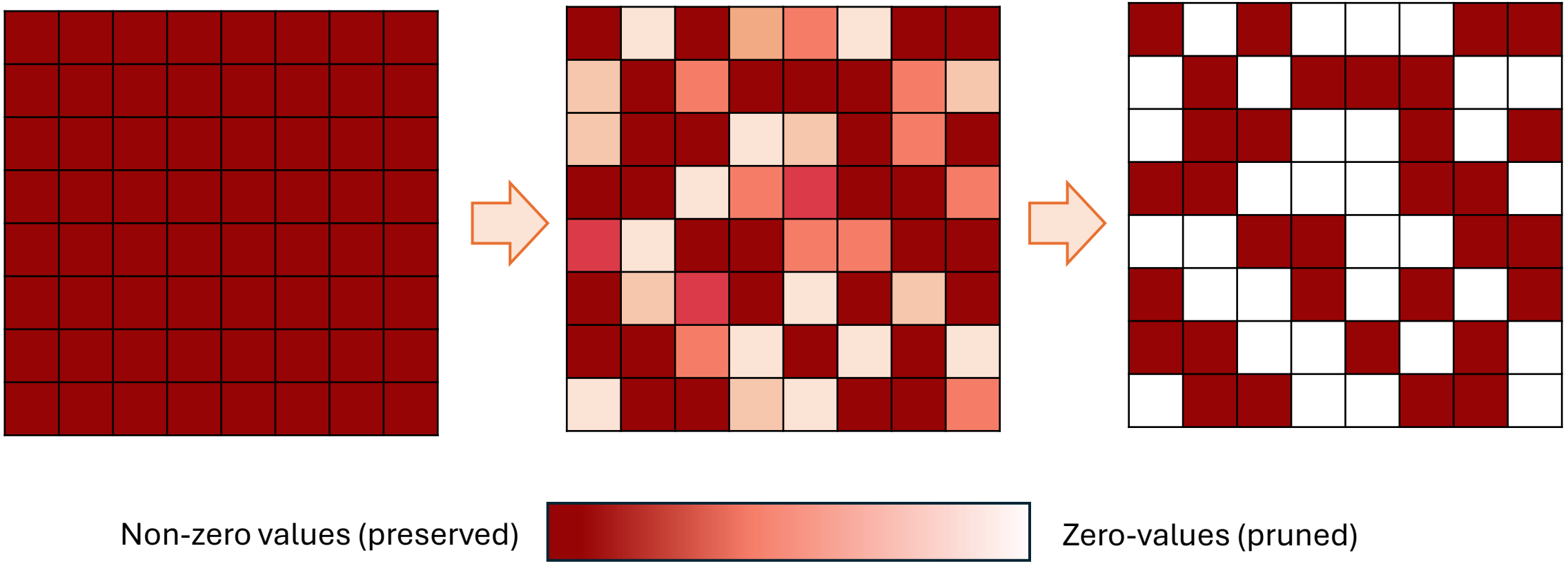
$$\|\mathbf{W}_p\|_0 < N$$

  - L represents the objective function for neural network training;
  - $\mathbf{x}$ is input, $\mathbf{W}$ is original weights, $\mathbf{W}_P$ is pruned weights;
  - $\|\mathbf{W}_p\|_0$ calculates the #nonzeros in W_P, and N is the target #nonzeros.





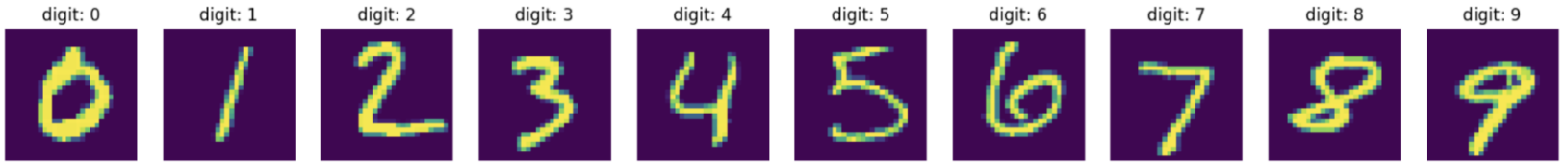
Neural Network Pruning



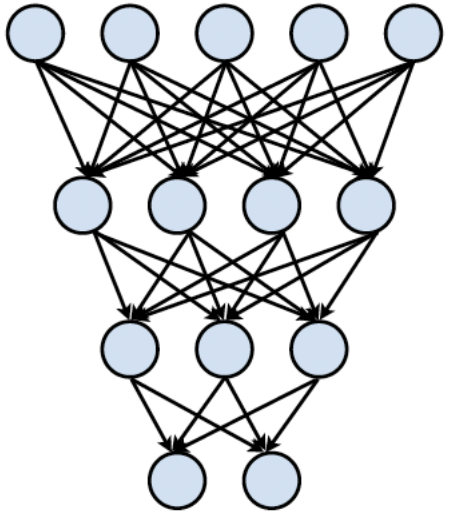
Let's see a demo!

- Refer to the notebook

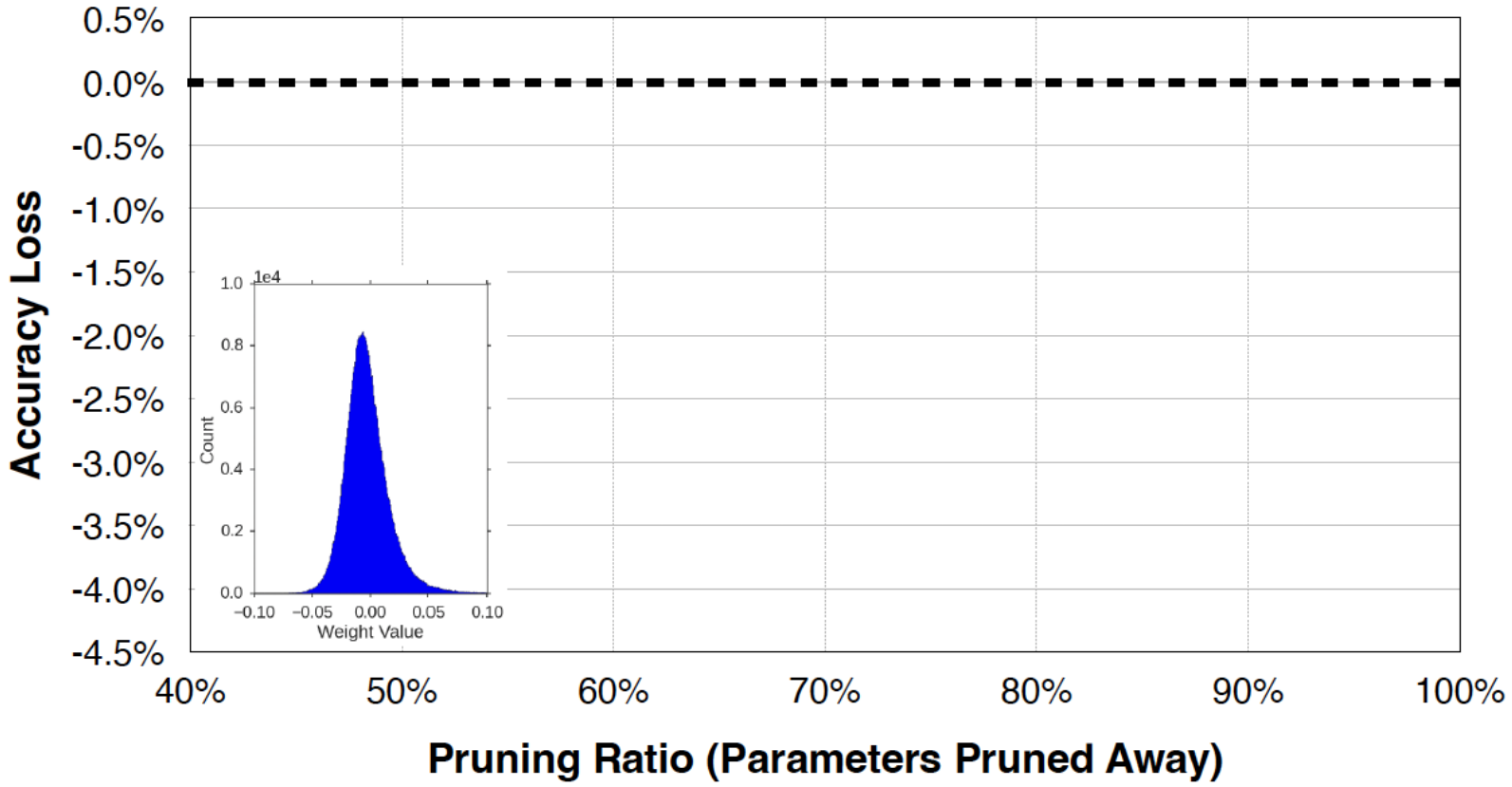
```
class Net(nn.Module):

    def __init__(self):

    super(Net, self).__init__()

    self.conv1 = nn.Conv2d(1, 32, 3, 1) # 1 x 32 x 3 x 3 = 288 parameters
    self.conv2 = nn.Conv2d(32, 64, 3, 1) # 32 x 64 x 3 x 3=18,432 paramters
    self.dropout1 = nn.Dropout(0.25)
    self.dropout2 = nn.Dropout(0.5)
    self.fc1 = nn.Linear(9216, 128) # 9216 x 128 = 1,179,648 parameters
    self.fc2 = nn.Linear(128, 10) # 128 x 10 = 1,280 parameters
```



