Model Compression for LLMs

Tanmoy Chakraborty
Associate Professor, IIT Delhi
https://tanmoychak.com/

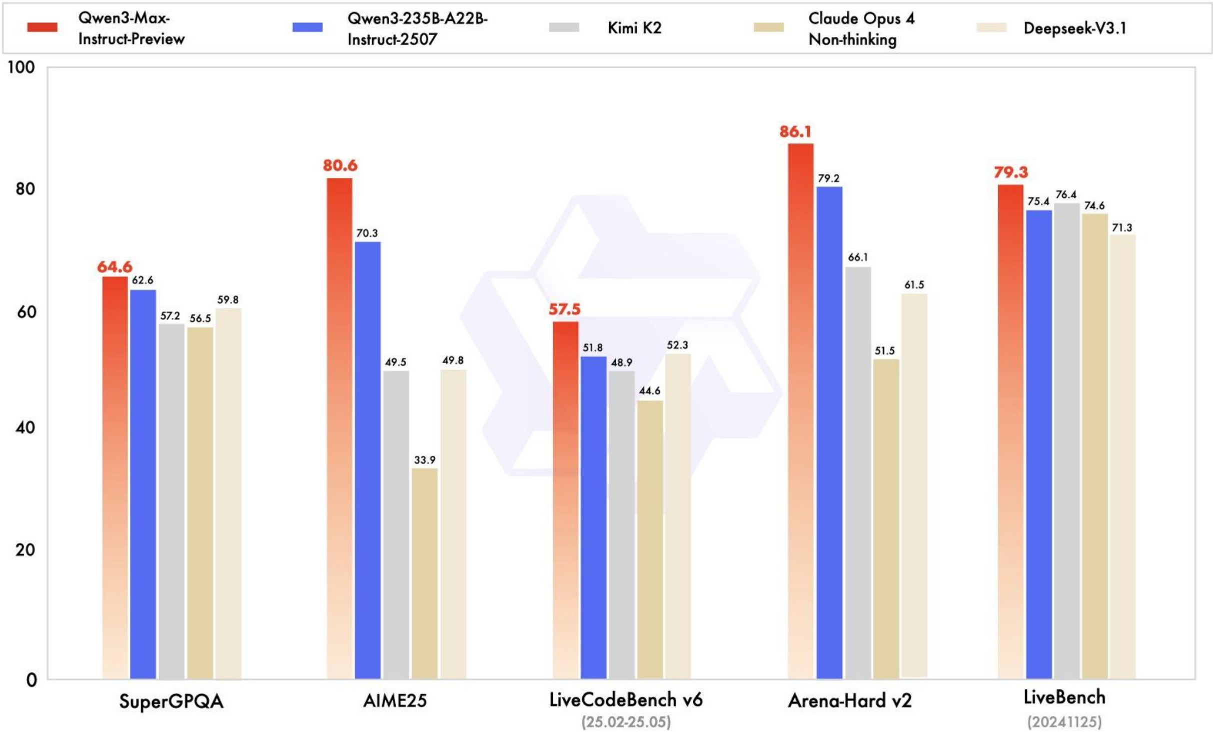
Qwen-3-Max-Preview

Qwen-3-Max is the company's first model to have more than 1 trillion parameters

Announced on
September 5, 2025

Qwen-3-Max Blog

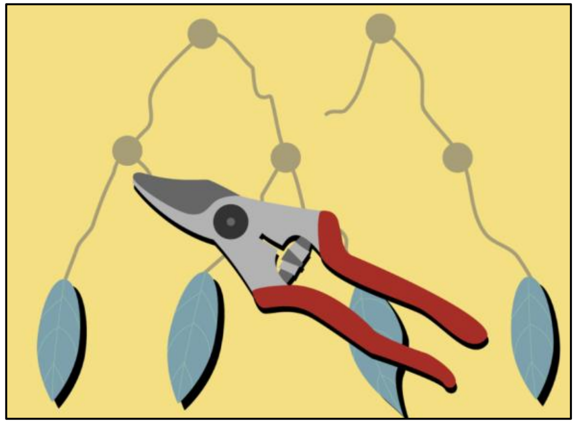
Qwen-3-Max-Preview supports 256k tokens context window along with 100+ languages with outstanding Chinese-English understanding.



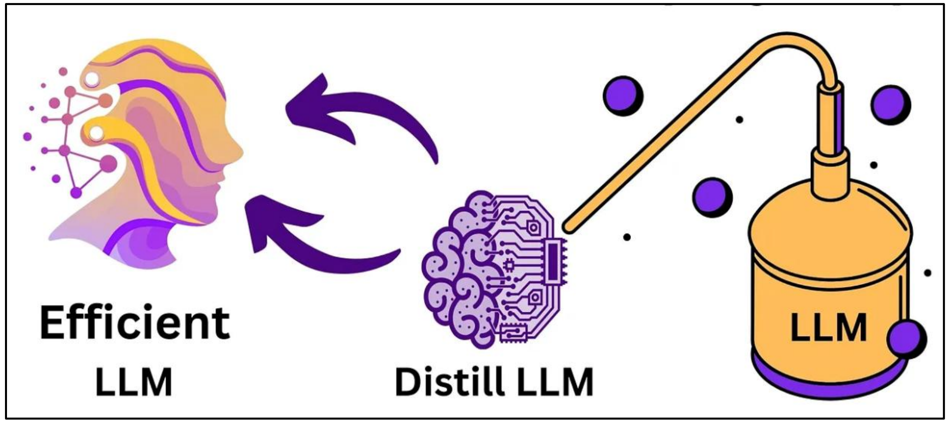
Qwen-3-Max-Preview achieves significant improvements in Reasoning abilities, Instruction Following, Multilingual processing and Reduced hallucinations. It shows great performance across other models especially on programming tasks.

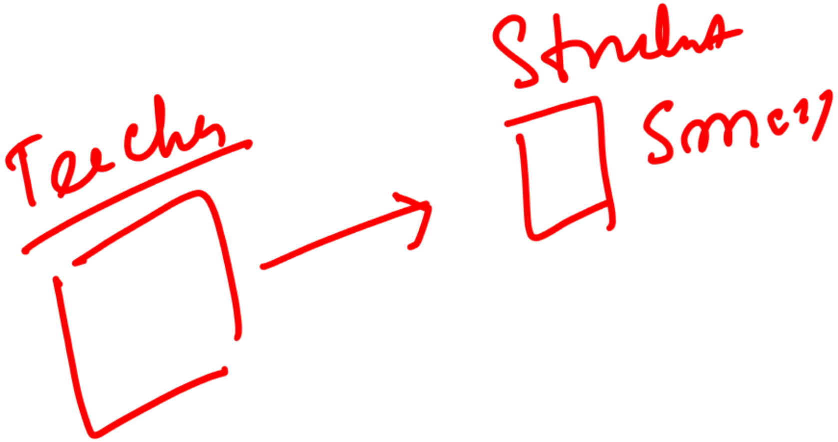
Model Compression

Model Pruning



Knowledge Distillation





Model Pruning

- **Basic Idea**: Among billions of parameters, some are bound to be less important than others
- Pruning involves removing weakly important parameters from pre-trained models

- **Lottery ticket hypothesis** (Frankle et al. 2019) suggests that a subnetwork exists for every neural networks, which when trained in isolation, reach test accuracy comparable to the original model.

