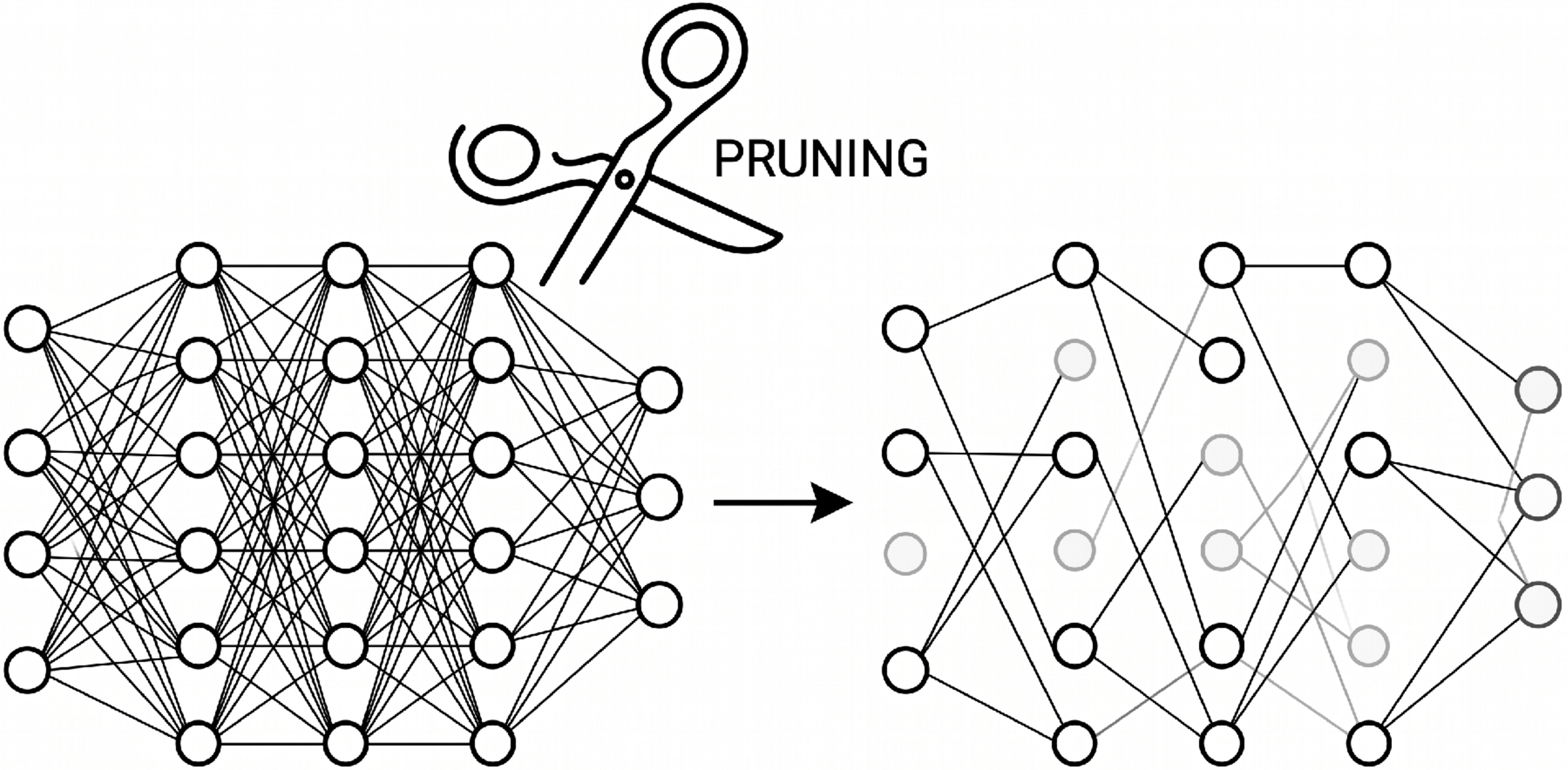
CSCI1470

Deep Learning

Randall Balestriero

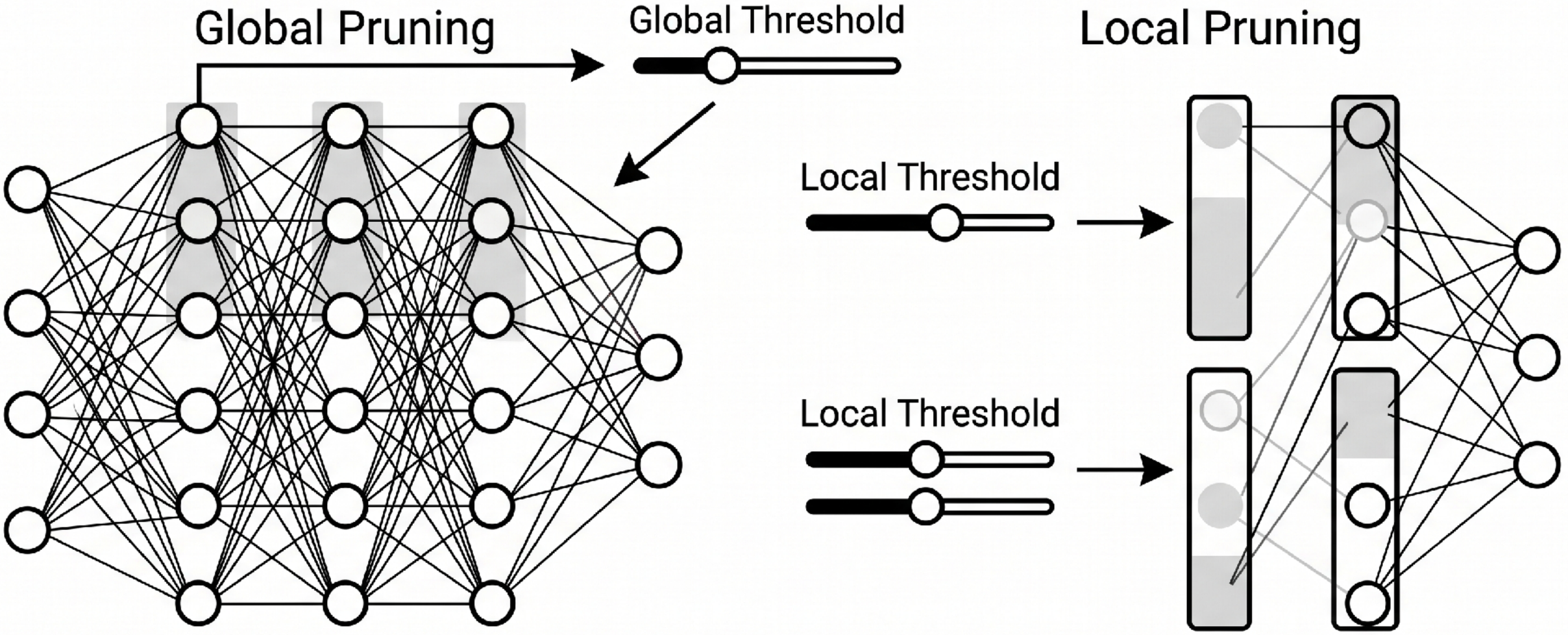
Recap



What is Model Pruning? (Simplifying the Model)

Any idea?

Global vs. Local Pruning



