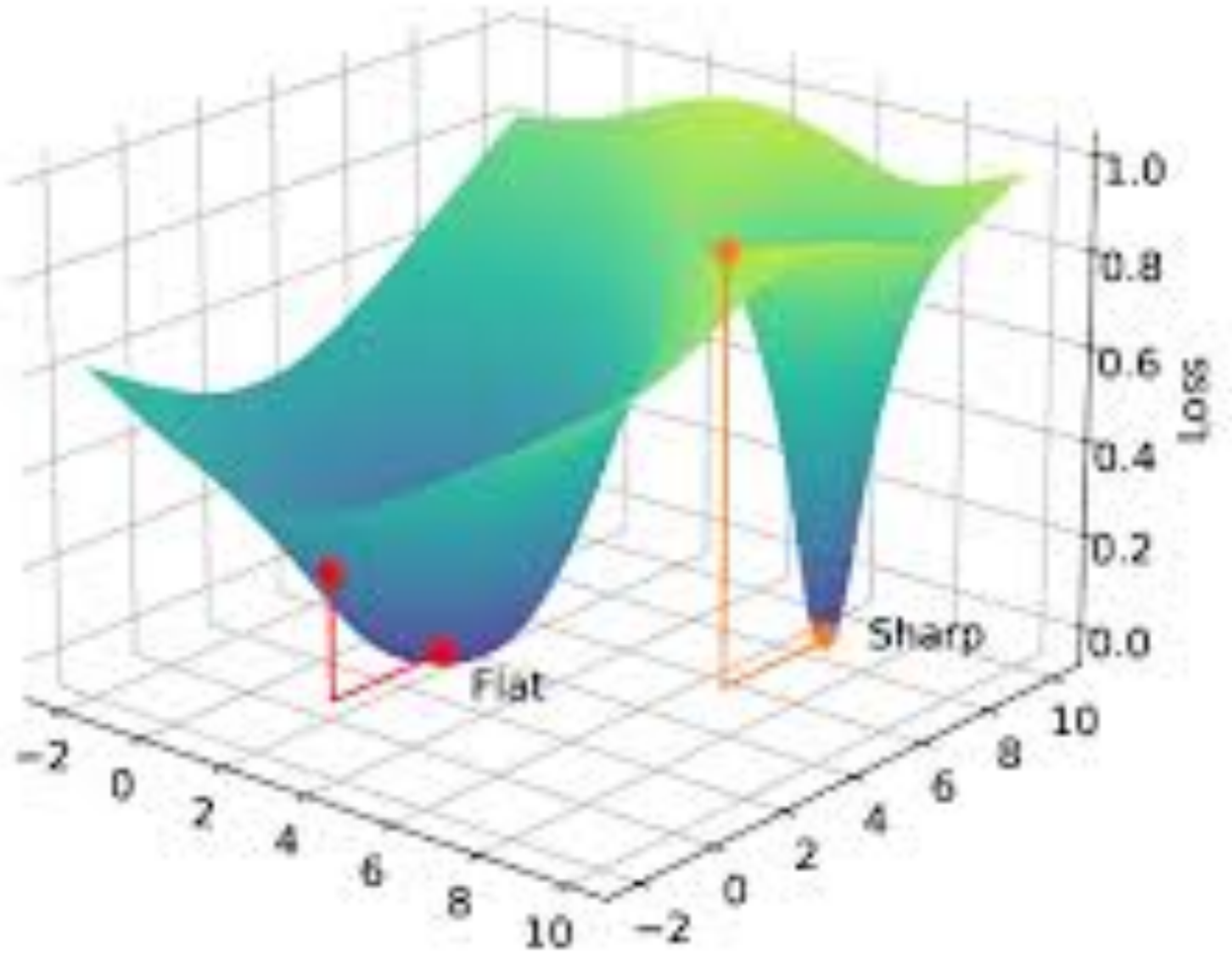
CSCI1470

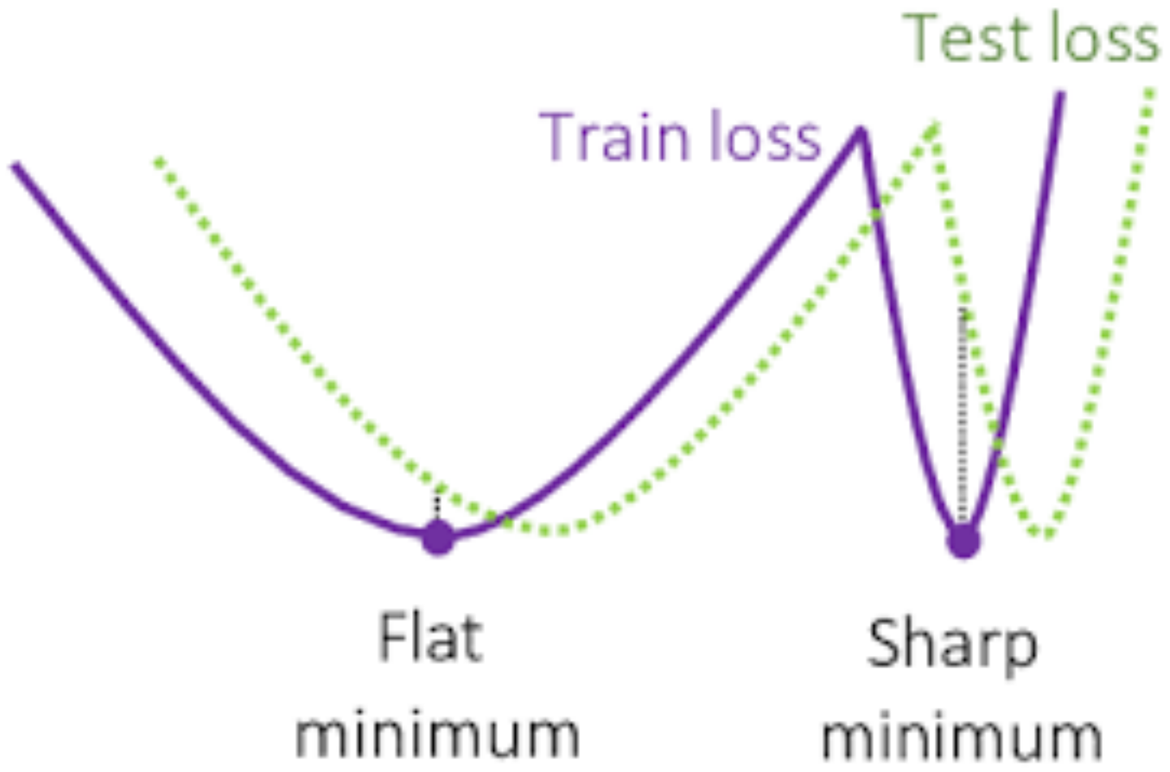
Deep Learning

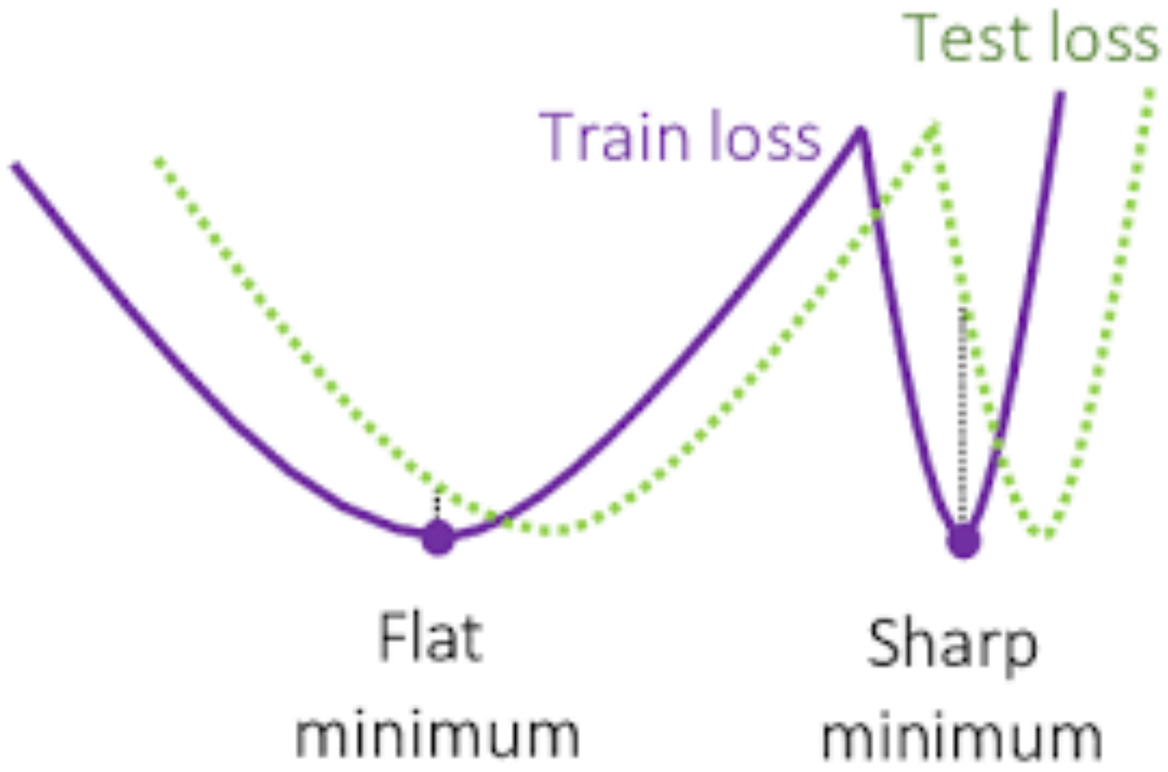
Randall Balestriero

Recap

Optimization for Better Generalization?


Flat
Sharp
Loss
1.0
0.8
0.6
0.4
0.2
0.0
−2
0
2
4
6
8
10


Test loss
Train loss
Flat
minimum
Sharp
minimum



Any idea?

The classic optimization framework of machine learning is empirical risk minimization

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{x}_i; \boldsymbol{\theta}) \tag{1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is a vector of parameters, $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ is a training set of inputs $\boldsymbol{x}_n \in \mathbb{R}^D$, and $\ell(\boldsymbol{x}; \boldsymbol{\theta})$ is a loss function quantifying the performance of parameters $\boldsymbol{\theta}$ on $\boldsymbol{x}$. SGD samples a minibatch $\mathcal{S} \subset \{1, \ldots, N\}$ of size $|\mathcal{S}| \ll N$ from the training set and updates the parameters through

$$\boldsymbol{\theta}_{t+1}^{\text{SGD}} = \boldsymbol{\theta}_t - \eta \boldsymbol{g}(\boldsymbol{\theta}_t), \text{ where } \boldsymbol{g}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}|}\sum_{i \in \mathcal{B}} \nabla \ell(\boldsymbol{\theta}; \boldsymbol{x}_i), \tag{2}$$

for a length specified by η, the learning rate.