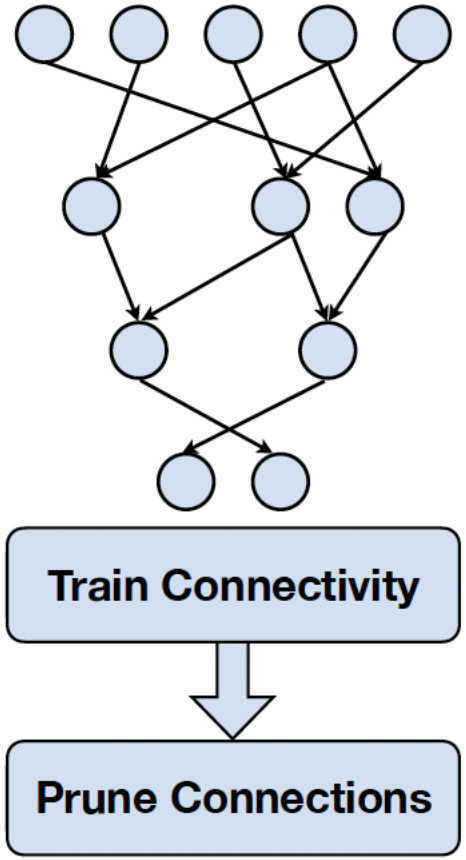
Neural network pruning

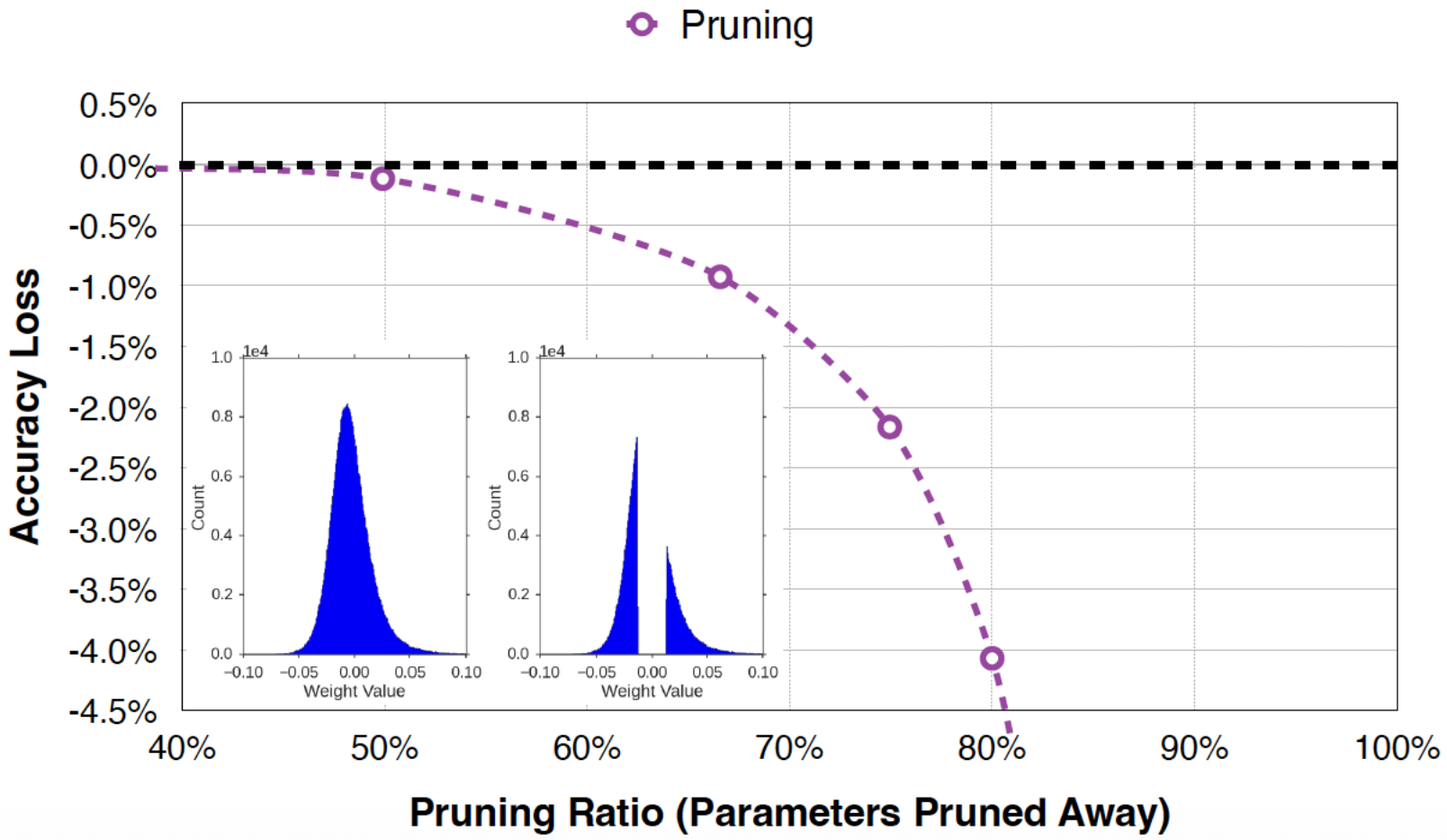


Train Connectivity

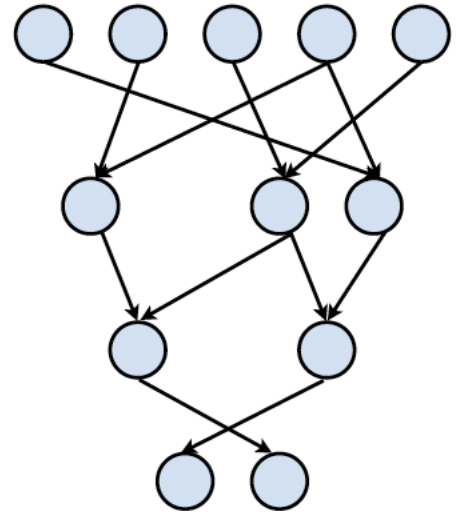


Neural network pruning





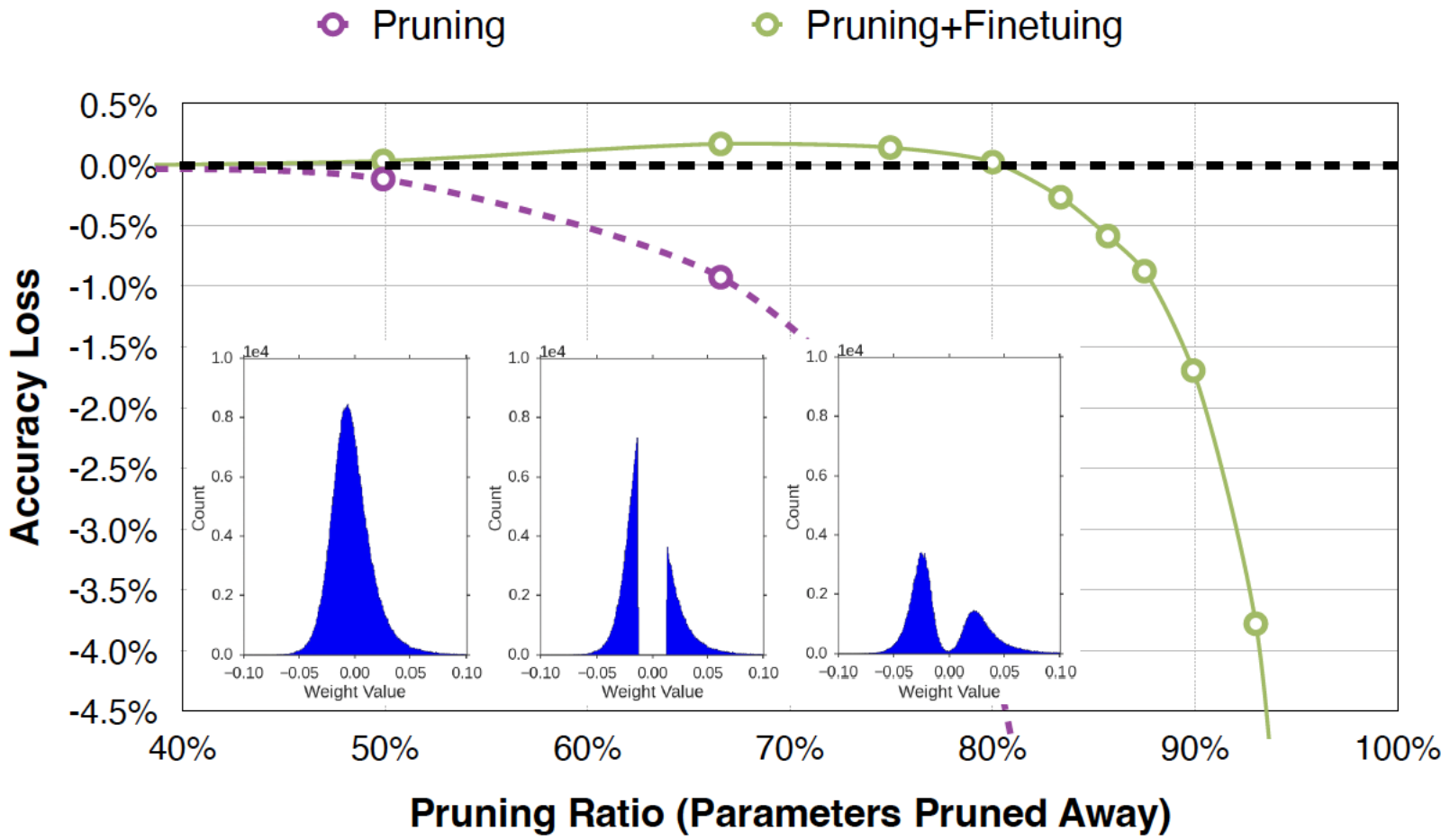
Neural network pruning



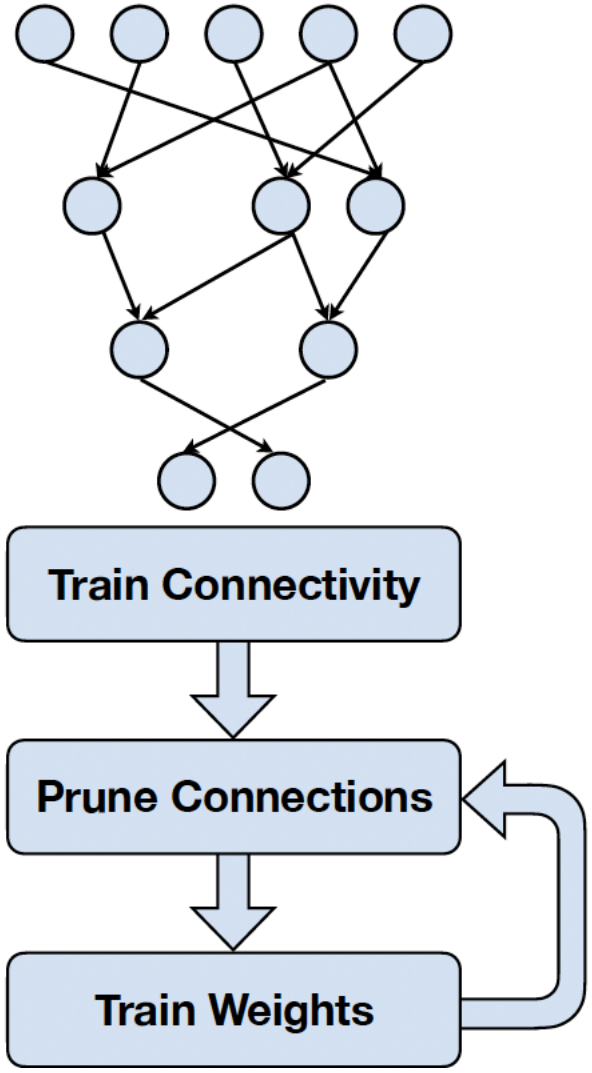
Train Connectivity

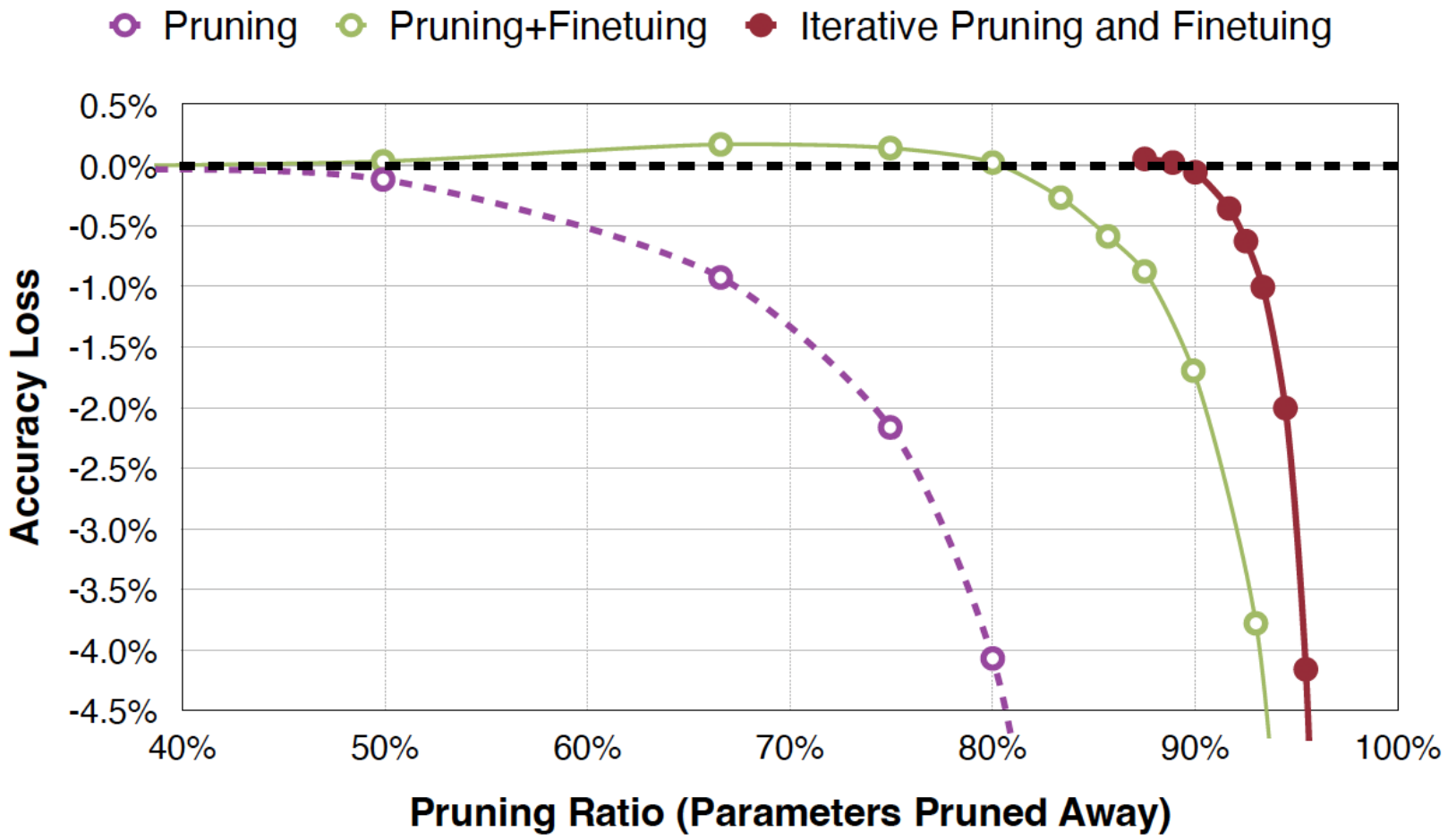
Prune Connections

Train Weights



Neural network pruning

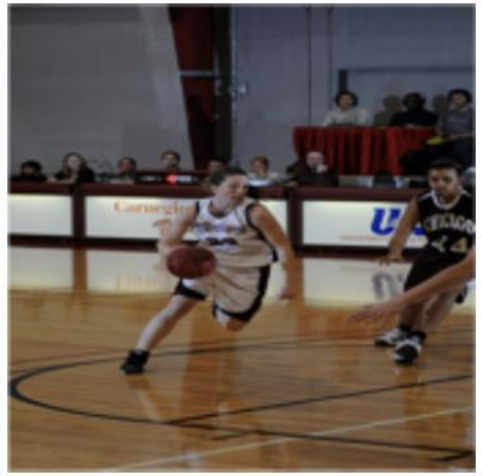




Neural Network Pruning

RI. SE

Pruning the NeuralTalk LSTM does not hurt image caption quality.



Baseline: a basketball player in a white uniform is playing with a ball .

Pruned 90%: a basketball player in a white uniform is playing with a basketball.

Baseline: a brown dog is running through a grassy field.

Pruned 90%: a brown dog is running through a grassy area.

Baseline: a man is riding a surfboard on a wave.

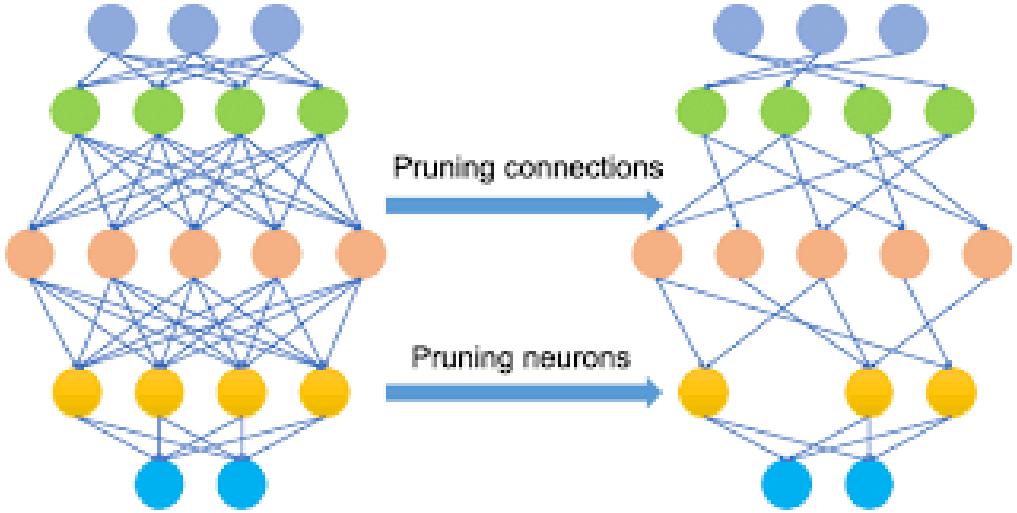
Pruned 90%: a man in a wetsuit is riding a wave on a beach.

Baseline: a soccer player in red is running in the field.

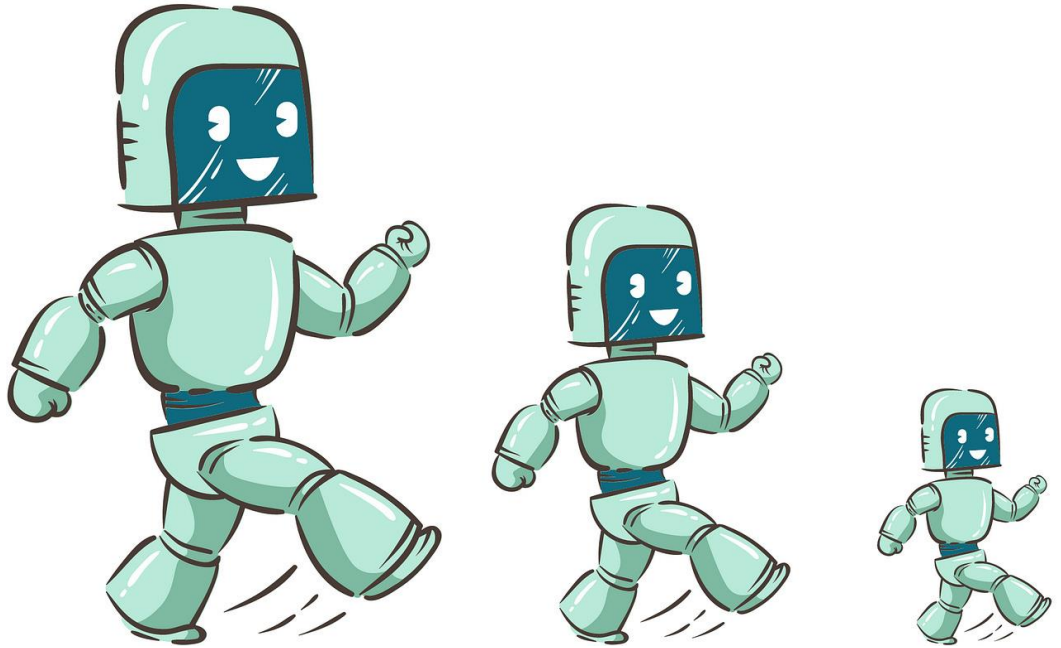