Global compares all weights simultaneously

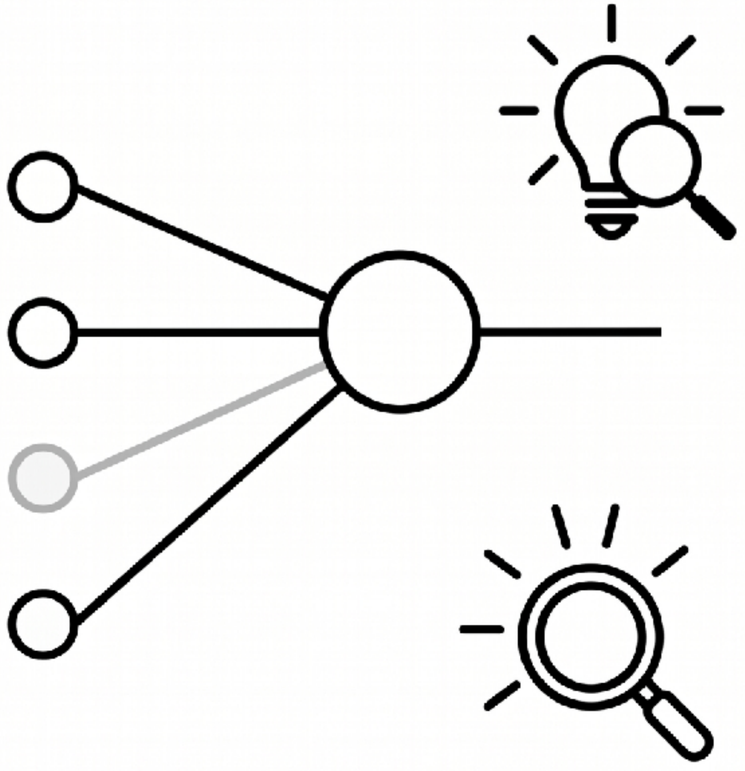
Local compares weights only within each layer

Alternative Pruning Criteria

Magnitude (Individual Size)

Cut small weights
(re-visiting importance)