- Identifying the winning ticket (subnetwork) is crucial, and can be derived by pruning a pre-trained network.

Why Pruning?

- Pruning interacts with compute vs performance tradeoff
- Pruning highlights which weights/layers encode critical knowledge, helping in model interpretation
- ✓ Moderate pruning sometimes improves generalization
- **Biological Analogy**: Synaptic pruning in the brain inspires efficient architectures.

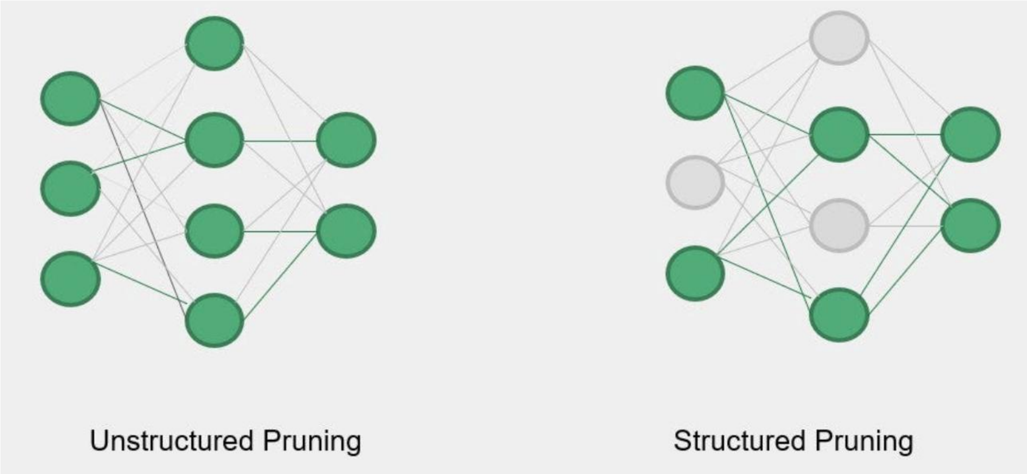


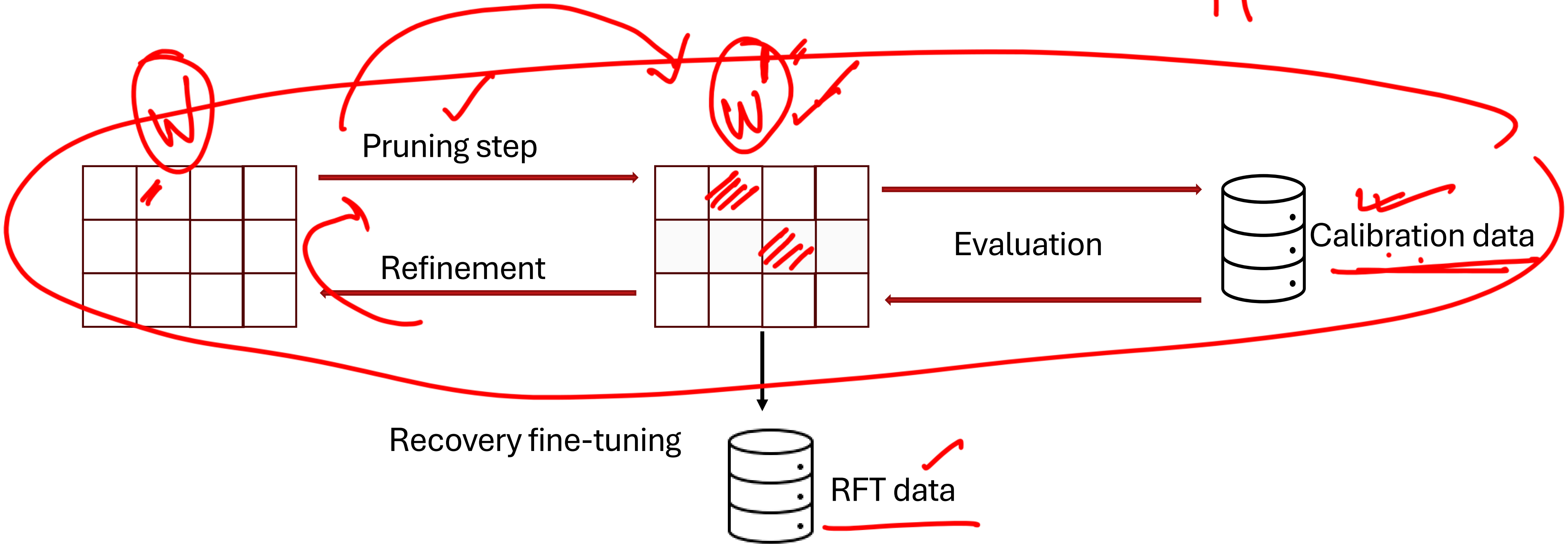
Image source: https://towardsdatascience.com/iterative-pruning-methods-for-artificial-neural-networks-in-julia-c605f547a485/

Trade-offs with Model Pruning

- **Accuracy vs. Efficiency**: Higher pruning saves compute but risks accuracy drops
- **Dependence on Hardware Type:** Several pruning strategies, particularly the unstructured ones depend heavily on hardware configurations for scaling up
- **Generalization**: Pruned models may generalize differently across domains.
- **Fine-tuning Needs**: Some pruning requires extra fine-tuning to recover performance.
- **Universality**: No single pruning strategy works best for all models or tasks.

Structure pruning requires calibration data

M W'