Pruned 95%: a man in a red shirt and black and white black shirt is running through a field.

Neural Network Pruning

- Make NNs smaller by removing connections and neurons
  - Which connections and neurons?
  - With what granularity?
  - How many?
  - When?



Pruning Criteria



Magnitude-based Pruning

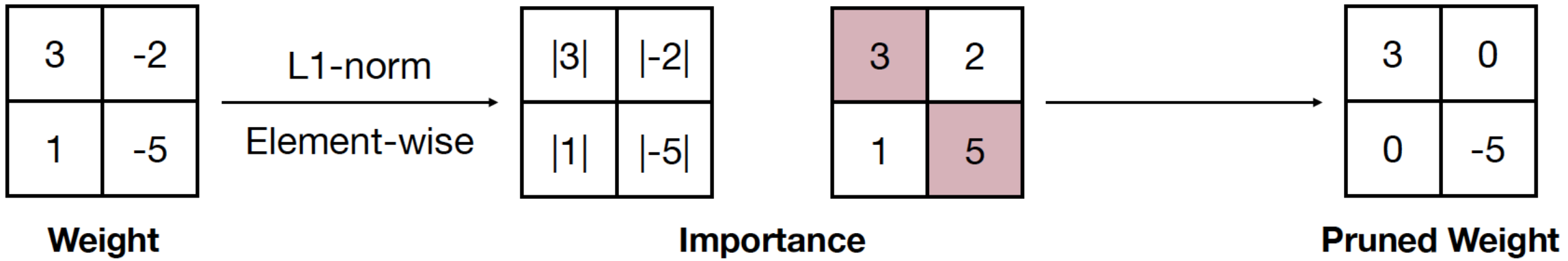
RI.
SE

A heuristic pruning criterion

- Magnitude-based pruning considers weights with ***larger absolute values*** are more important than other weights.
  - For element-wise pruning,

$$Importance = |W|$$

- **Example**



Learning Both Weights and Connections for Efficient Neural Network [Han *et al*., NeurIPS 2015]

Magnitude-based Pruning

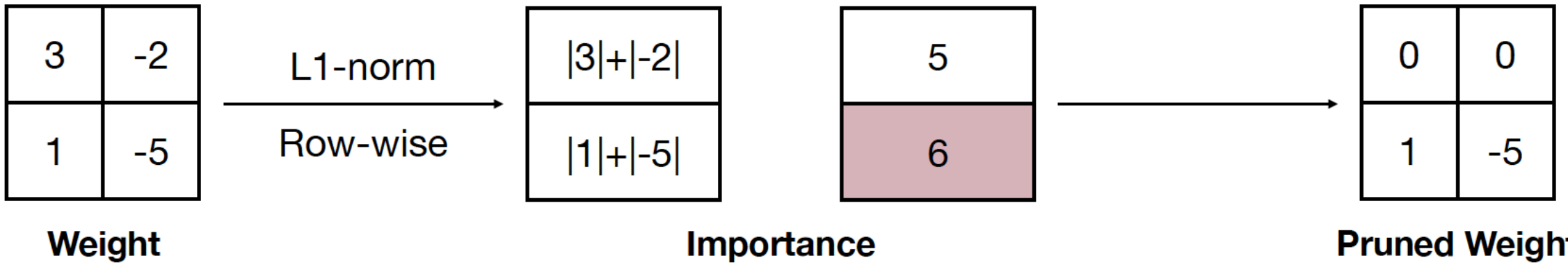
RI. SE

A heuristic pruning criterion

- Magnitude-based pruning considers weights with ***larger absolute values*** are more important than other weights.
  - For row-wise pruning, the L1-norm magnitude can be defined as,

$$Importance = \sum_{i \in S} |w_i|, \text{ where } \mathbf{W}^{(S)} \text{ is the structural set } S \text{ of parameters } \mathbf{W}$$