SHARPNESS-AWARE MINIMIZATION FOR EFFICIENTLY IMPROVING GENERALIZATION

Pierre Foret *
Google Research
pierre.pforet@gmail.com

Ariel Kleiner
Google Research
akleiner@gmail.com

Hossein Mobahi
Google Research
hmobahi@google.com

Behnam Neyshabur
Blueshift, Alphabet
neyshabur@google.com

Aware Minimization (SAM) problem:

$$\min_{\boldsymbol{w}} L_S^{SAM}(\boldsymbol{w}) + \lambda ||\boldsymbol{w}||_2^2 \quad \text{where} \quad L_S^{SAM}(\boldsymbol{w}) \triangleq \max_{||\boldsymbol{\epsilon}||_p \leq \rho} L_S(\boldsymbol{w} + \boldsymbol{\epsilon}), \tag{1}$$

SHARPNESS-AWARE MINIMIZATION FOR EFFICIENTLY IMPROVING GENERALIZATION

Pierre Foret *
Google Research
pierre.pforet@gmail.com

Ariel Kleiner
Google Research
akleiner@gmail.com

Hossein Mobahi
Google Research
hmobahi@google.com

Behnam Neyshabur
Blueshift, Alphabet
neyshabur@google.com

Aware Minimization (SAM) problem:

$$\min_{\boldsymbol{w}} L_S^{SAM}(\boldsymbol{w}) + \lambda ||\boldsymbol{w}||_2^2 \quad \text{where} \quad L_S^{SAM}(\boldsymbol{w}) \triangleq \max_{||\boldsymbol{\epsilon}||_p \leq \rho} L_S(\boldsymbol{w} + \boldsymbol{\epsilon}), \tag{1}$$

Remind you of anything?

Averaging Weights Leads to Wider Optima and Better Generalization

Pavel Izmailov[*1] **Dmitrii Podoprikhin**[*2,3] **Timur Garipov**[*4,5] **Dmitry Vetrov**[2,3] **Andrew Gordon Wilson**[1]
[1]Cornell University, [2]Higher School of Economics, [3]Samsung-HSE Laboratory,
[4]Samsung AI Center in Moscow, [5]Lomonosov Moscow State University

Algorithm 1 Stochastic Weight Averaging [48]

Input: Loss function $\mathcal{L}$, training budget in number of iterations b, training dataset $\mathcal{D} := \cup_{i=1}^{n}\{\boldsymbol{x}_i\}$, mini-batch size $|\mathcal{B}|$, averaging start epoch E, averaging frequency ν, (scheduled) learning rate η, initial weights $\boldsymbol{\theta}_0$.

for $k \leftarrow 1, \ldots, b$ **do**
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$
  Compute gradient $\boldsymbol{g} \leftarrow \nabla\mathcal{L}\left(\boldsymbol{\theta}_t\right)$
  Update parameters $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta\boldsymbol{g}$
  if $k \geq E$ and $\mathrm{mod}(k, \nu) = 0$ **then**
    $\boldsymbol{\theta}_{t+1}^{\mathrm{SWA}} = \left(\boldsymbol{\theta}_t^{\mathrm{SWA}} \cdot l + \boldsymbol{\theta}_{t+1}^{\mathrm{SWA}}\right) / \left(l + 1\right)$
  end if
end for
return $\boldsymbol{\theta}^{\mathrm{SWA}}$

Algorithm 2 Sharpness-Aware Minimization [22]

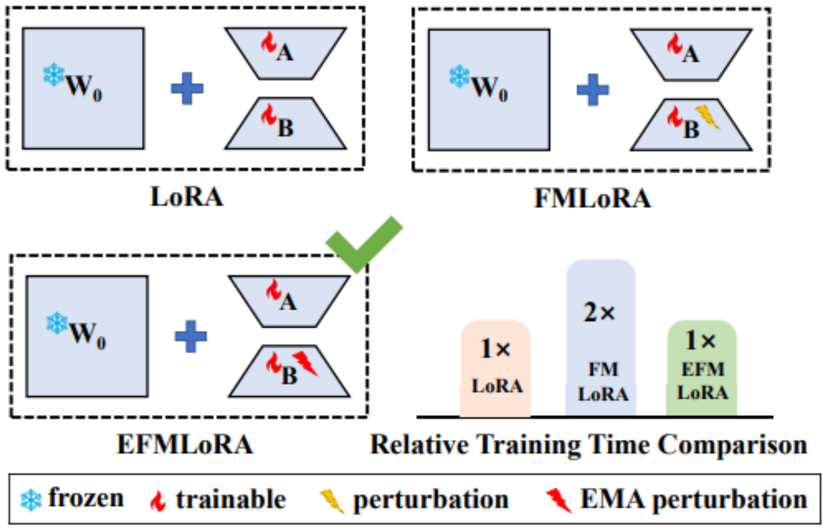
Input: Loss function $\mathcal{L}$, training budget in number of iterations b, training dataset $\mathcal{D} := \cup_{i=1}^{n}\{\boldsymbol{x}_i\}$, mini-batch size $|\mathcal{B}|$, neighborhood radius ρ, (scheduled) learning rate η, initial weights $\boldsymbol{\theta}_0$.