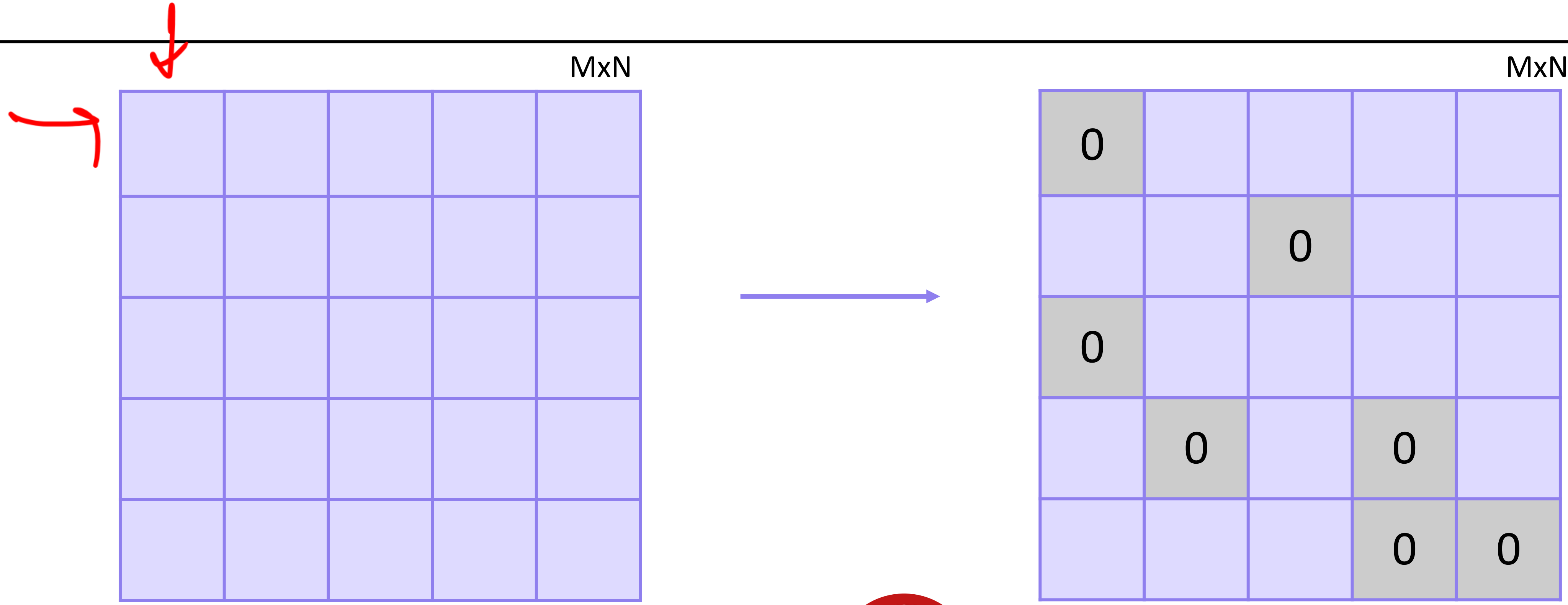


Pruning Techniques

- Two primary techniques to prune models:
  - Unstructured Pruning ✓
  - Structured Pruning ✓

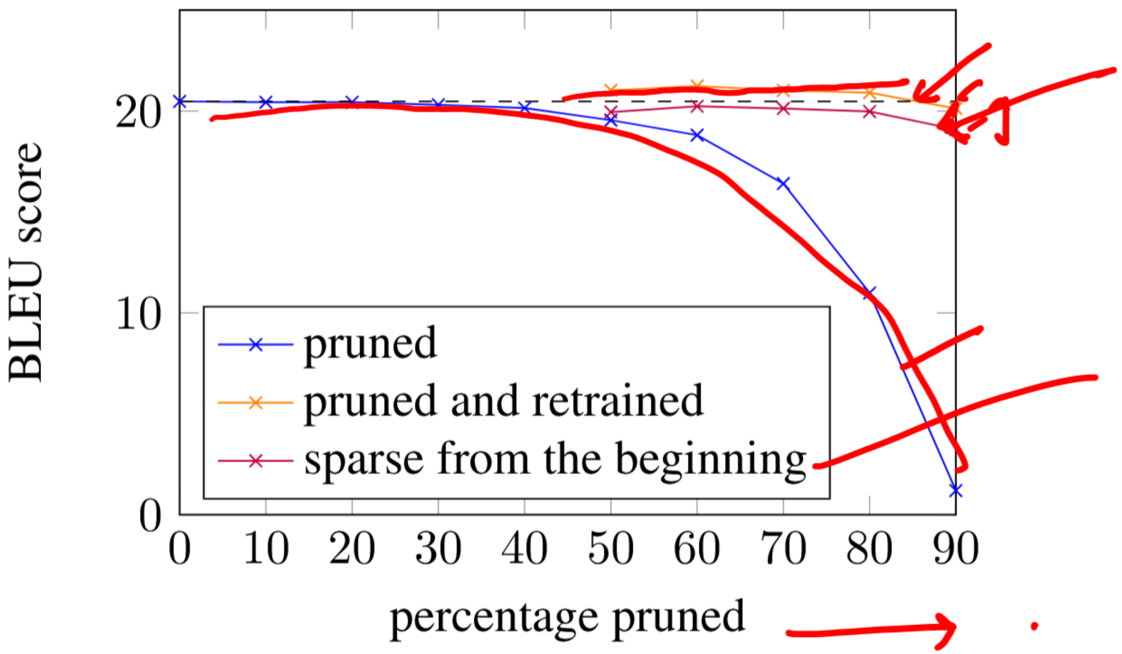
Unstructured Pruning

- Involves zeroing out individual model weights
- No “pattern” in which weights are pruned



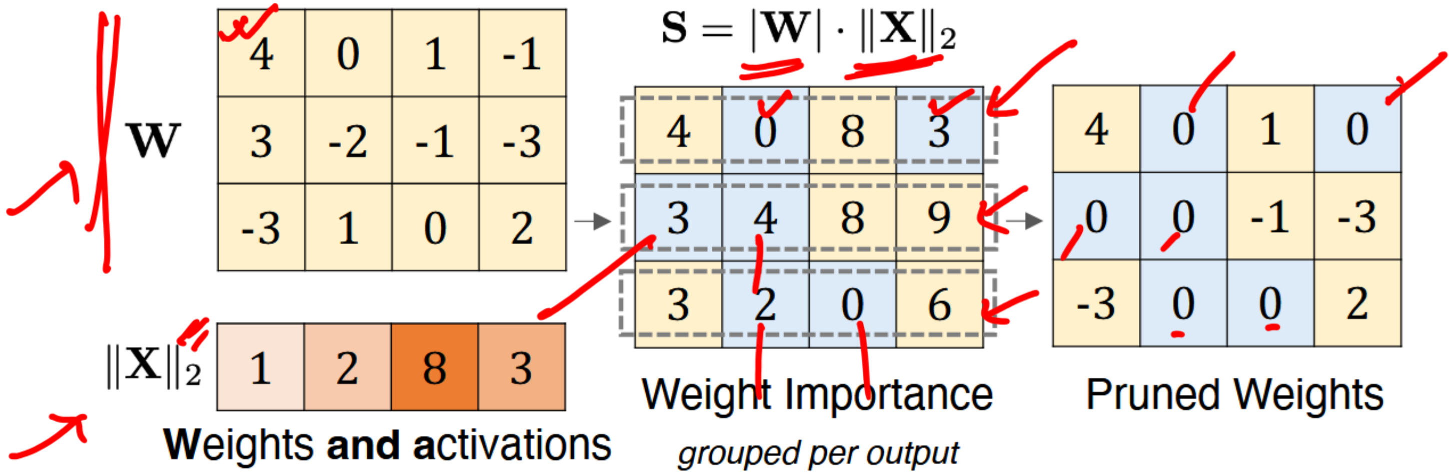
Magnitude Pruning (Han et al. 2015, See et al. 2016)

- **Idea**: Importance of weight ∝ Magnitude of weight
- x% of weights with smallest magnitudes are zeroed out



Wanda (Sun et al. 2023)

- Similar to magnitude pruning but with the following tweaks:
  - Scales individual weights (W) by respective normalized input activations (|X|)
  - Calculated row-wise & not for entire matrix at once
- Utilizes external data (*aka*. calibration data) as input activations
- No re-training required



$$\mathbf{W} = \begin{bmatrix} 4 & 0 & 1 & -1 \\ 3 & -2 & -1 & -3 \\ -3 & 1 & 0 & 2 \end{bmatrix}$$

$$\|\mathbf{X}\|_2 = \begin{bmatrix} 1 & 2 & 8 & 3 \end{bmatrix}$$

Weights and activations

$$\mathbf{S} = |\mathbf{W}| \cdot \|\mathbf{X}\|_2 = \begin{bmatrix} 4 & 0 & 8 & 3 \\ 3 & 4 & 8 & 9 \\ 3 & 2 & 0 & 6 \end{bmatrix}$$