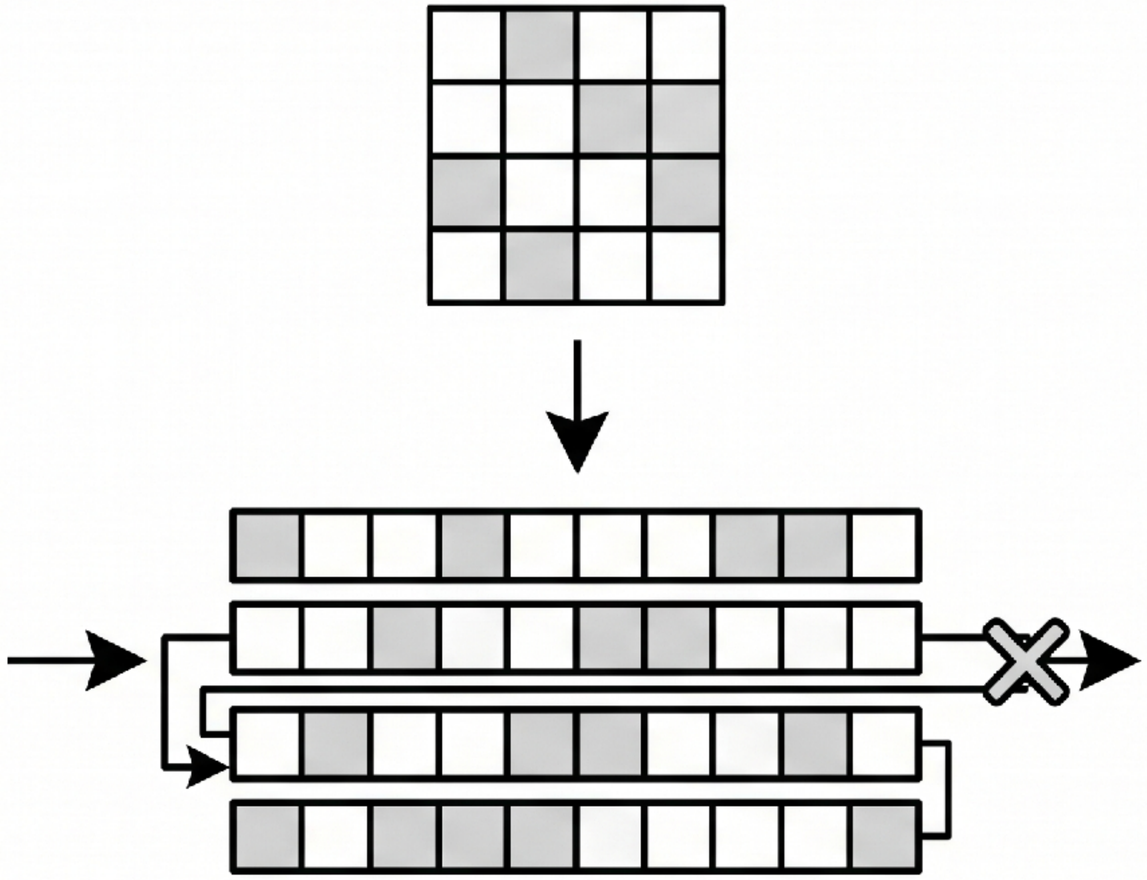
Sensitivity (Impact)



Cut weights with least impact on output
(sensitivity/Hessian-based intuition)

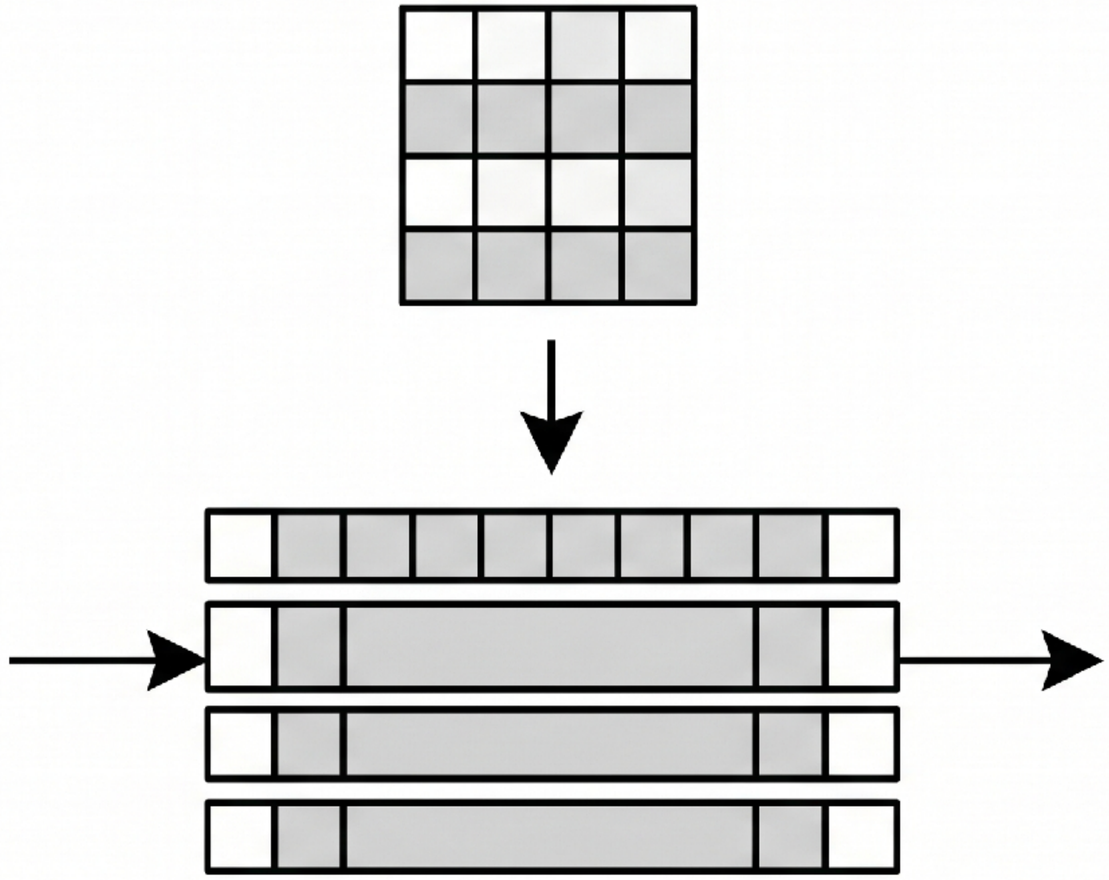
Structural Impact on Memory

Unstructured Pruning



Hard to Accelerate (Random memory access)