- **Example**



Learning Both Weights and Connections for Efficient Neural Network [Han *et al*., NeurIPS 2015]

Magnitude-based Pruning

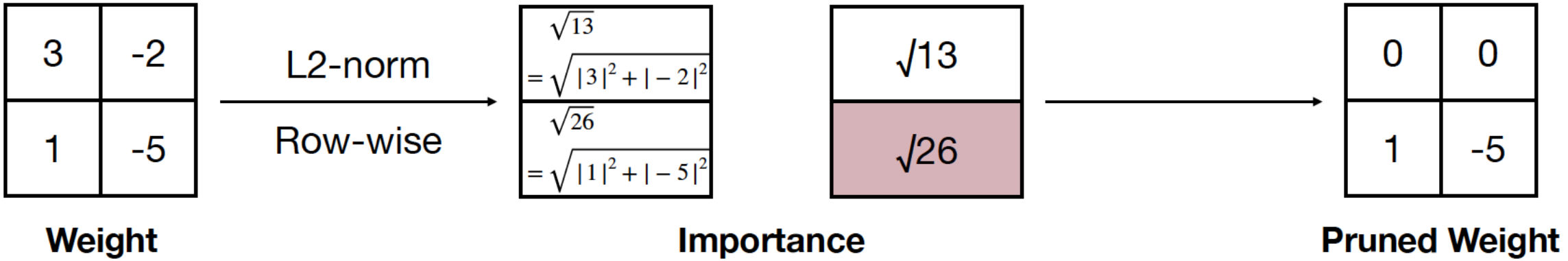
RI.
SE

A heuristic pruning criterion

- Magnitude-based pruning considers weights with ***larger absolute values*** are more important than other weights.
  - For row-wise pruning, the L2-norm magnitude can be defined as,

$$Importance = \sqrt{\sum_{i \in S} |w_i|^2}, \text{ where } \mathbf{W}^{(S)} \text{ is the structural set } S \text{ of parameters } \mathbf{W}$$

- **Example**



Learning Both Weights and Connections for Efficient Neural Network [Han *et al*., NeurIPS 2015]

Magnitude-based Pruning

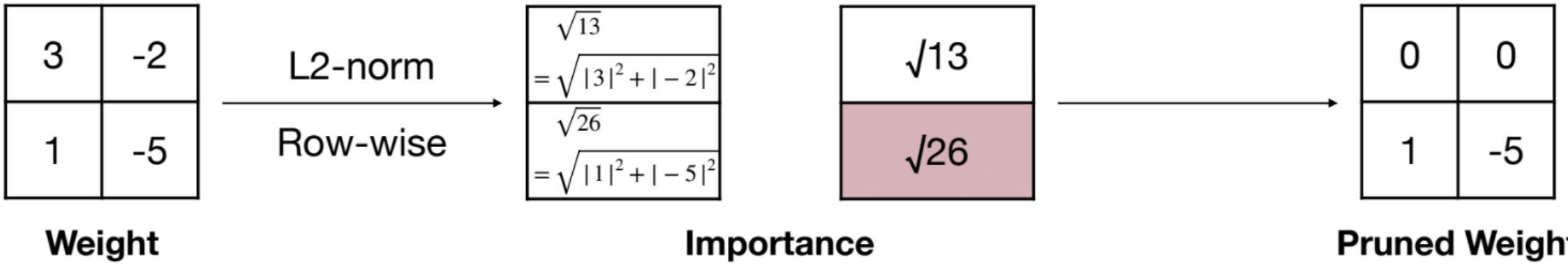
RI. SE

A heuristic pruning criterion

- Magnitude-based pruning considers weights with ***larger absolute values*** are more important than other weights.
- Magnitude is also known as L_p-norm defined as,

$$\|\mathbf{W}^{(S)}\|_p = \left(\sum_{i \in S} |w_i|^p \right)^{\frac{1}{p}}, \text{ where } \mathbf{W}^{(S)} \text{ is a structural set of parameters}$$

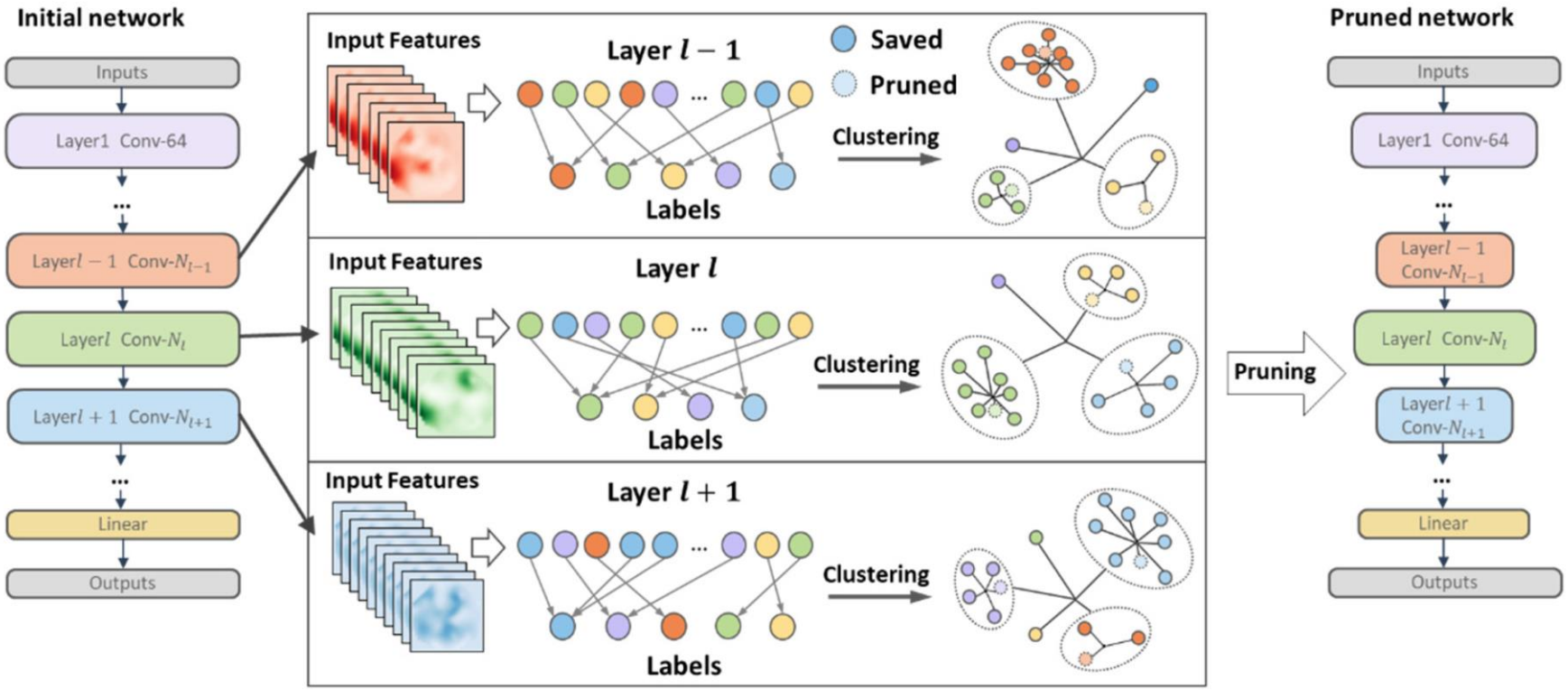
- **Example**



Learning Structured Sparsity in Deep Neural Networks [Wen *et al.*, NeurIPS 2016]

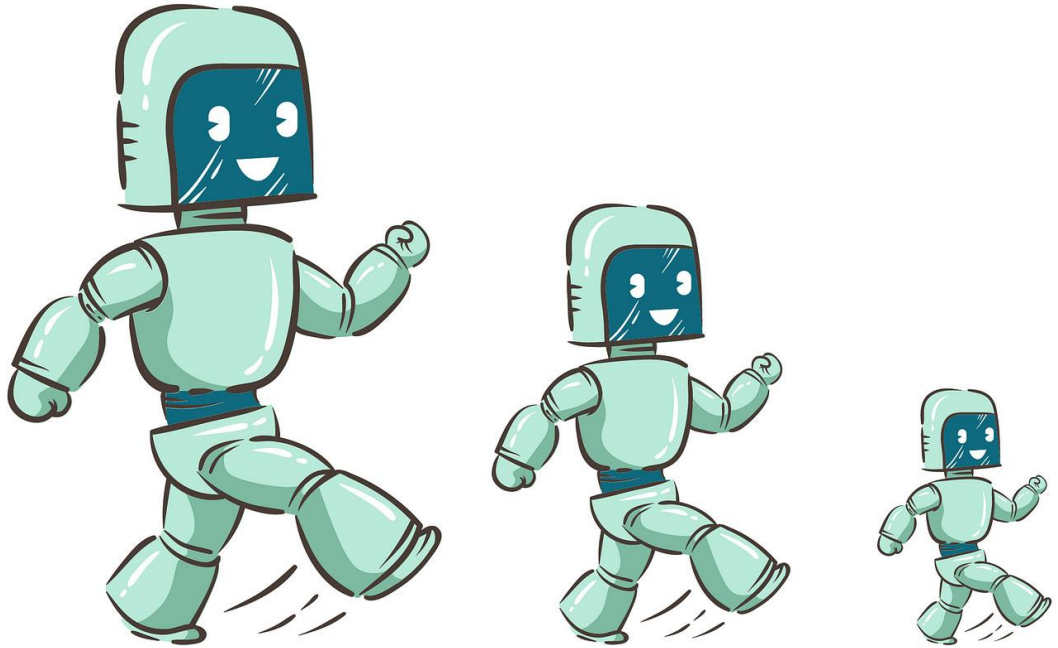
Pruning Based On Similarity And Clustering

- Remove the **redundancy across weights or filters** in the network.