Weight Importance

grouped per output

$$\begin{bmatrix} 4 & 0 & 1 & 0 \\ 0 & 0 & -1 & -3 \\ -3 & 0 & 0 & 2 \end{bmatrix}$$

Pruned Weights

x% pruning

SparseGPT (Franter et al. 2023)

$W_\ell X_\ell$

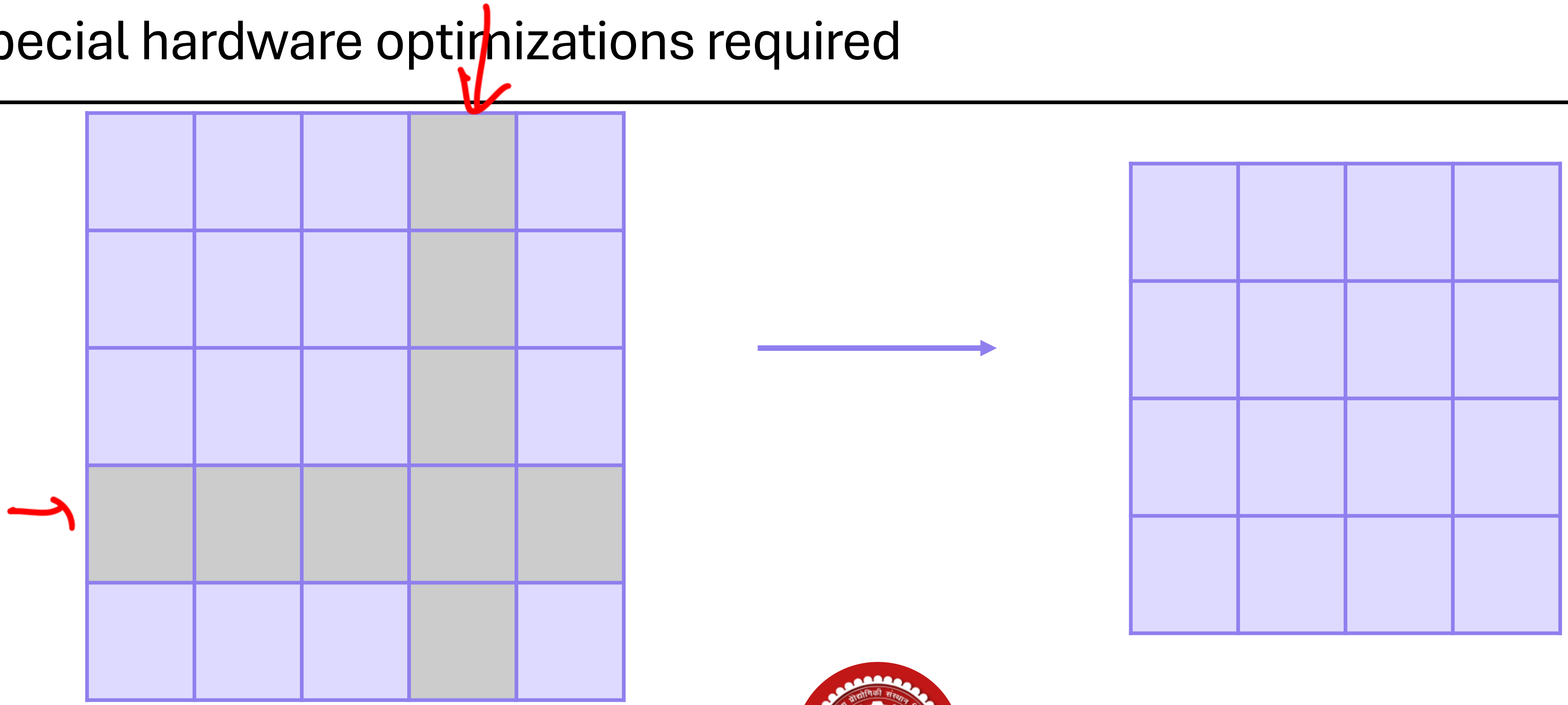
- One-shot pruning method, supporting both unstructured and semi-structured pruning
- Poses the problem of pruning as sparse regression problem, where the learning objective is $\text{argmin}_{\text{mask } \mathbf{M}_\ell, \widehat{\mathbf{W}}_\ell} ||\mathbf{W}_\ell \mathbf{X}_\ell - (\mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell)\mathbf{X}_\ell||_2^2$.
- For optimally solving the above learning objective, it uses a second-order Hessian approximation for learning the masking matrix M for each weight matrix W.

Main Disadvantage of Unstructured Pruning

- Leads to sparser weight matrices but does not reduce dimensional size
- Therefore, hardware optimizations are required to fully leverage sparsity & make computation more efficient

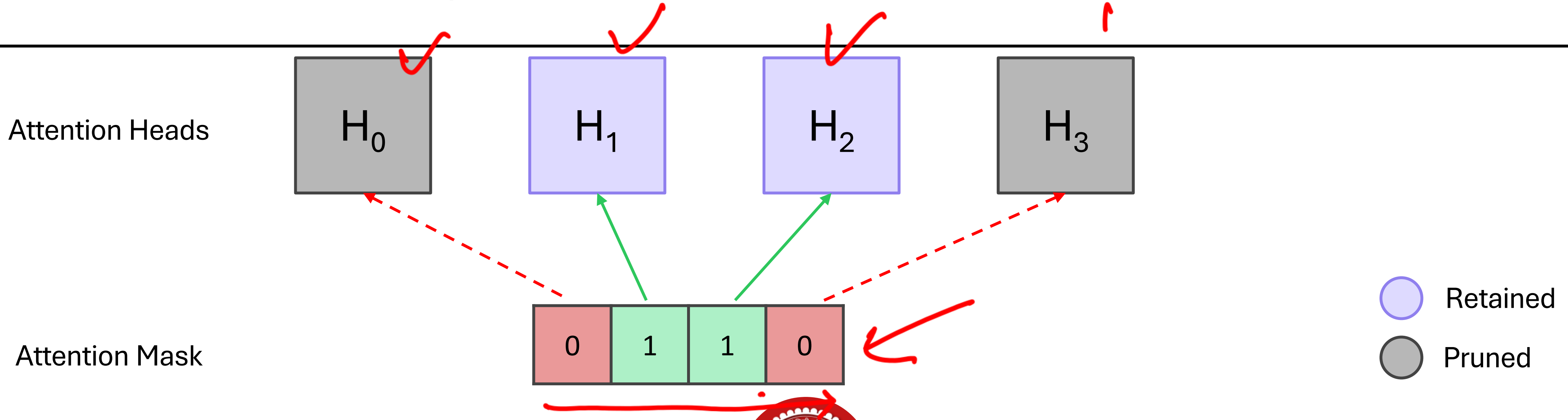
Structured Pruning

- Removes specific groups (structures) of weights such as neurons, weight channels, and even entire layers
- Leads to smaller weight matrices
- No special hardware optimizations required



Pruning BERT-Based Models (McCarley et al. 2019)