Structured Pruning

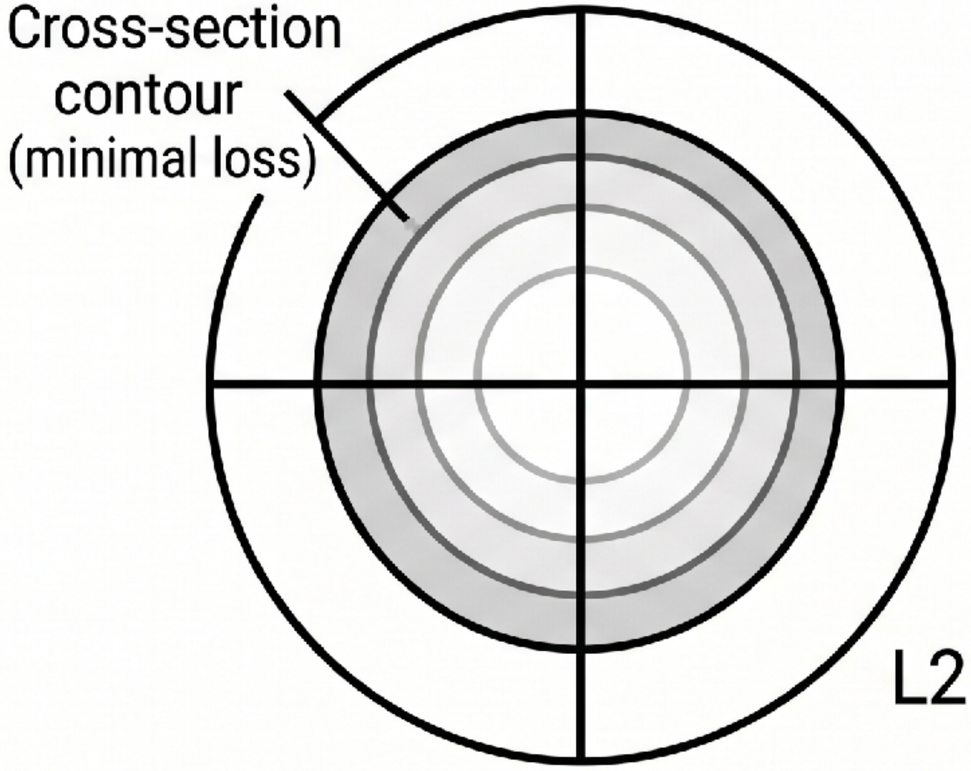


Easy to Accelerate (Skips dense blocks)

How to make pruning easier?

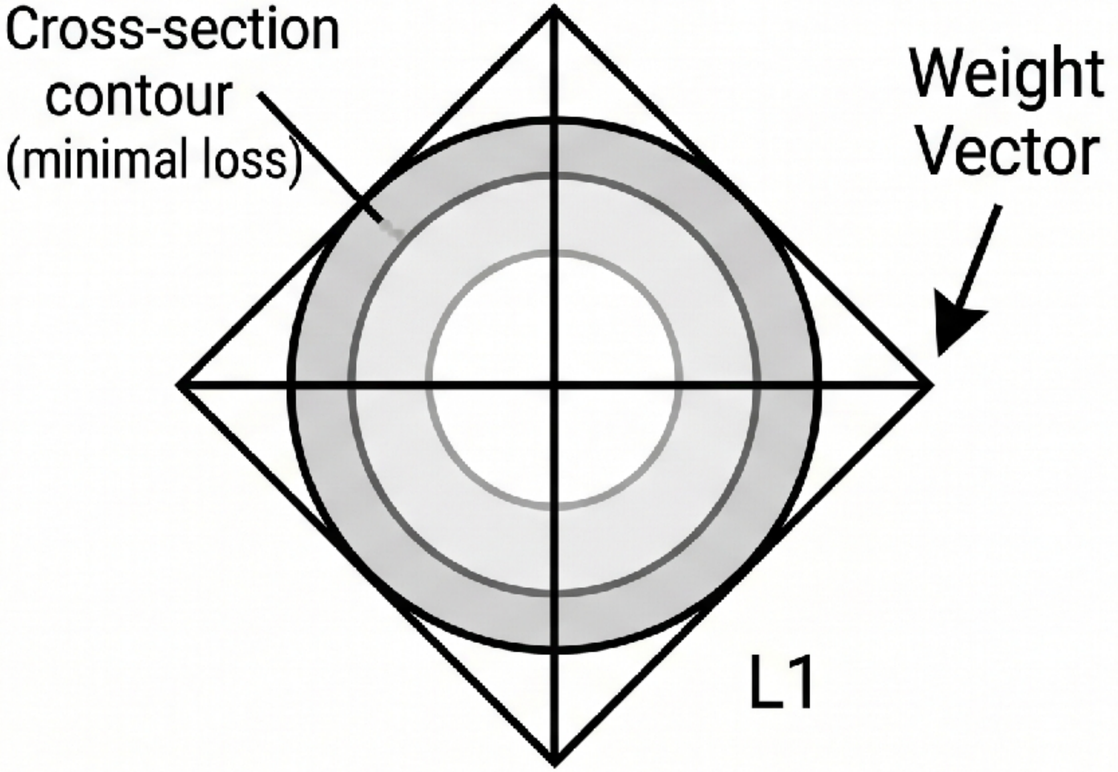
L1 Regularization as a Pruning Tool (Intuition)

L2 Regularization (Weight Decay)



Shrinks large weights, keeps all non-zero.