for $k \leftarrow 1, \ldots, b$ **do**
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$
  Compute worst-case perturbation $\widehat{\boldsymbol{\epsilon}} \leftarrow \rho \frac{\nabla\mathcal{L}(\boldsymbol{\theta})}{\|\nabla\mathcal{L}(\boldsymbol{\theta})\|_2}$
  Compute gradient $\boldsymbol{g} \leftarrow \nabla\mathcal{L}\left(\boldsymbol{\theta}_t^{\mathrm{SAM}} + \widehat{\boldsymbol{\epsilon}}\right)$
  Update parameters $\boldsymbol{\theta}_{t+1}^{\mathrm{SAM}} \leftarrow \boldsymbol{\theta}_t^{\mathrm{SAM}} - \eta\boldsymbol{g}$
end for
return $\boldsymbol{\theta}^{\mathrm{SAM}}$

Efficiently Seeking Flat Minima for Better Generalization in Fine-Tuning Large Language Models and Beyond

Jiaxin Deng[1*], Qingcheng Zhu[2*], Junbiao Pang[1†], Linlin Yang[3], Zhongqian Fu[4], Baochang Zhang[5‡]

[1]School of Information Science and Technology, Beijing University of Technology, Beijing, China
[2]School of Electronic Information Engineering , Beihang University, Beijing, China
[3]State Key Laboratory of Media Convergence and Communication, Communication University of China, Beijing, China
[4]Huawei Noah's Ark Lab, China
[5]Hangzhou Research Institute, School of Artificial Intelligence, Beihang University, China
dengjiaxin@emails.bjut.edu.cn, zhuqc@buaa.edu.cn, junbiao_pang@bjut.edu.cn, lyang@cuc.edu.cn, fuzhongqian@huawei.com, bczhang@buaa.edu.cn



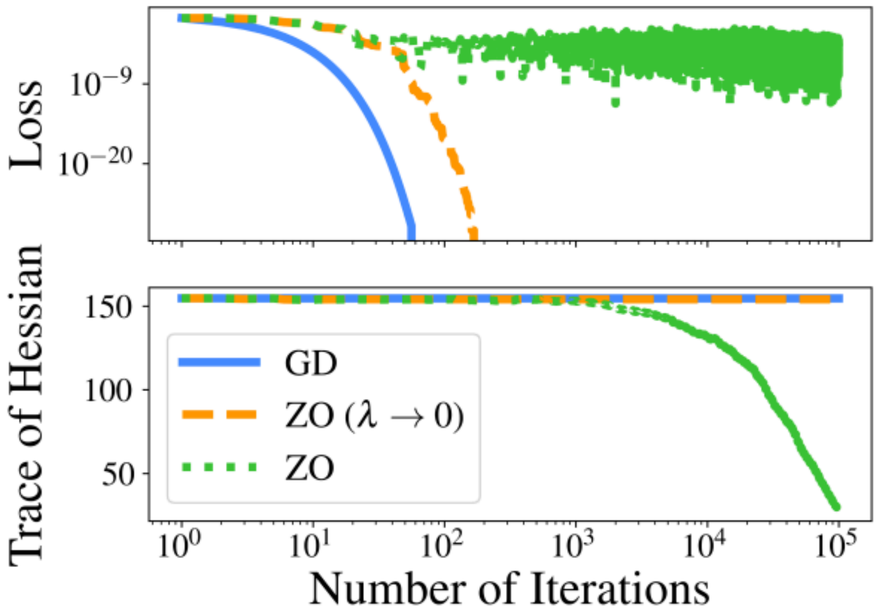
Zeroth-Order Optimization Finds Flat Minima