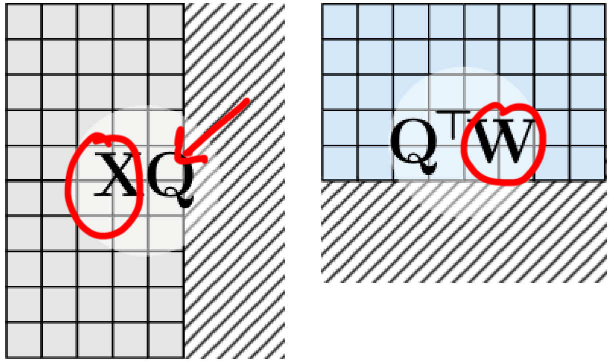
- Uses binary masks (vectors of trainable parameters) to determine which components of the model to prune
- 2 types of masks learnt for pruning:
  - Attention masks: Used to prune individual attention heads
  - FFN masks: Used to prune FFN slices



SliceGPT (Ashkboos et al. 2024)

- Instead of pruning individual weights, SliceGPT removes entire *rows and columns* from weight matrices (also known as *slicing)*. This directly reduces hidden dimension and embedding size.

- Before deleting, the model applies orthogonal rotations to weight matrices. These rotations keep outputs mathematically identical, so accuracy is preserved during slicing.



$X \quad W$

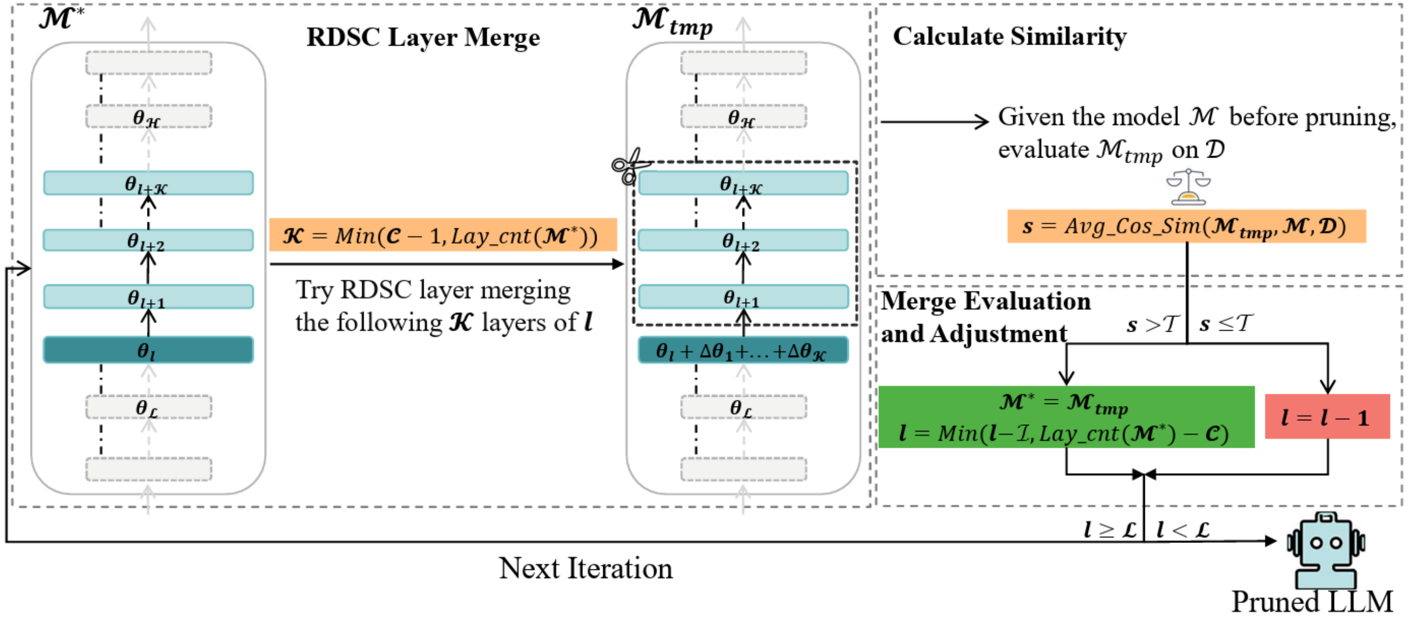
$XQ \times \underbrace{Q^{T}W}_{x}$

SliceGPT (Ashkboos et al. 2024)

- **Important observation:** Orthogonally transforming an unpruned weight matrix preserves layer's output (*aka* computational invariance)
- To compute the orthogonal matrix Q for each prunable weight matrices, they compute the gram matrix on the calibration dataset – $\mathbf{C}_\ell = \sum_i \mathbf{X}_{\ell,i}^\top \mathbf{X}_{\ell,i}$
- The orthogonal matrix Q is the eigenvectors of the matrix C.
- The least important dimensions (based on the eigenvalues) are sliced from the resultant matrix XQ and Q^TW.

Layer Collapse (Yang et al. 2024)

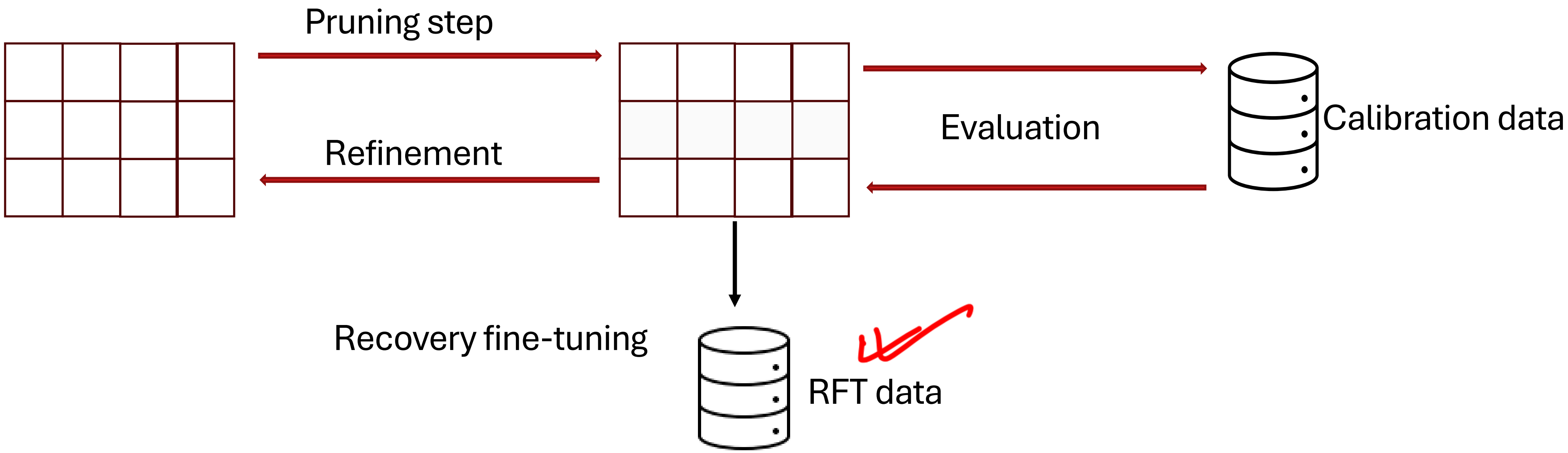
- Layer Collapse (LaCo) removes redundant layers directly from a pre-trained LLM. It performs the layer dropping by collapsing consecutive layers.
- The method calculates similarity of layer outputs and collapses subsequent layers if the layer-wise similarity on a calibration dataset is below a chosen threshold.