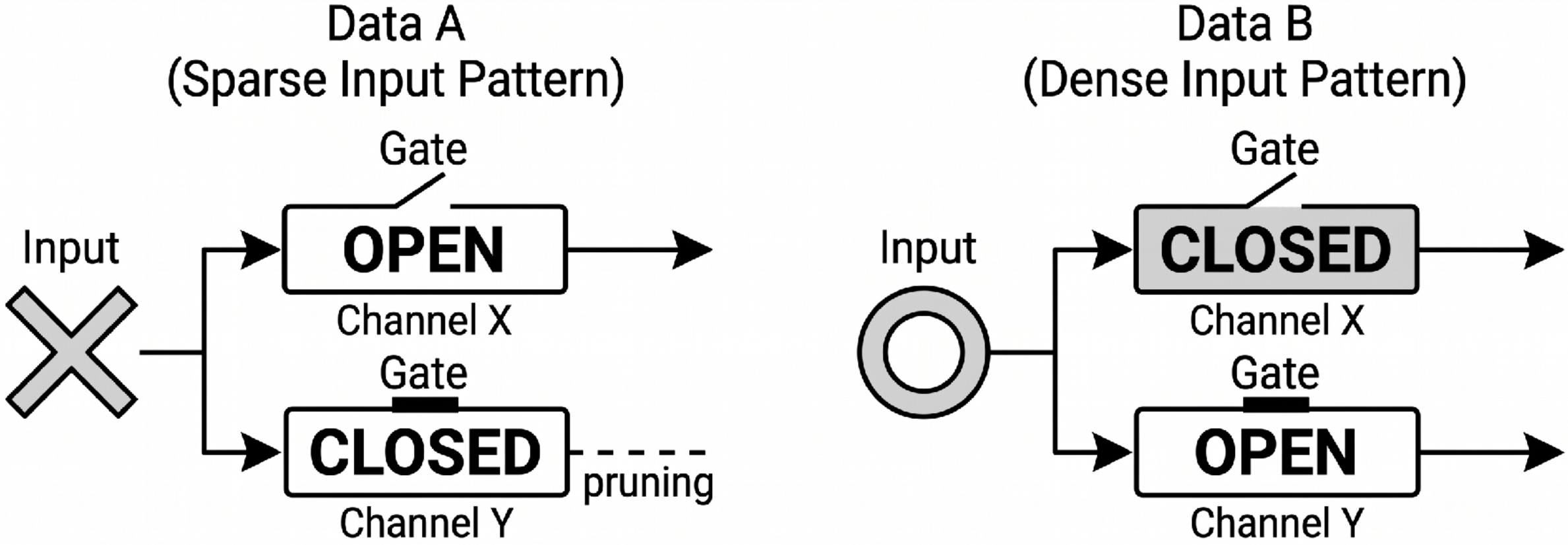
L1 Regularization (Sparse Learning)



Pushes weights exactly to zero, learns sparsity.

Dynamic/Activation-Based Gating (Channel-by-Input)

(dynamic pruning is not fixed)



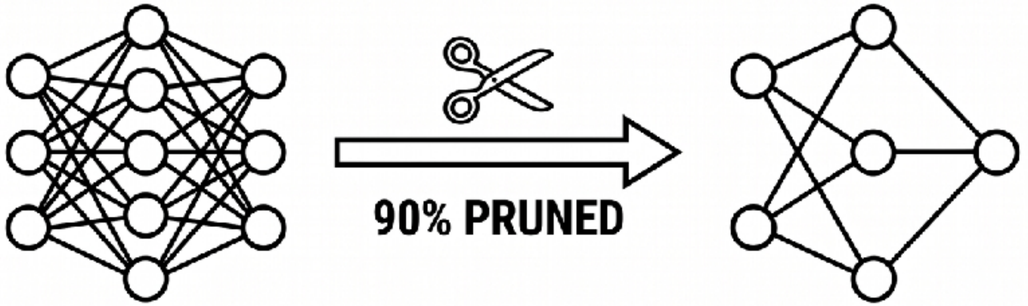
Channel Y inactive for Data A, dynamically pruned.

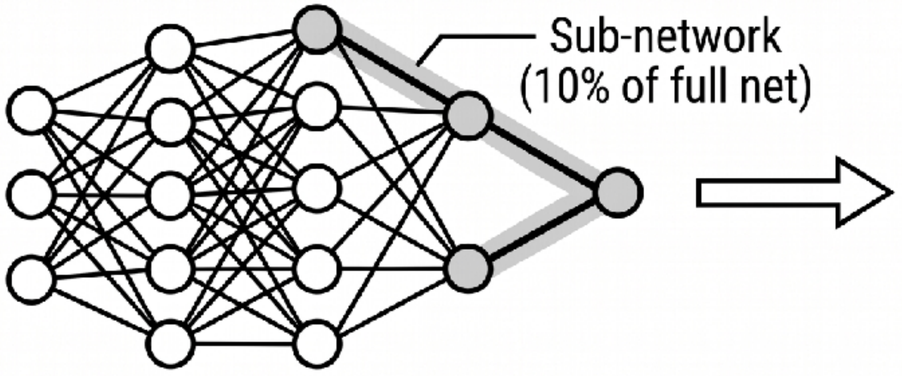
Channel X inactive for Data B, dynamically pruned.

