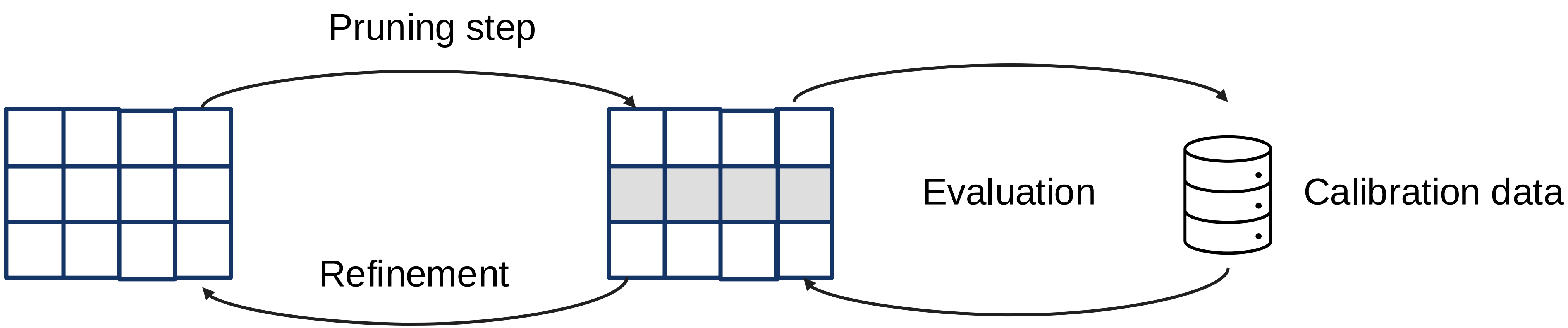
You Only Prune Once: Designing Calibration-Free Model Compression with Policy Learning

Ayan Sengupta, Siddhant Chaudhury, Tanmoy Chakraborty

Structured Model Pruning



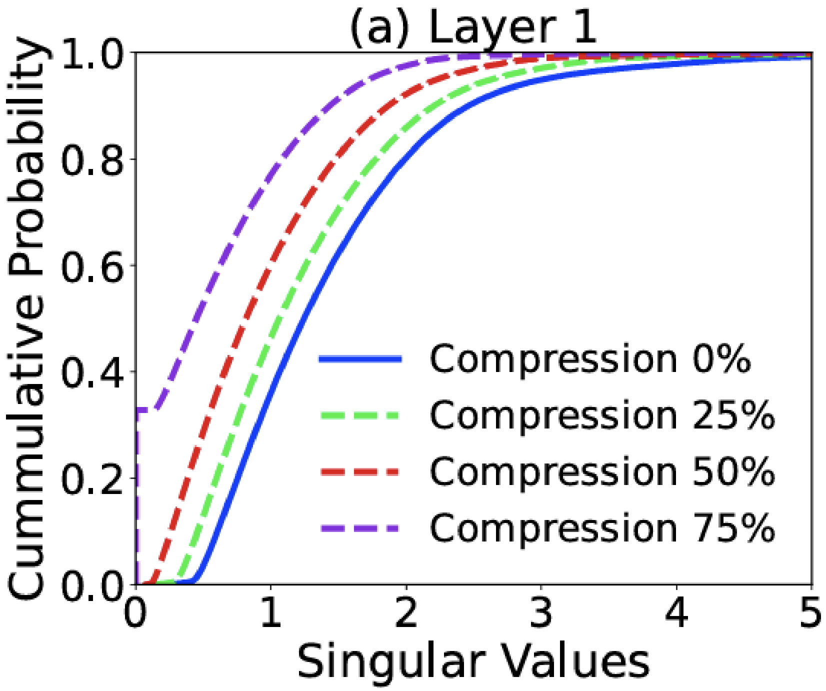
Existing structured pruning methods – SliceGPT (Ashkboos et al., 2024), LLM Pruner (Ma et al., 2023), Layer Collapse (Yang et al., 2024) use calibration data to determine the unimportant components of a pre-trained model for pruning.