$\mathcal{M}^*$
RDSC Layer Merge
$\theta_{\mathcal{H}}$
$\theta_{l+\mathcal{K}}$
θ_{l+2}
θ_{l+1}
θ_l
$\theta_{\mathcal{L}}$
$\mathcal{K} = Min(\mathcal{C}-1, Lay\_cnt(\mathcal{M}^*))$
Try RDSC layer merging the following $\mathcal{K}$ layers of l
$\mathcal{M}_{tmp}$
$\theta_l + \Delta\theta_1 + \ldots + \Delta\theta_{\mathcal{K}}$
Calculate Similarity
Given the model $\mathcal{M}$ before pruning, evaluate $\mathcal{M}_{tmp}$ on $\mathcal{D}$
$s = Avg\_Cos\_Sim(\mathcal{M}_{tmp}, \mathcal{M}, \mathcal{D})$
Merge Evaluation and Adjustment
$s > \mathcal{T}$
$s \leq \mathcal{T}$
$\mathcal{M}^* = \mathcal{M}_{tmp}$
$l = Min(l - \mathcal{I}, Lay\_cnt(\mathcal{M}^*) - \mathcal{C})$
$l = l - 1$
$l \geq \mathcal{L}$
$l < \mathcal{L}$
Next Iteration
Pruned LLM

Sengupta et al. ICLR’25

Structure pruning requires calibration data



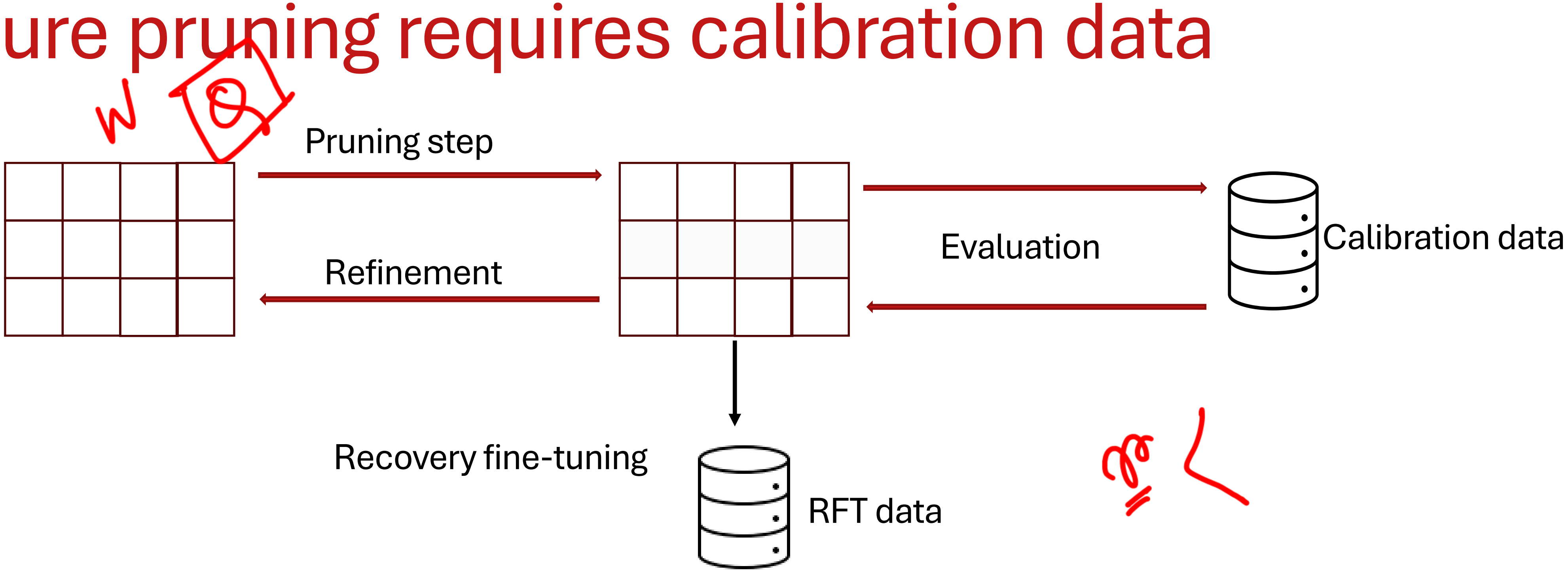
Existing structured pruning methods – SliceGPT (Ashkboos et al., 2024), LLM Pruner (Ma et al., 2023), Layer Collapse (Yang et al., 2024) use calibration data to determine the unimportant components of a pre-trained model for pruning.

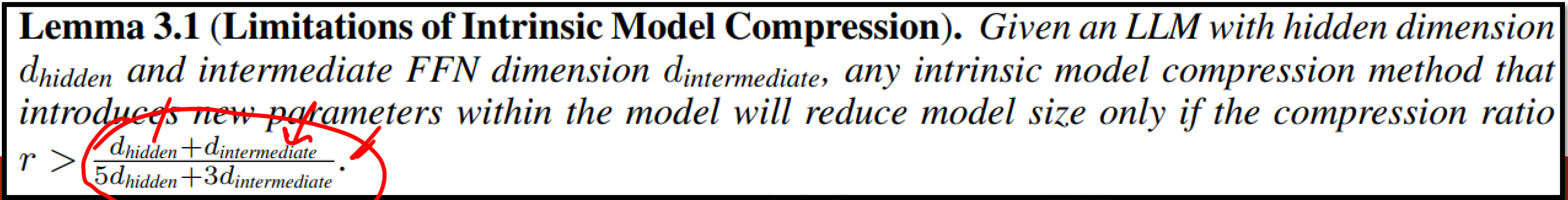
Limitations

1. Over-reliance on calibration data makes the compressed model sensitive to the data selection, becomes less reliable on downstream tasks (Ji et al., 2025)
2. Recovery fine-tuning (RFT) is crucial for preserving performance of the models, post-compression

Sengupta et al. ICLR'25

Structure pruning requires calibration data





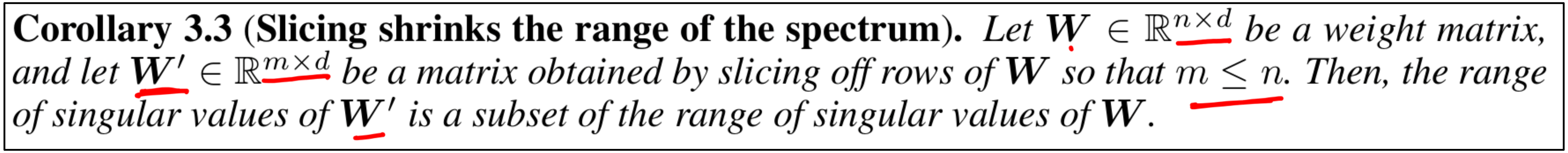
Lemma 3.1 (Limitations of Intrinsic Model Compression). *Given an LLM with hidden dimension d_{hidden} and intermediate FFN dimension $d_{intermediate}$, any intrinsic model compression method that introduces new parameters within the model will reduce model size only if the compression ratio* $r > \frac{d_{hidden}+d_{intermediate}}{5d_{hidden}+3d_{intermediate}}$.