THE LOTTERY TICKET HYPOTHESIS.

I. THE FINDING (DISCOVERY)

1. Start: **LARGE, DENSE, RANDOMLY INITIALIZED NETWORK.**
2. Process: **ITERATIVE PRUNING.**
3. End: **SMALL, SPARSE SUB-NETWORK**
   (with Original Initial Weights).





II. THE HYPOTHESIS & ITS PROPERTIES

- **CORE HYPOTHESIS**:
  A randomly-initialized network contains a "**Winning Ticket**": a sparse sub-network that can, when trained in isolation, match or exceed the original network's performance.
- **KEY PROPERTIES** (for the Winning Ticket):

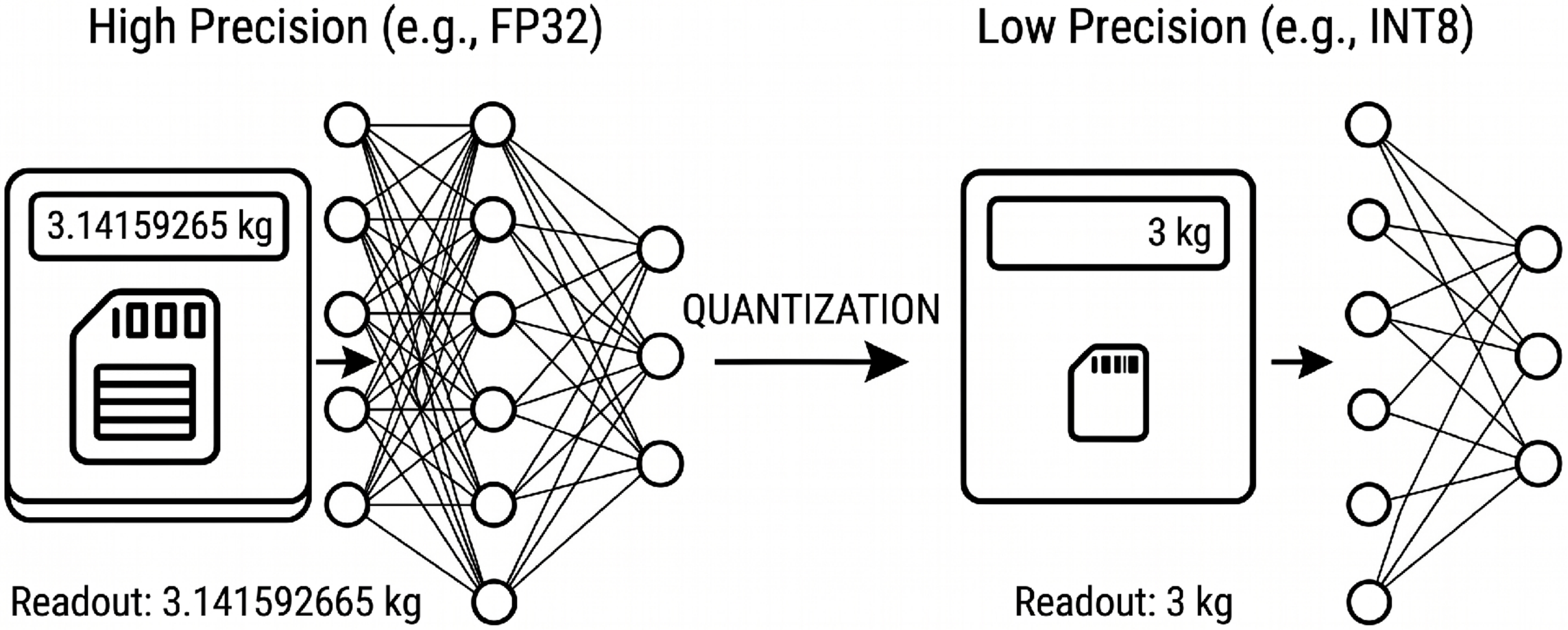
1. **SIZE:**
   Much smaller (<10-20%)
2. **TRAINABILITY:**
   Trains effectively.
3. **INITIALIZATION:**
   Crucially uses original initial weights.
   (Random re-init fails).

TRAINED WINNING TICKET		ORIGINAL DENSE NETWORK
✓	VS.	
ACCURACY: HIGH		**ACCURACY: HIGH**
SIZE: SMALL		**SIZE: LARGE**

TAKEAWAY: Pruning is key. You don't need to train a large network if you find the right small one.

Any other way to speed up, save memory?

What is Quantization? (The Intuition)

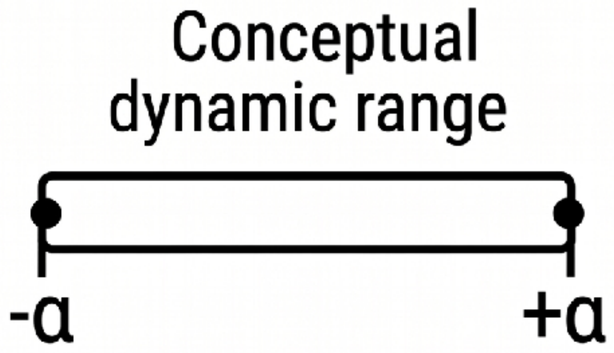


Reduce precision to save memory and speed.