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

Can we use Intrinsic Metrics for Pruning

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 $\mathbf{W}$.* [4]

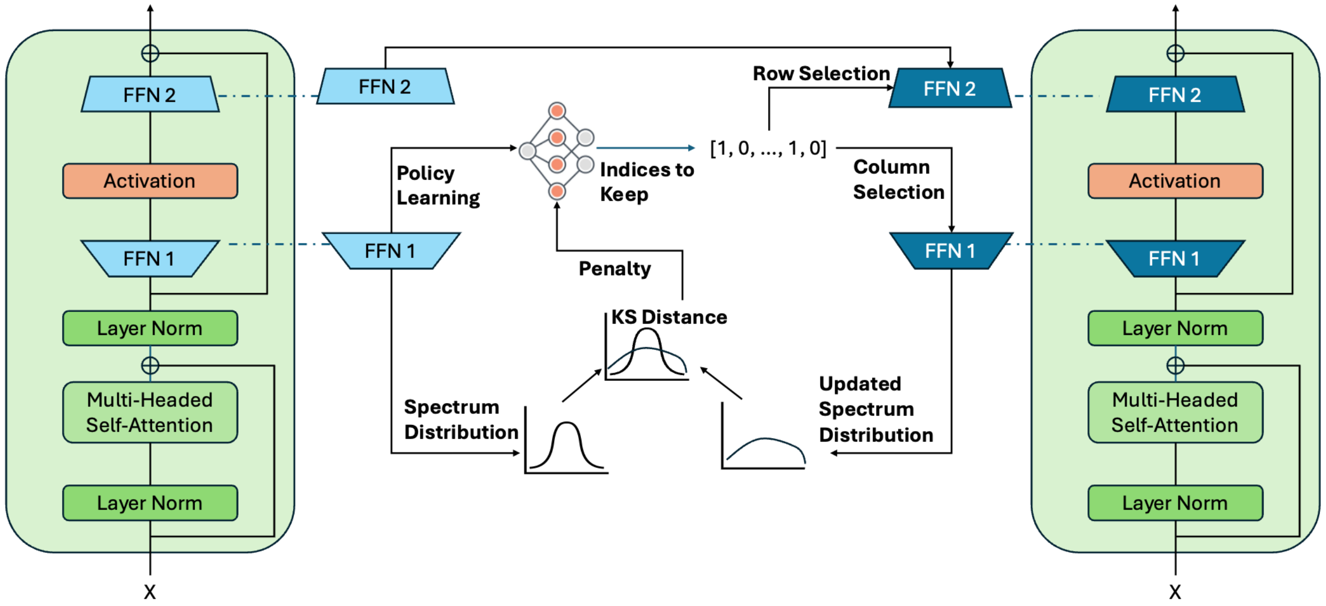


Singular values of a matrix determine the importance of each component.

With more compression, the distribution of singular values becomes more right-skewed