ICLR’25

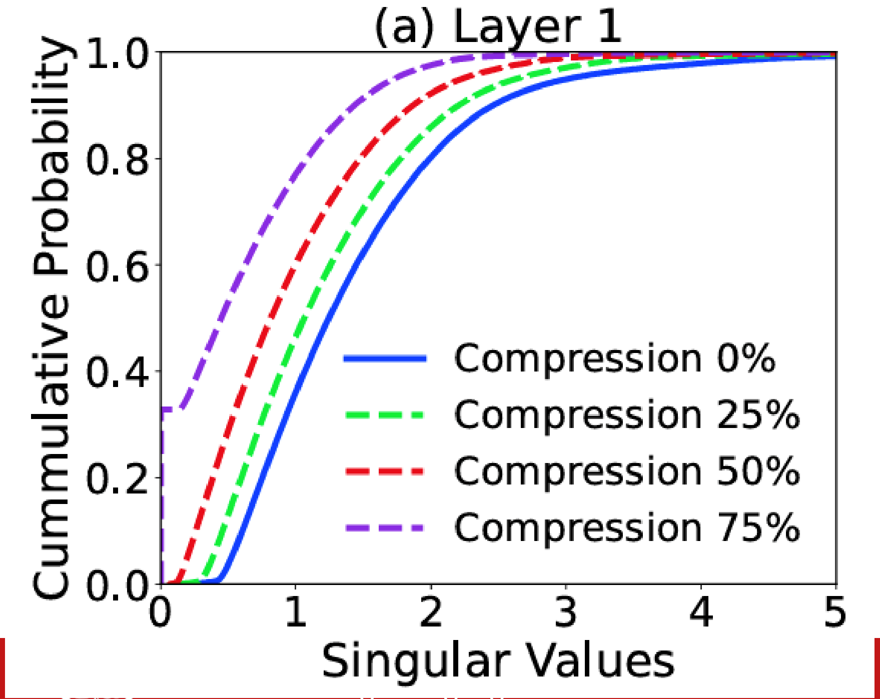
Can we use Intrinsic Metrics for Pruning

W

Poincare Separa theory



Corollary 3.3 (Slicing shrinks the range of the spectrum). *Let* $\mathbf{W} \in \mathbb{R}^{n \times d}$ *be a weight matrix, and let* $\mathbf{W}' \in \mathbb{R}^{m \times d}$ *be a matrix obtained by slicing off rows of* $\mathbf{W}$ *so that* $m \leq n$*. Then, the range of singular values of* $\mathbf{W}'$ *is a subset of the range of singular values of* $\mathbf{W}$.

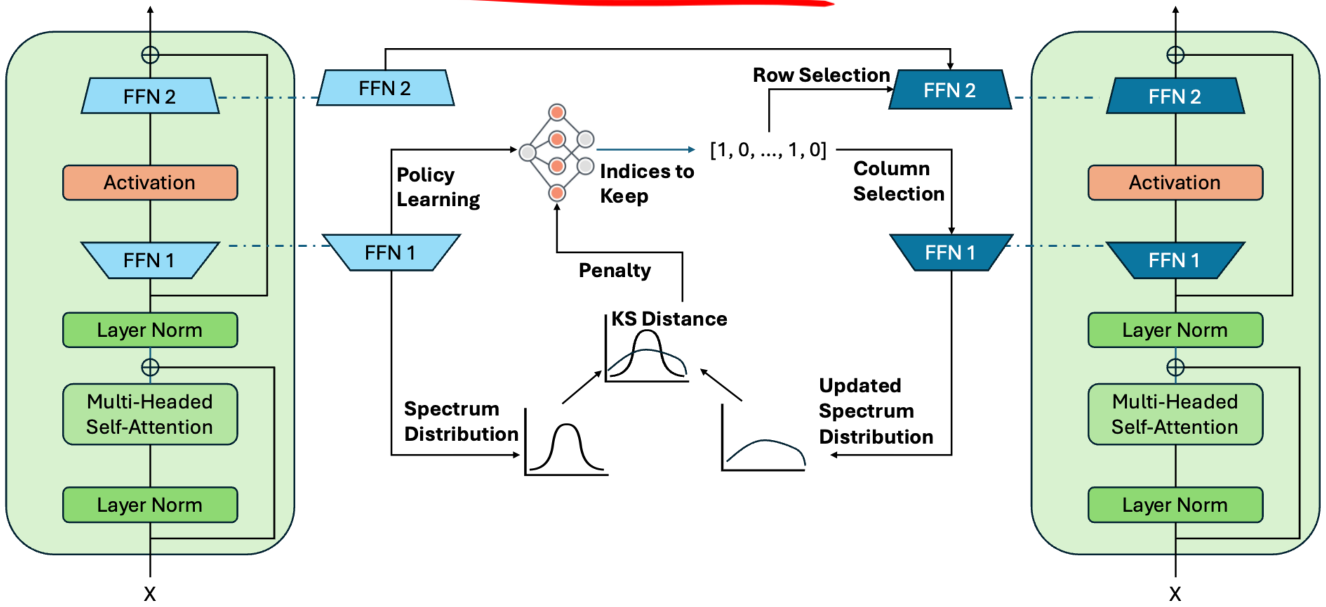


Singular values of a matrix determine the importance of each component.

Can we preserve the singular value structure (spectral structure) to preserve the performance of compressed model?