How would you quantize weights?

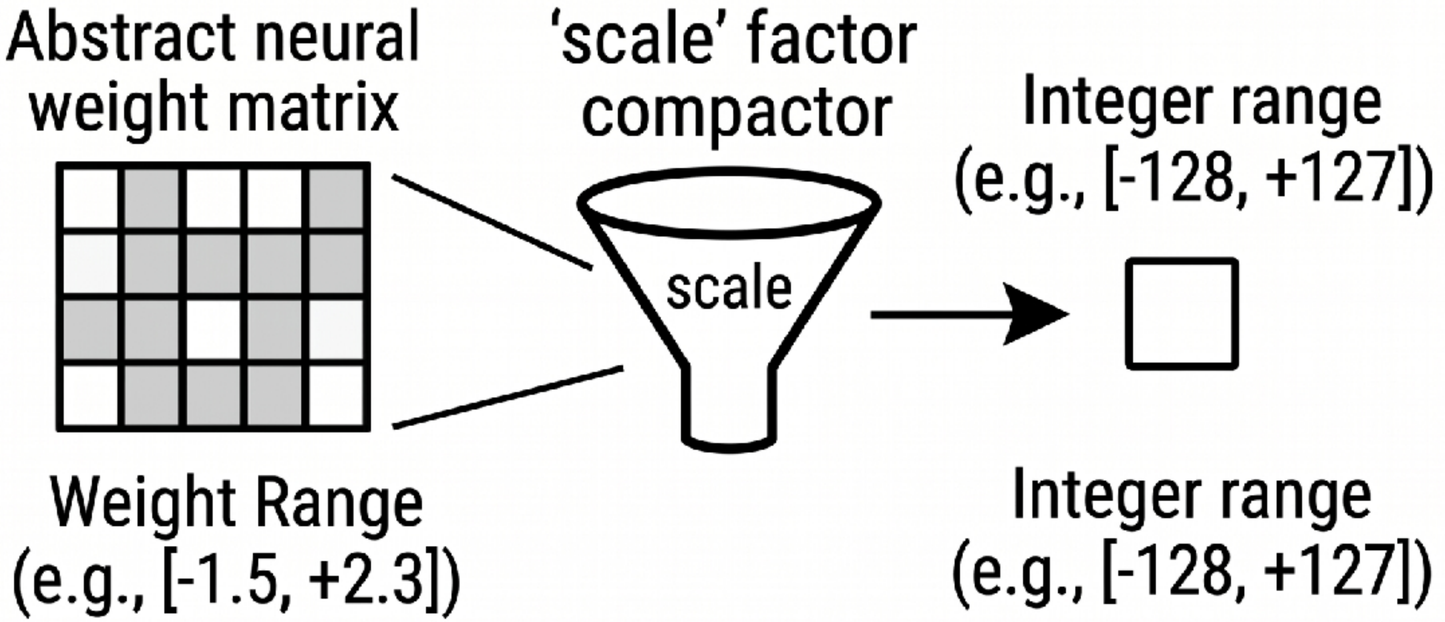
Basic Symmetric Uniform Quantization

Step 1: Determine Dynamic Range



Find min/max of weights/inputs

Step 2: Linear Mapping (e.g., to INT8)



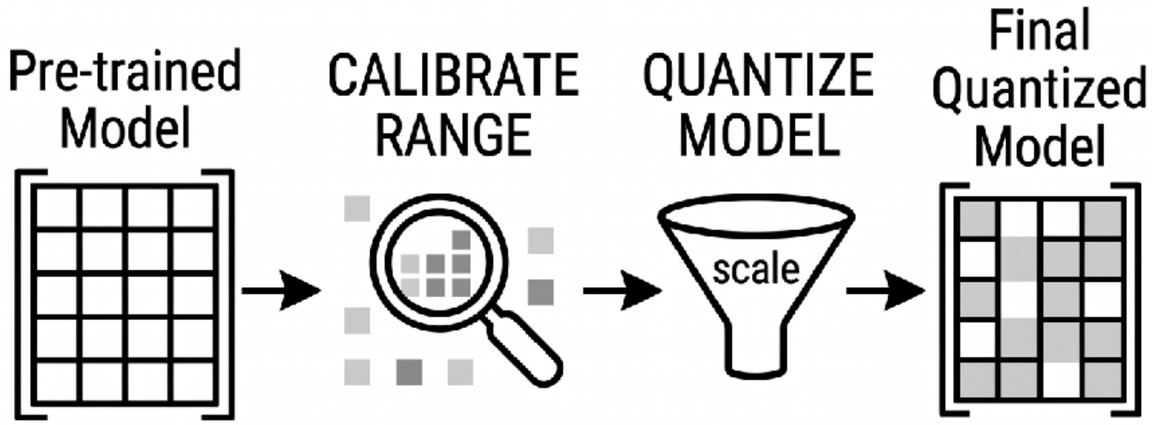
$$Q(x) = \text{round}(x/\text{scale})$$

Uses symmetric range of float inputs.