Wu T, Song C, Zeng P, Xia C. Cluster-Based Structural Redundancy Identification for Neural Network Compression. *Entropy*. 2023

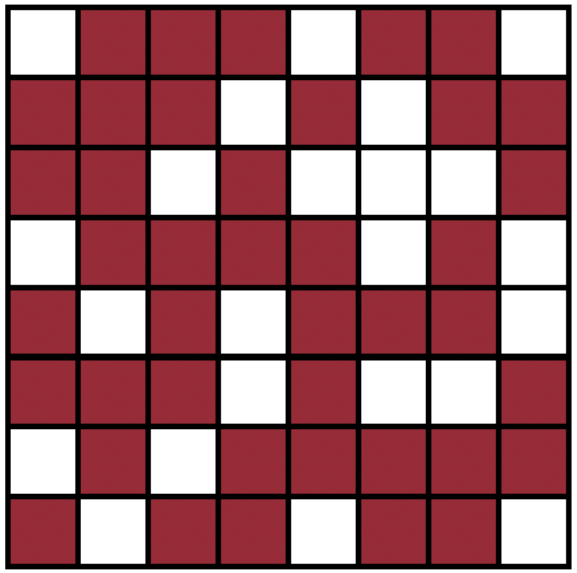
Prunning Granularity

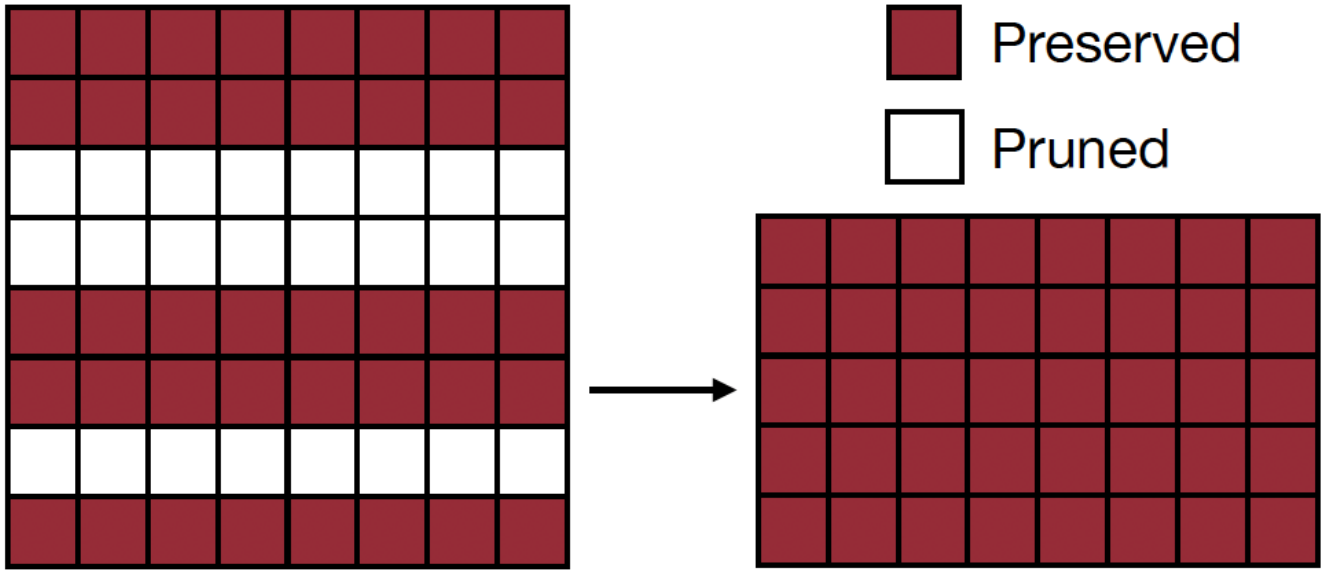


Pruning at Different Granularities

RI. SE

A simple example of 2D weight matrix





Fine-grained/Unstructured

- More flexible pruning index choice
- Hard to accelerate (irregular)

Coarse-grained/Structured

- Less flexible pruning index choice (a subset of the fine-grained case)
- Easy to accelerate (just a smaller matrix!)

Pruning at Different Granularities

RI. SE

The case of convolutional layers

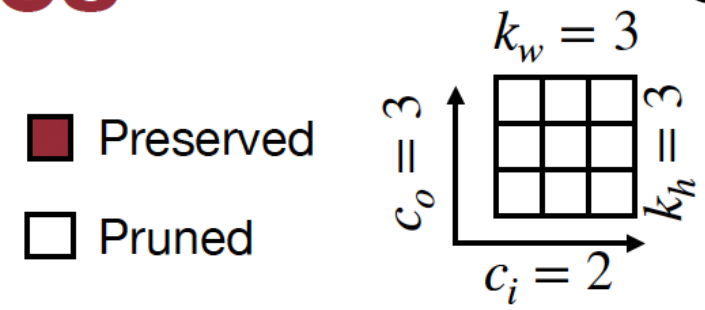
- The weights of convolutional layers have 4 dimensions $[c_o, c_i, k_h, k_w]$:
  - c_i: input channels (or channels)
  - c_o: output channels (or filters)
  - k_h: kernel size height
  - k_w: kernel size width

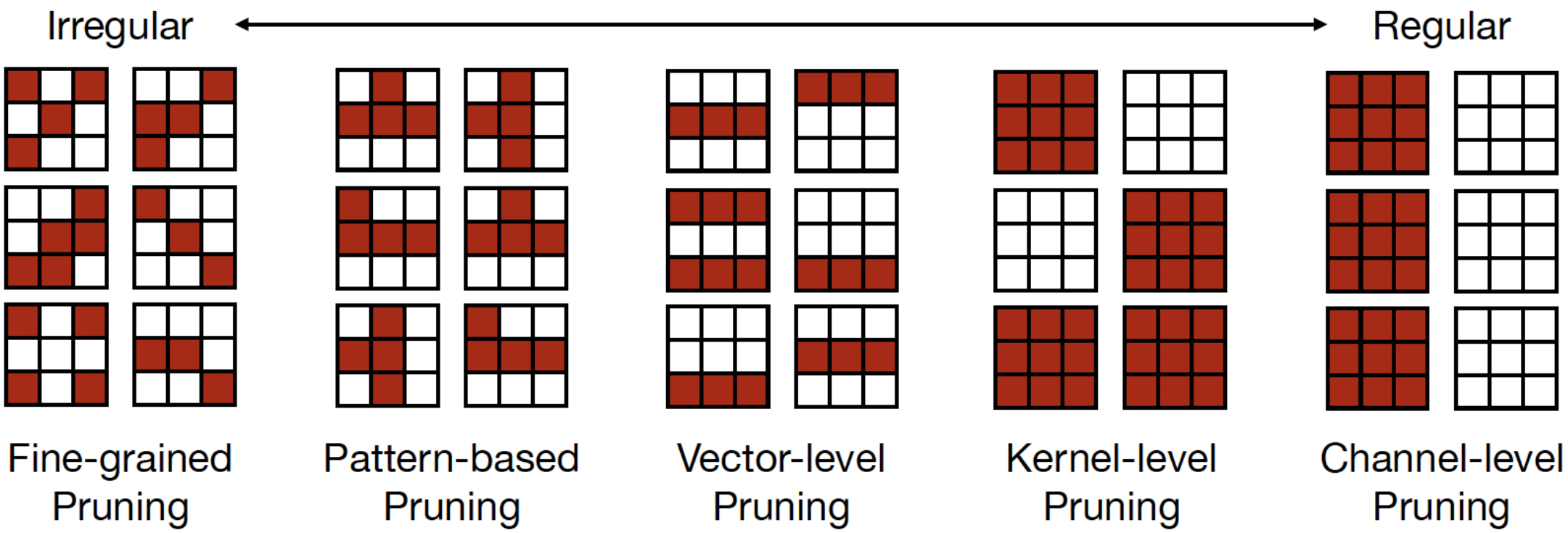
- The 4 dimensions give us more choices to select pruning granularities

Pruning at Different Granularities

The case of convolutional layers

- Some of the commonly used pruning granularities





like Tetris :)

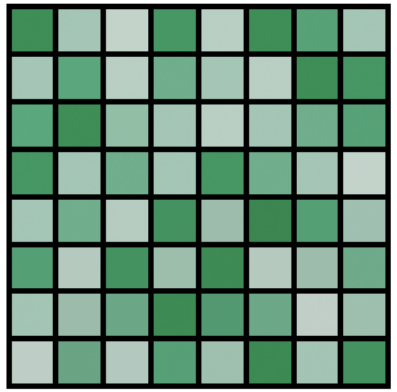
Exploring the granularity of sparsity in convolutional neural networks [Mao *et al.*, CVPR-W]

Pruning at Different Granularities

RI.
SE

Let's look into some cases

- **Pattern-based Pruning: N:M sparsity**



Dense Matrix

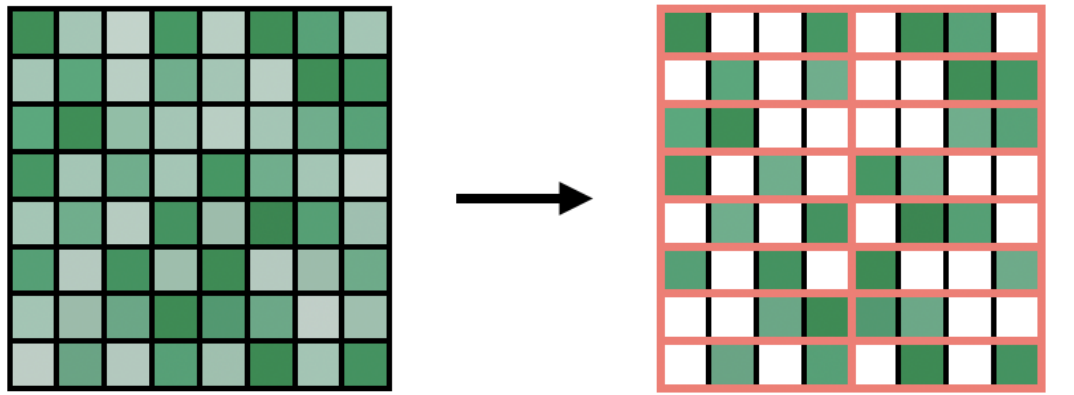
Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT

Pruning at Different Granularities

RI. SE

Let's look into some cases

- **Pattern-based Pruning: N:M sparsity**



Dense Matrix

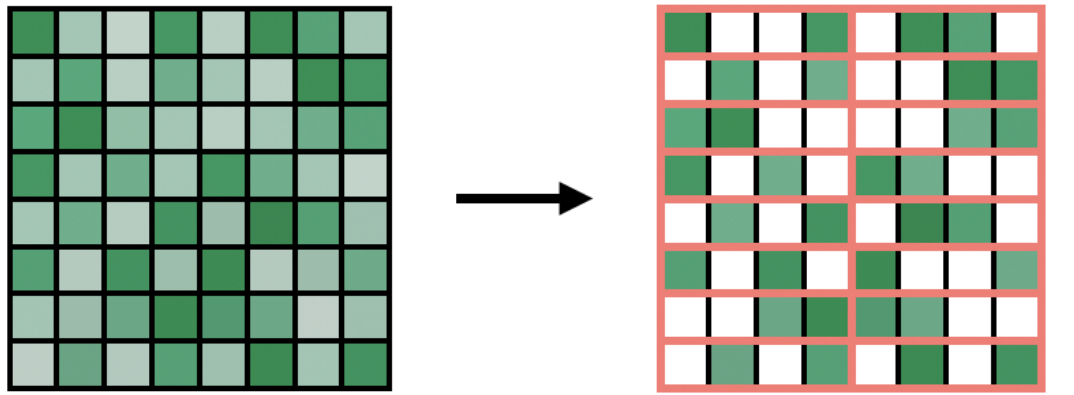
2:4 Sparse Matrix

Pruning at Different Granularities

RI. SE

Let's look into some cases

- **Pattern-based Pruning: N:M sparsity**
  - N:M sparsity means that in each contiguous M elements, N of them is pruned
  - A classic case is 2:4 sparsity (50% sparsity)