Liang Zhang[1,2], Bingcong Li[1], Kiran Koshy Thekumparampil[3],
Sewoong Oh[4], Michael Muehlebach[2], and Niao He[1]

To be more specific, for the optimization problem $\min_{x\in\mathbb{R}^d} f(x)$ with parameters $x \in \mathbb{R}^d$ and a loss function $f : \mathbb{R}^d \to \mathbb{R}$, zeroth-order optimization with the standard two-point gradient estimator [74] (Algorithm 1) iteratively updates x by substituting the computationally intractable gradient with

$$g_\lambda(x, u) := \frac{f(x + \lambda u) - f(x - \lambda u)}{2\lambda} u, \tag{1}$$

where $u \sim \mathcal{N}(0, \mathrm{I}_d)$ is a standard Gaussian random vector and $\lambda \in \mathbb{R}$ is a smoothing parameter. It



(a) Comparison of GD and ZO.

Switching to deployment!

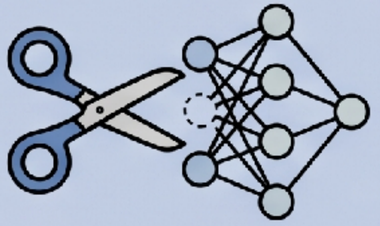
What do we want to optimize for?

What do we want to optimize for?

- Latency
- Memory
- Hardware

OPTIMIZING A TRAINED MODEL FOR FAST & EFFICIENT DEPLOYMENT

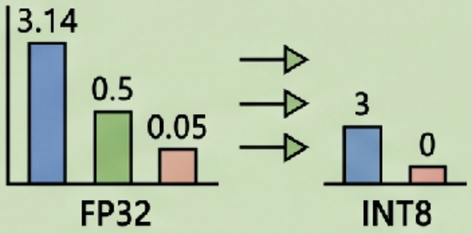
1. PRUNING



Reduce model size and computation by removing redundant weights and neurons.

- Remove connections with low contribution
- Creates a sparse model
- Achieves smaller memory footprint

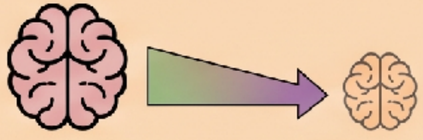
2. QUANTIZATION



Reduce precision of weights and activations with minimal accuracy loss.

- Convert from FP32 to smaller types (INT8, FP16)
- Significantly increases inference speed on compatible hardware
- Smaller memory usage

3. KNOWLEDGE DISTILLATION



Train a compact 'Student' model to mimic the performance of a large, complex 'Teacher' model.