Sengupta et al. ICLR’25

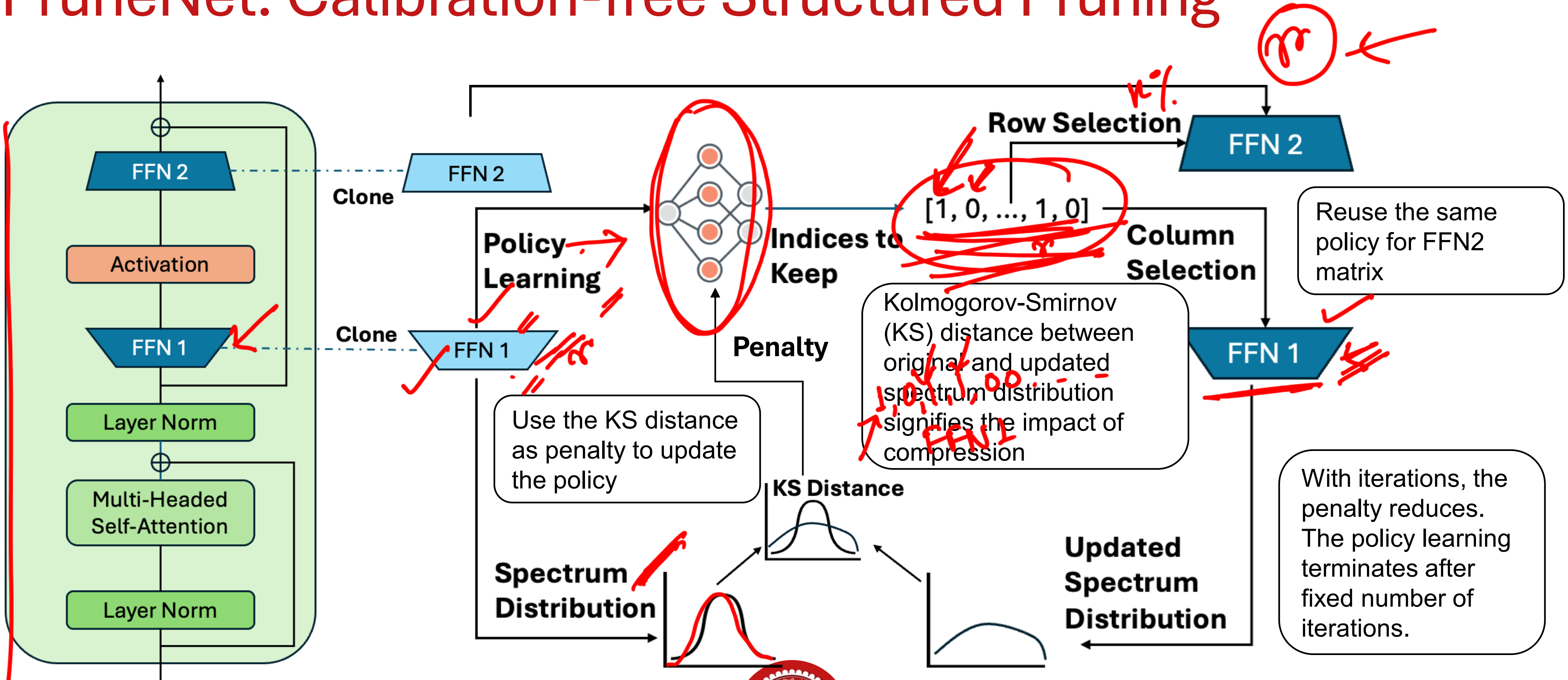
PruneNet: Calibration-free Structured Pruning



- ***PruneNet*** treats model compression as a **policy-learning process** that assesses the parameter importance once (using intrinsic methods) and can reuse the policy to compress the model at multiple compression ratios, at once.
- PruneNet is highly flexible, reusable and does not use sensitive and unreliable mechanisms like calibration.

ICLR’25

PruneNet: Calibration-free Structured Pruning



Reuse the same policy for FFN2 matrix

Kolmogorov-Smirnov (KS) distance between original and updated spectrum distribution signifies the impact of compression

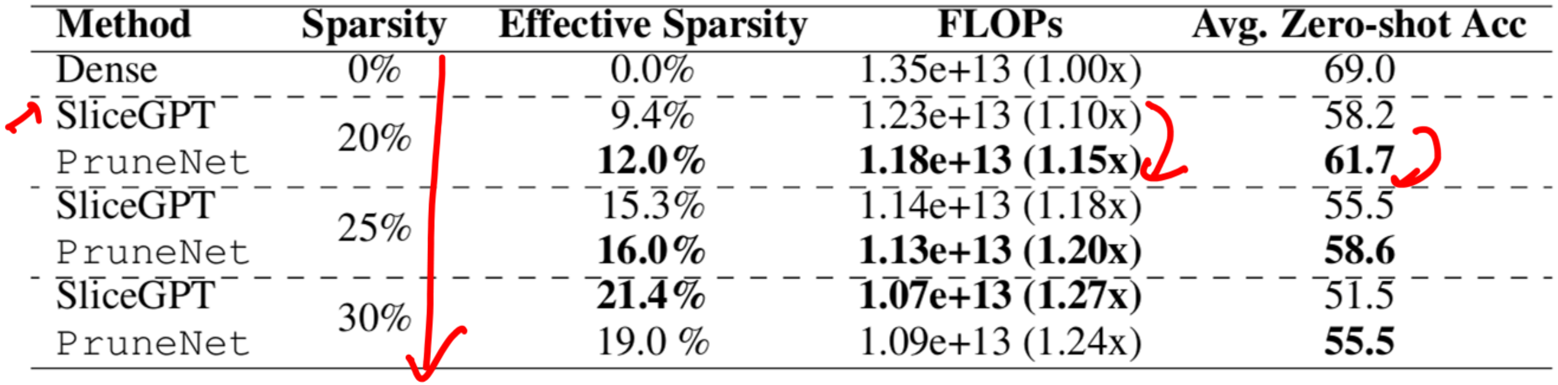
Use the KS distance as penalty to update the policy

With iterations, the penalty reduces. The policy learning terminates after fixed number of iterations.

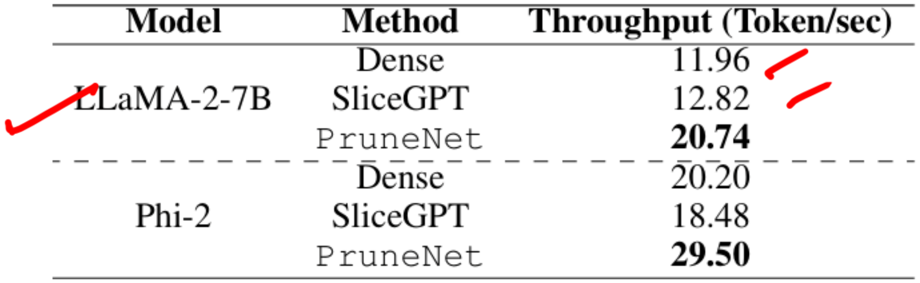
A policy learner assesses the different column indices of FFN1 matrix for a Transformer block

ICLR’25

Effectiveness of PruneNet: Empirical Evidence



Method	Sparsity	Effective Sparsity	FLOPs	Avg. Zero-shot Acc
Dense	0%	0.0%	1.35e+13 (1.00x)	69.0
SliceGPT	20%	9.4%	1.23e+13 (1.10x)	58.2
PruneNet		**12.0%**	**1.18e+13 (1.15x)**	**61.7**
SliceGPT	25%	15.3%	1.14e+13 (1.18x)	55.5
PruneNet		**16.0%**	**1.13e+13 (1.20x)**	**58.6**
SliceGPT	30%	**21.4%**	**1.07e+13 (1.27x)**	51.5
PruneNet		19.0 %	1.09e+13 (1.24x)	**55.5**



Model	Method	Throughput (Token/sec)
LLaMA-2-7B	Dense	11.96
	SliceGPT	12.82
	PruneNet	**20.74**
Phi-2	Dense	20.20
	SliceGPT	18.48
	PruneNet	**29.50**

PruneNet achieves higher effective sparsity and efficiency while maintaining better performance on downstream tasks.

Effective sparsity indicates the memory reduction in the compressed model.

LLaMA-2-7B compressed with PruneNet exhibits 73% better inference throughput than the original model.

ICLR'25

Takeaways

- LLaMA-2-7B compressed with PruneNet exhibits **73% better inference throughput** than the original model.

- PruneNet can compress LLaMA-2-7B in just 15 minutes by 30%, achieving over 80% retention of its zero-shot performance.

- PruneNet is **architecture-agnostic** and can be applied on any pre-trained network, without the need for any calibration.

https://github.com/LCS2-IIITD/PruneNet