What happens to model performance?

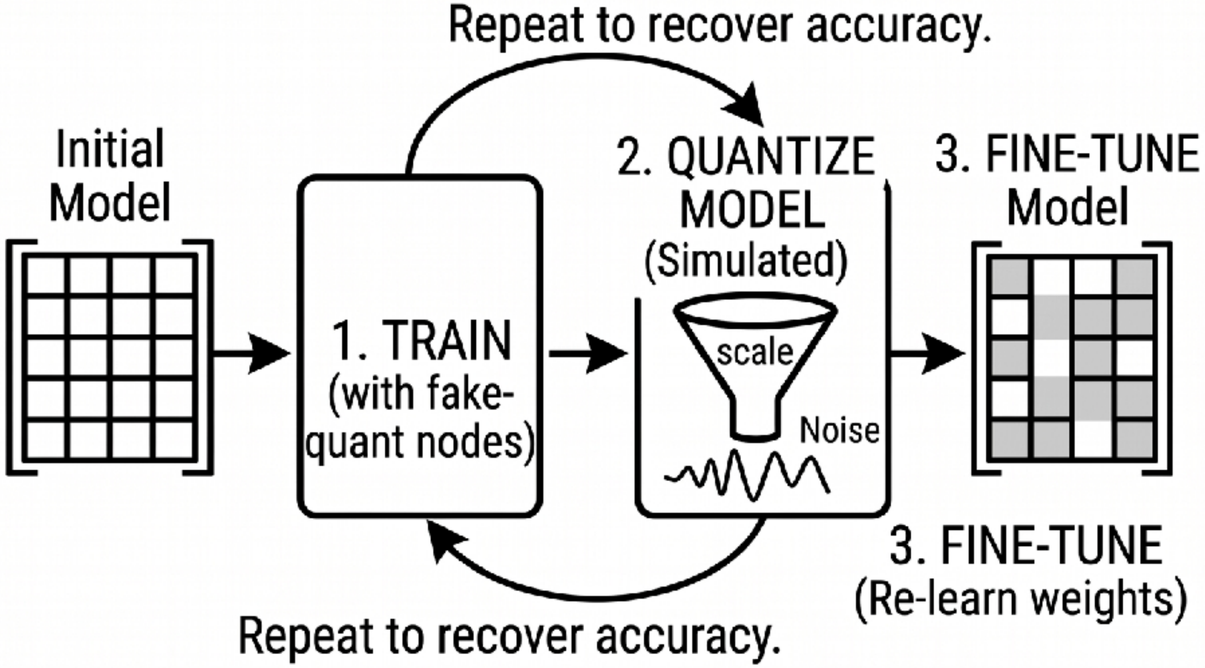
Post-Training Quantization vs. Quantization-Aware Training (QAT)

POST-TRAINING QUANTIZATION (Off-the-shelf)



Uses pre-trained model.
Minimal complexity.
Lower accuracy.

QUANTIZATION-AWARE TRAINING (QAT)



Simulates quantization during training.
Re-learns weight updates.
Higher accuracy, more effort.

Integrating Fairness and Model Pruning Through Bi-level Optimization

Yucong Dai*, Gen Li*, Feng Luo, Xiaolong Ma†, Yongkai Wu†

Clemson University {yucongd,gen,luofeng,xiaolom,yongkaw}@clemson.edu
*Equal contribution †Corresponding author

Less Is More? Examining Fairness in Pruned Large Language Models for Summarising Opinions

Nannan Huang
RMIT University, Australia
amber.huang@student.rmit.edu.au

Haytham M. Fayek
RMIT University, Australia
haytham.fayek@ieee.org

Xiuzhen Zhang
RMIT University, Australia
xiuzhen.zhang@rmit.edu.au

Preserving Fairness and Safety in Quantized LLMs Through Critical Weight Protection

Muhammad Alif Al Hakim[1], **Alfan Farizki Wicaksono**[1], **Fajri Koto**[2]
[1]Universitas Indonesia [2]MBZUAI
malif.al@ui.ac.id

Understanding the Unfairness in Network Quantization

Bing Liu [1 2 3] **Wenjun Miao** [1] **Boyu Zhang** [1] **Qiankun Zhang** ✉ [1 2 3] **Bin Yuan** [1 3 4 5] **Jing Wang** [6] **Shenghao Liu** [1]
Xianjun Deng [1]

torch.nn.utils.prune.ln_structured

torch.nn.utils.prune.ln_structured(*module*, *name*, *amount*, *n*, *dim*, *importance_scores=None*) [source]

Prune tensor by removing channels with the lowest L`n`-norm along the specified dimension.

Prunes tensor corresponding to parameter called `name` in `module` by removing the specified `amount` of (currently unpruned) channels along the specified `dim` with the lowest L`n`-norm. Modifies module in place (and also return the modified module) by:

1. adding a named buffer called `name+'_mask'` corresponding to the binary mask applied to the parameter `name` by the pruning method.
2. replacing the parameter `name` by its pruned version, while the original (unpruned) parameter is stored in a new parameter named `name+'_orig'`.

```
from lightning.pytorch.callbacks import ModelPruning

# set the amount to be the fraction of parameters to prune
trainer = Trainer(callbacks=[ModelPruning("l1_unstructured", amount=0.5)])
```

Thank you!
See you Wednesday!