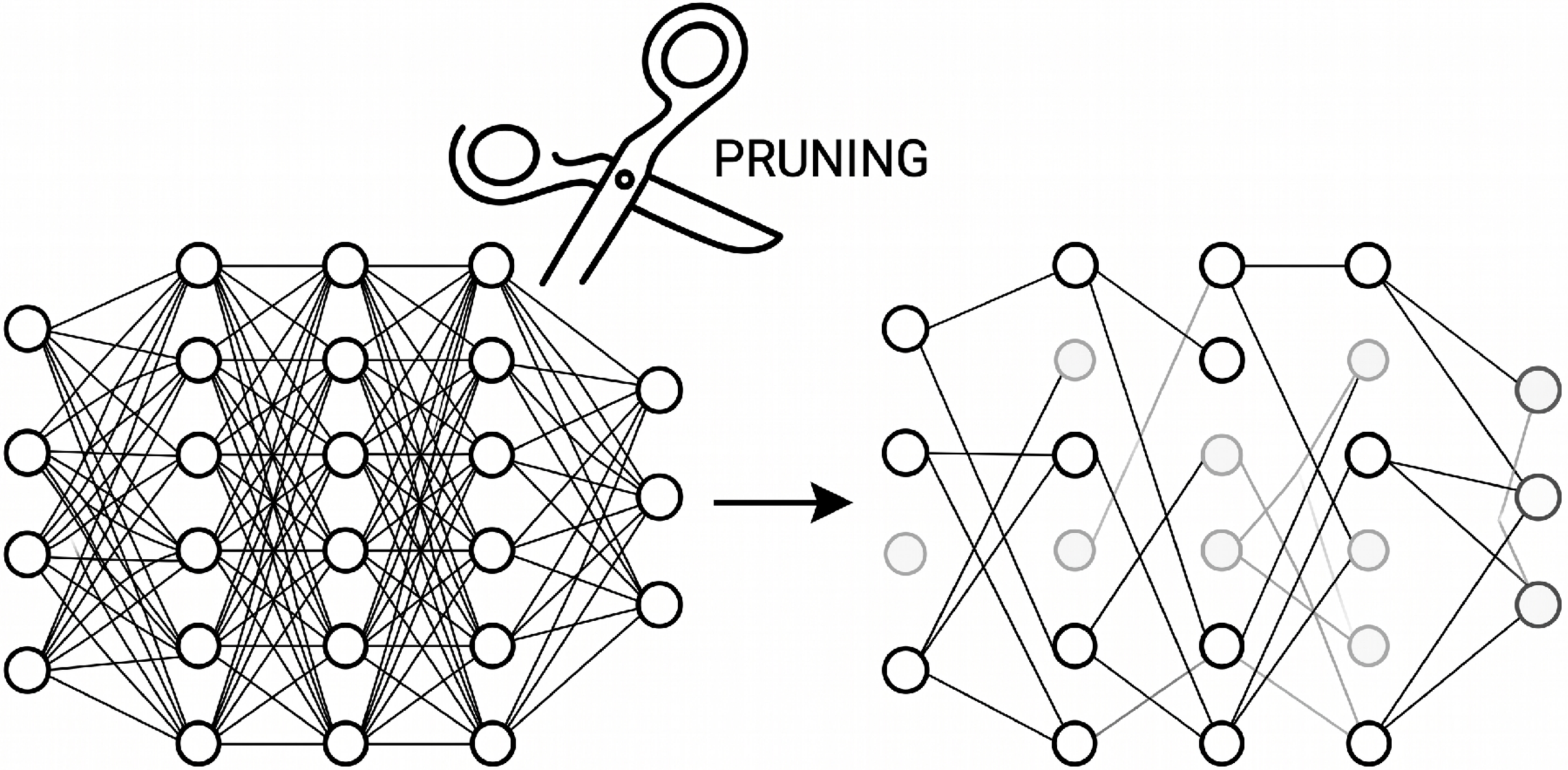
- Student model learns output distributions and representations
- Achieves comparable accuracy with much lower latency
- Ideal for edge devices