Dense Matrix

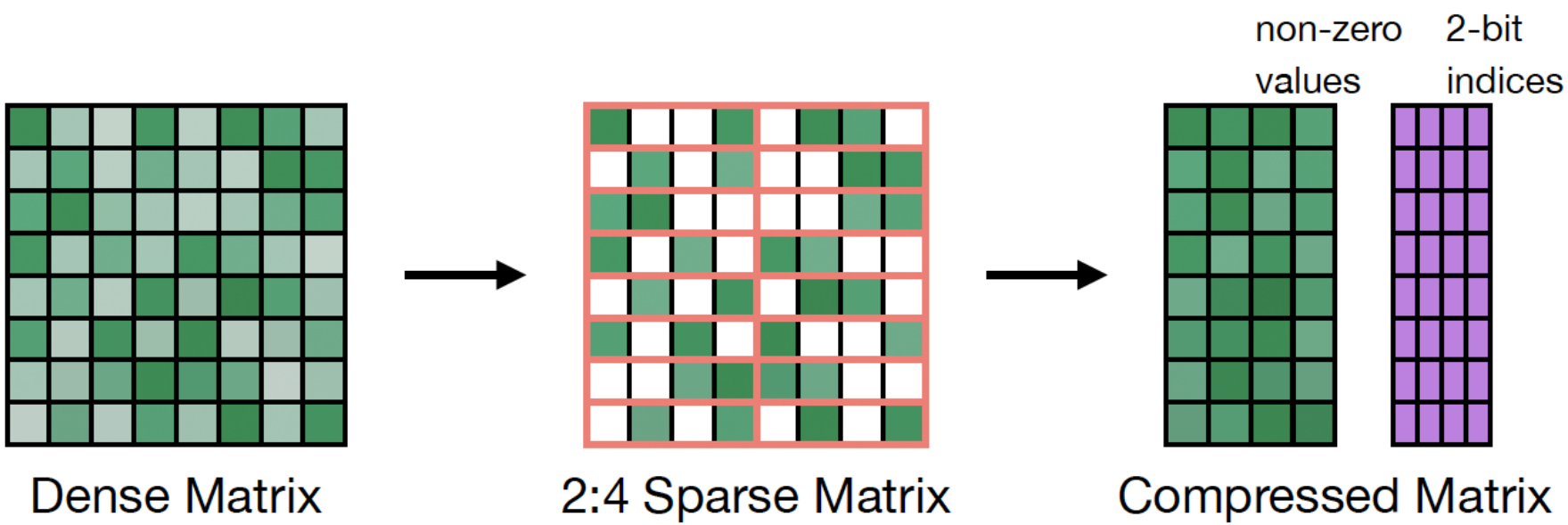
2:4 Sparse Matrix

Pruning at Different Granularities

RI. SE

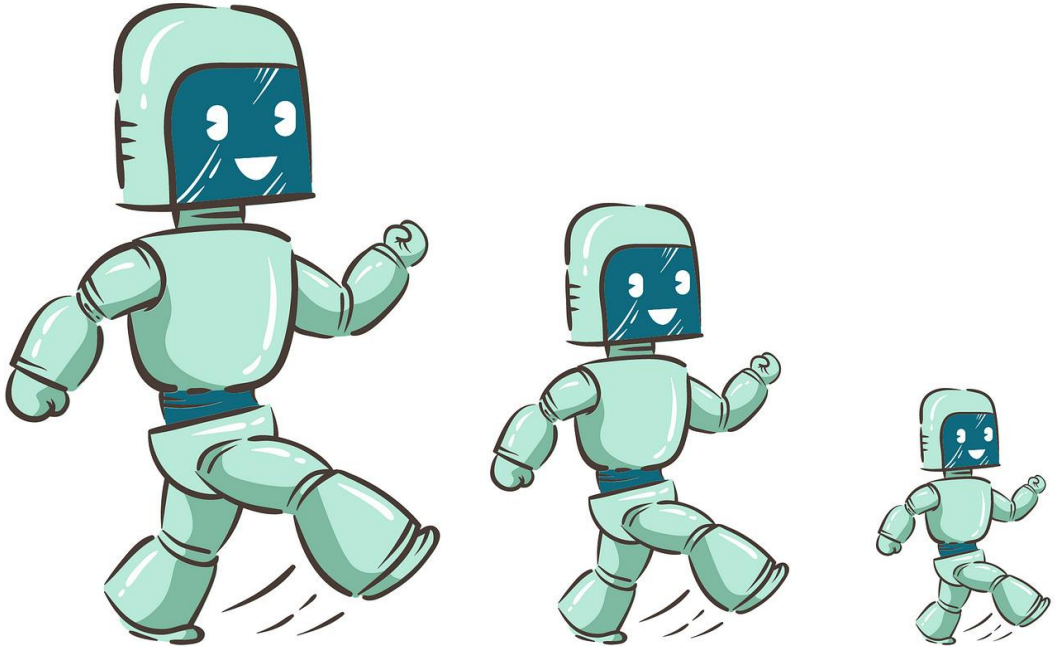
Let's look into some cases

- **Pattern-based Pruning: N:M sparsity**
  - N:M sparsity means that in each contiguous M elements, N of them is pruned
  - A classic case is 2:4 sparsity (50% sparsity)
  - It is supported by NVIDIA's Ampere GPU Architecture, which delivers up to 2x speed up



Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT

Pruning Ratio



Finding Pruning Ratios

RI.
SE

Analyze the sensitivity of each layer

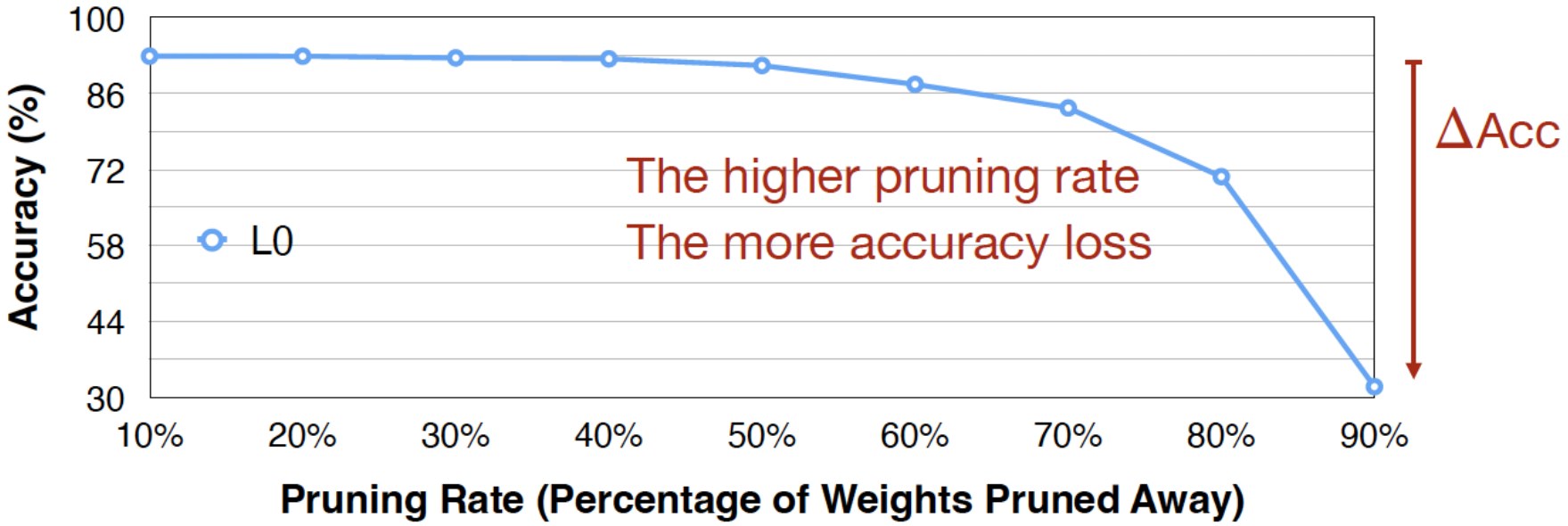
- We need different pruning ratios for each layer since different layers have different **sensitivity**
  - Some layers are more sensitive (e.g., first layer)
  - Some layers are more redundant

Finding Pruning Ratios

RI. SE

Analyze the sensitivity of each layer

- The process of Sensitivity Analysis (* VGG-11 on CIFAR-10 dataset)
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, ..., 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio

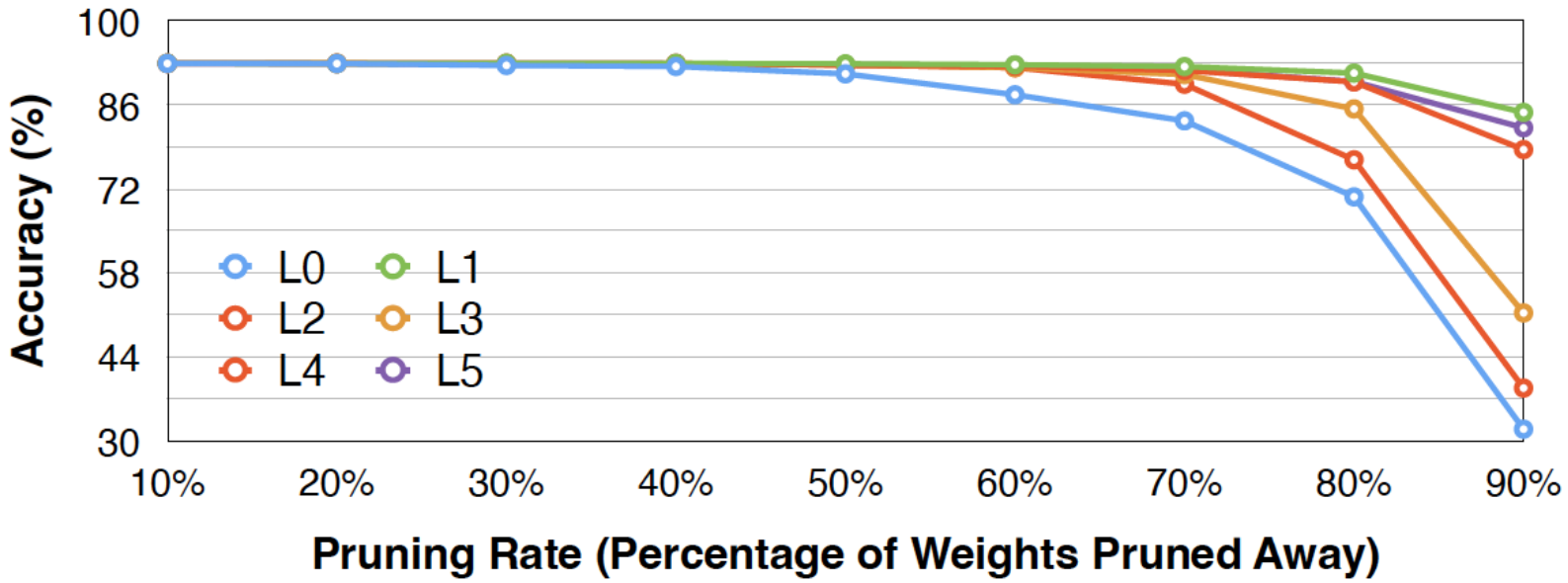


Finding Pruning Ratios

RI. SE

Analyze the sensitivity of each layer

- The process of Sensitivity Analysis (* VGG-11 on CIFAR-10 dataset)
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, ..., 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio
  - Repeat the process for all layers

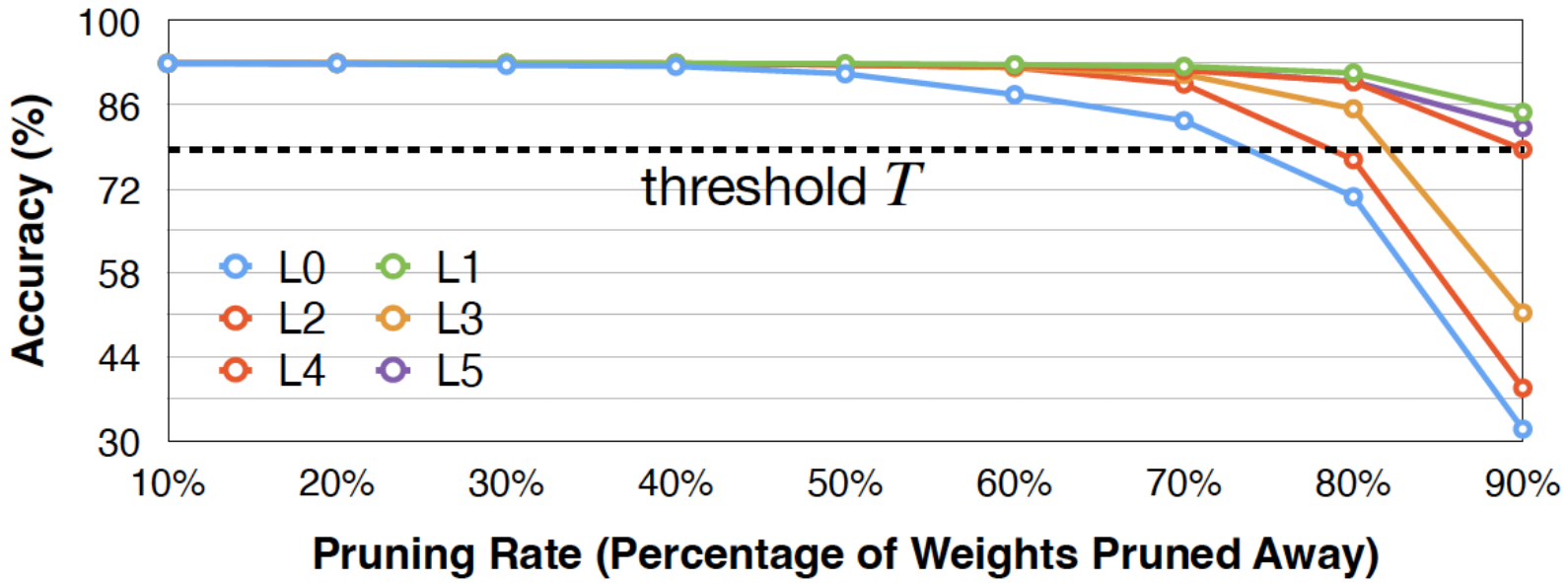


Finding Pruning Ratios

RI. SE

Analyze the sensitivity of each layer

- The process of Sensitivity Analysis (* VGG-11 on CIFAR-10 dataset)
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, ..., 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio
  - Repeat the process for all layers
  - Pick a degradation threshold T such that the overall pruning rate is desired

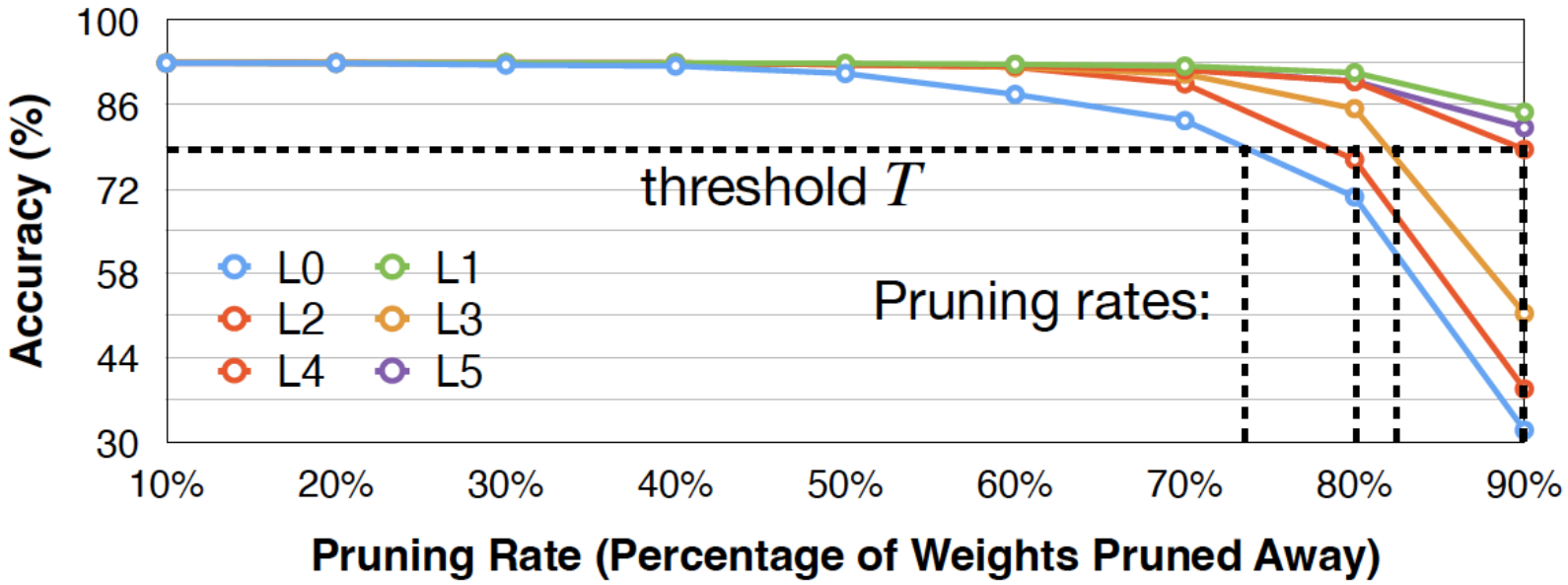


Finding Pruning Ratios

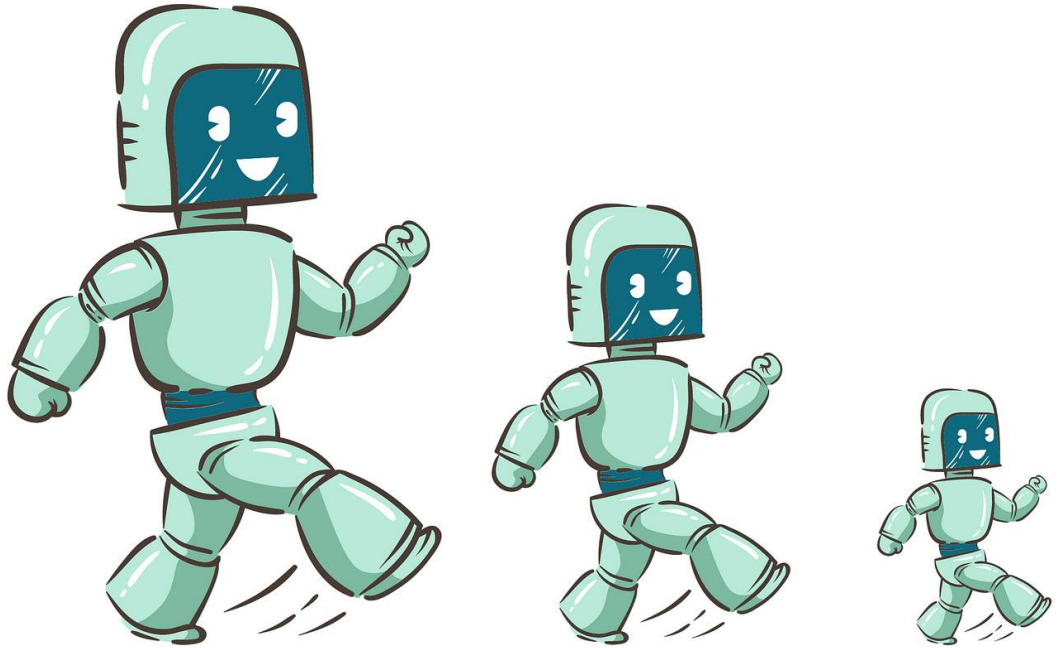
RI. SE

Analyze the sensitivity of each layer

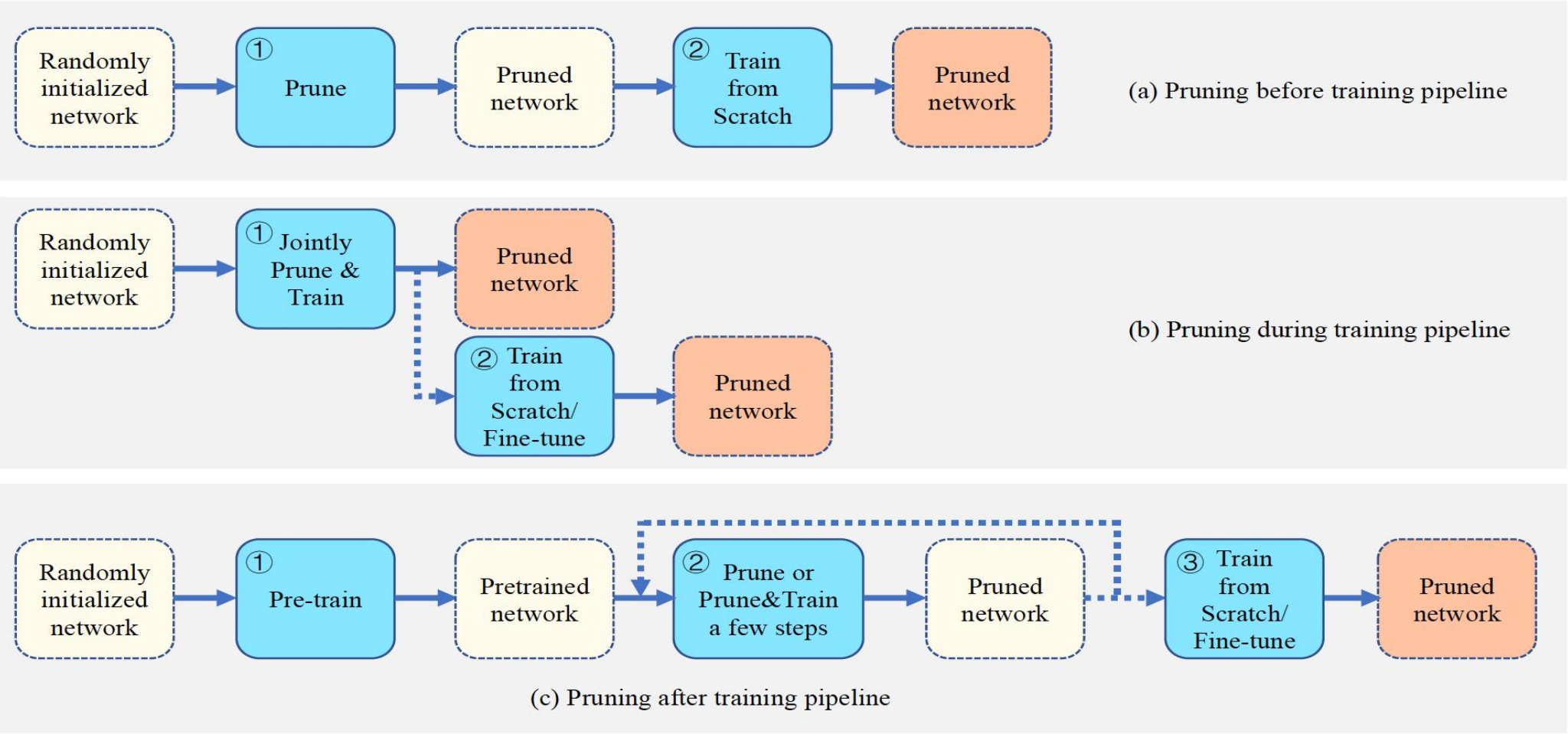
- The process of Sensitivity Analysis (* VGG-11 on CIFAR-10 dataset)
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, ..., 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio
  - Repeat the process for all layers
  - Pick a degradation threshold T such that the overall pruning rate is desired



When To Prune?