4. COMPILATION (e.g., TENSORRT)

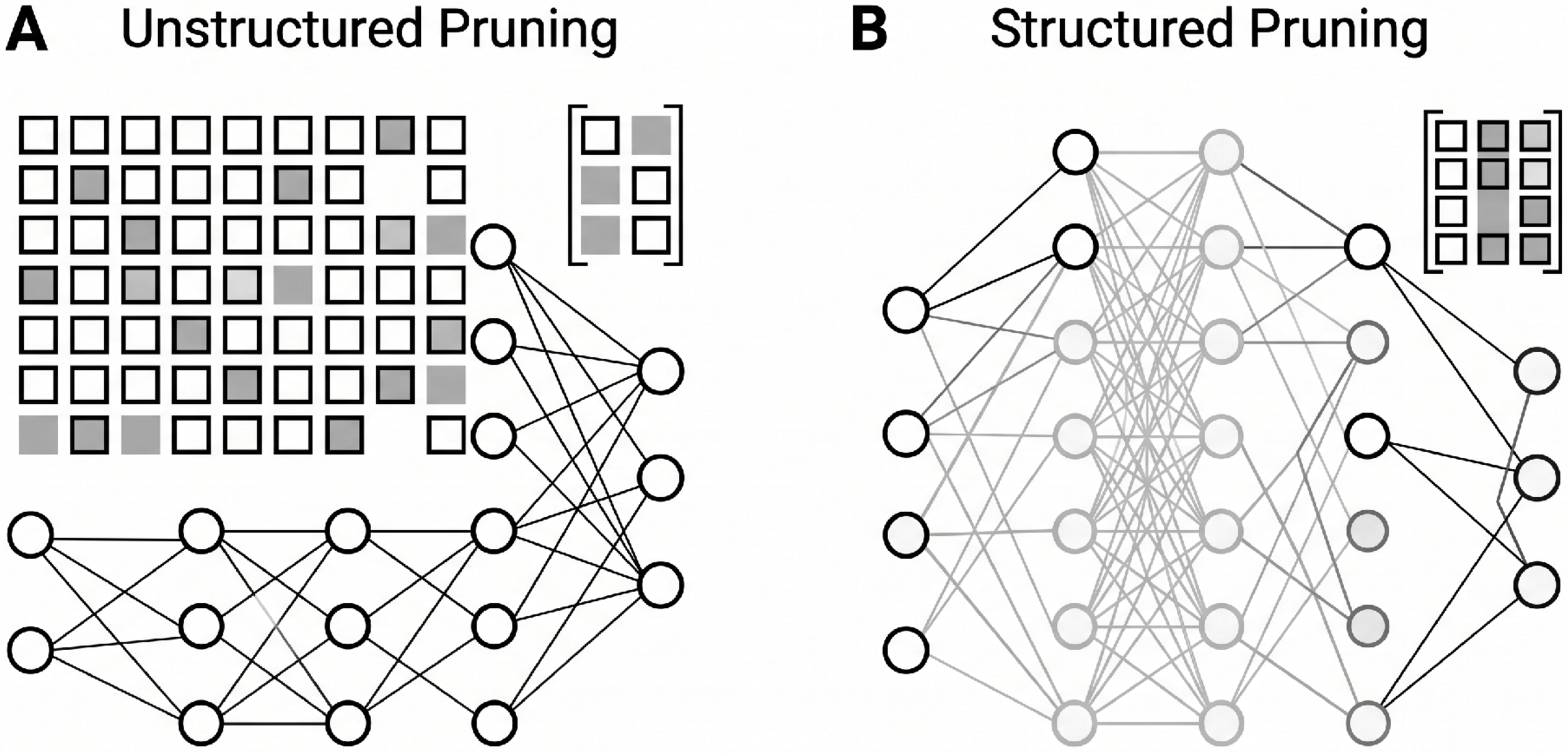
TensorRT
OpenVINO
ONNX RUNTIME

Apply a hardware-specific optimization compiler for maximum performance.

- Layer fusion, constant folding, and precision selection
- Tailored execution for target GPUs, CPUs, or NPUs
- Directly optimizes for target deployment environment



What is Model Pruning? (Simplifying the Model)


A
Unstructured Pruning
B
Structured Pruning
Remove individual weights anywhere
(matrix becomes sparse)
Remove entire neurons, channels, or layers
(retains structured connectivity)

Thank you!
See you Wednesday!