When To Prune?



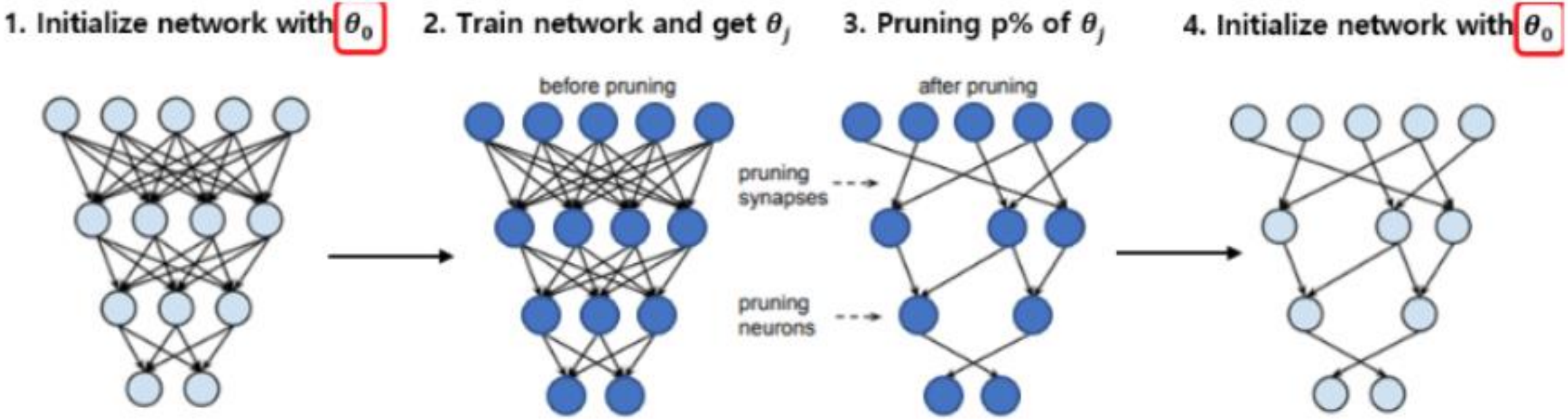
The lottery ticket hypothesis

- Within a **large, randomly-initialized network**, there exists a much **smaller subnetwork** that, if trained in isolation from the beginning with the **same initial weights**, can reach **comparable performance** to the full network.

- That's what they call a *'winning ticket'*

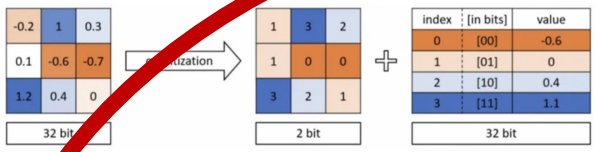
Frankle, J., & Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks.

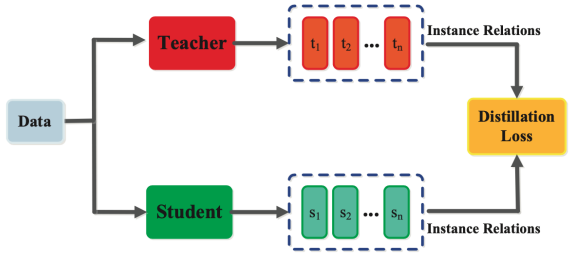
The lottery ticket hypothesis

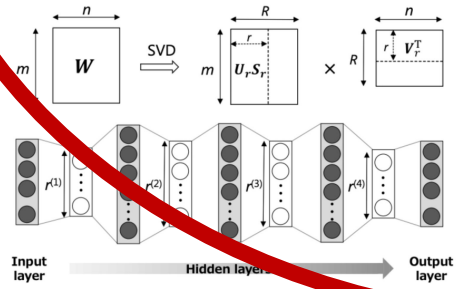


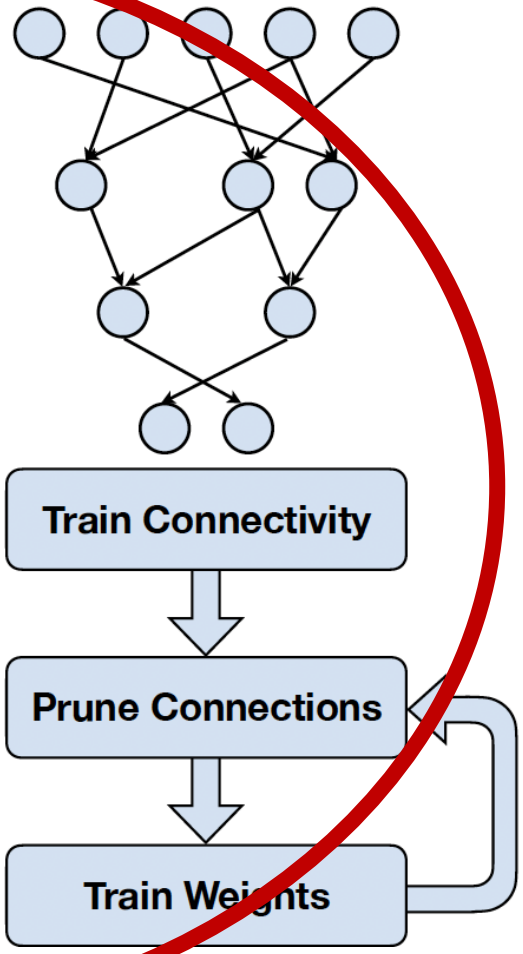
Frankle, J., & Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks.

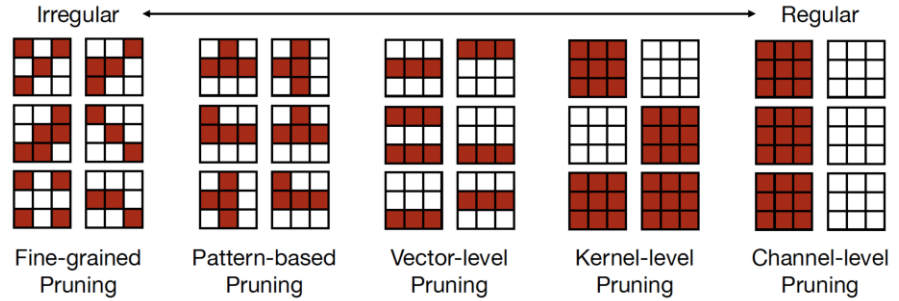
Takeaways

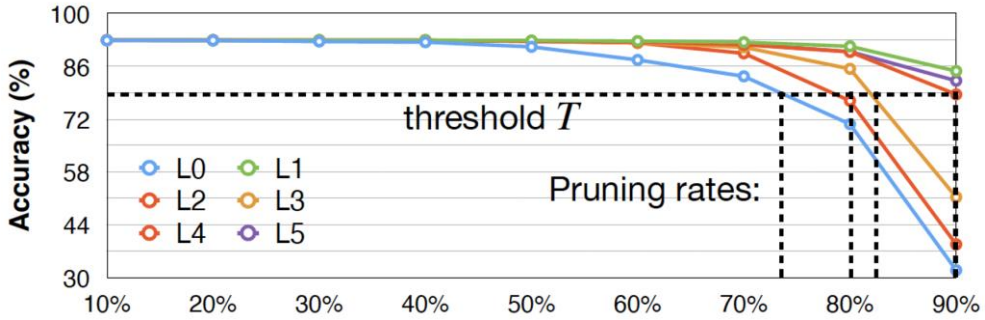

-0.2 1 0.3
0.1 -0.6 -0.7
1.2 0.4 0
32 bit
1 3 2
1 0 0
3 2 1
2 bit
index [in bits] value
0 [00] -0.6
1 [01] 0
2 [10] 0.4
3 [11] 1.1
32 bit

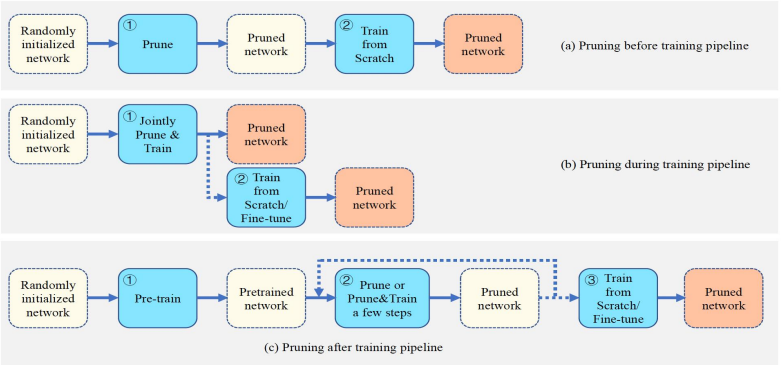

Teacher
Data
Student
Instance Relations
Distillation Loss
Instance Relations


W
SVD
Input layer
Hidden layer
Output layer


Train Connectivity
Prune Connections
Train Weights


Irregular
Regular
Fine-grained Pruning
Pattern-based Pruning
Vector-level Pruning
Kernel-level Pruning
Channel-level Pruning


Accuracy (%)
100
86
72
58
44
30
threshold T
L0 L1
L2 L3
L4 L5
Pruning rates:
10% 20% 30% 40% 50% 60% 70% 80% 90%


Randomly initialized network
① Prune
Pruned network
② Train from Scratch
Pruned network
(a) Pruning before training pipeline
Randomly initialized network
① Jointly Prune & Train
Pruned network
② Train from Scratch/ Fine-tune
Pruned network
(b) Pruning during training pipeline
Randomly initialized network
① Pre-train
Pretrained network
② Prune or Prune&Train a few steps
Pruned network
③ Train from Scratch/ Fine-tune
Pruned network
(c) Pruning after training pipeline

References

- Cheng, H., Zhang, M., & Shi, J. Q. (2024). A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Frankle, J., & Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.
- Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems, 28*.
- Mao, H., Han, S., Pool, J., & Dally, W. J. (2017). Exploring the granularity of sparsity in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.
- Vadera, S., & Ameen, S. (2022). Methods for pruning deep neural networks. *IEEE Access, 10*, 63280–63300.
- Wu, T., Song, C., Zeng, P., & Xia, C. (2023). Cluster-based structural redundancy identification for neural network compression. *Entropy*.
- *TinyML and Efficient Deep Learning Computing*, MIT Course, Fall 2024. Retrieved from https://efficientml.ai

Thank you!