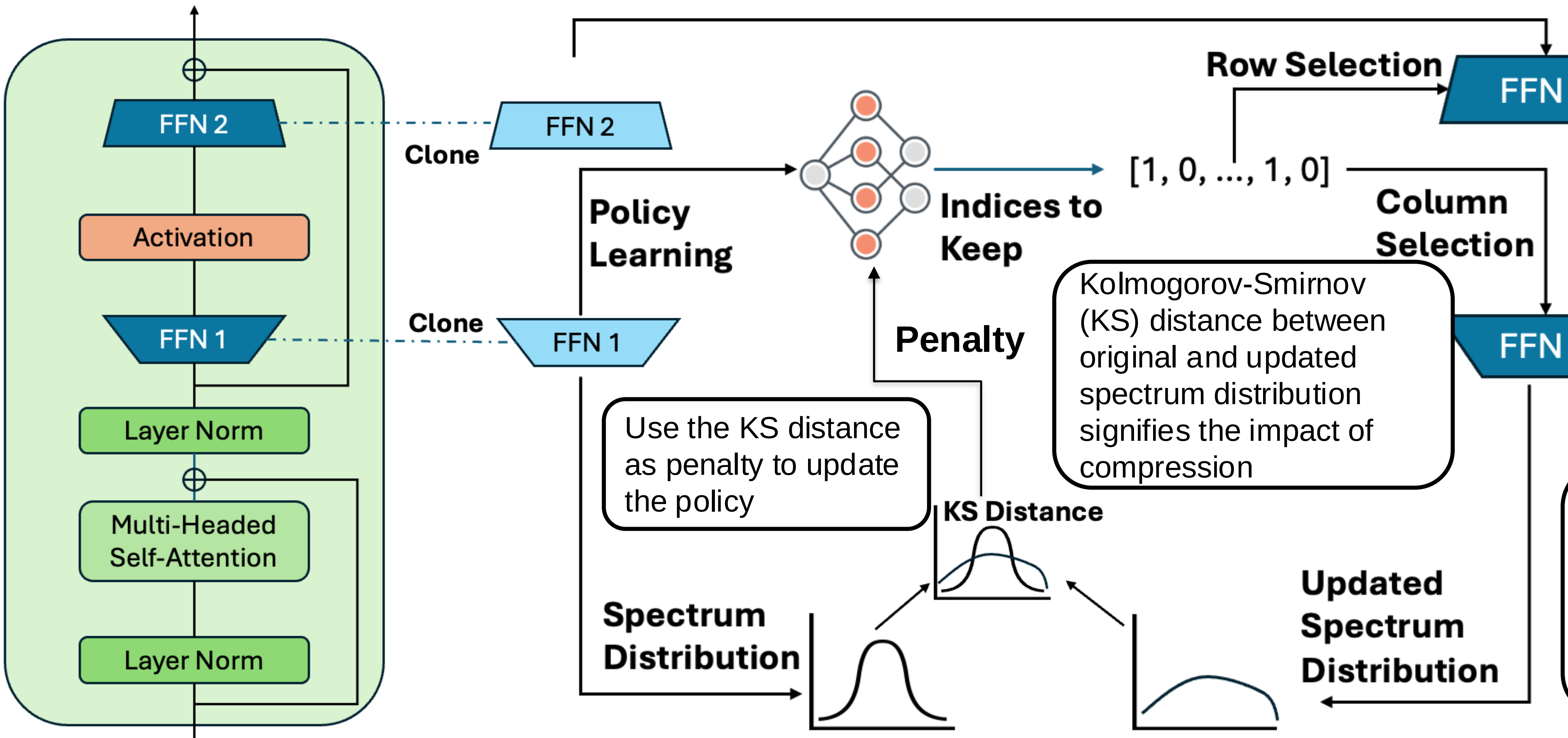
Can we preserve the singular value structure (spectral structure) to preserve the performance of compressed model?

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.

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

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.

Performance of Compressed LLMs without RFT

Model	Comp. Ratio	Method	PIQA	WinoGrande	HellaSwag	ARC-e	ARC-c	Avg.
LLaMA-2-7B	0%	Dense	79.11 (100%)	69.06 (100%)	75.99 (100%)	74.58 (100%)	46.25 (100%)	69.00 (100%)
	20%	SliceGPT	69.42 (88%)	65.11 (94%)	59.04 (78%)	59.76 (80%)	**37.54 (81%)**	58.17 (84%)
		PruneNet	**75.30 (95%)**	**65.51 (95%)**	**66.43 (87%)**	**63.80 (85%)**	37.29 (81%)	**61.67 (89%)**
	25%	SliceGPT	66.87 (84%)	**63.38 (92%)**	54.16 (71%)	58.46 (78%)	34.56 (75%)	55.48 (80%)
		PruneNet	**72.09 (91%)**	62.43 (90%)	**62.33 (82%)**	**60.14 (81%)**	**36.18 (78%)**	**58.63 (85%)**
	30%	SliceGPT	63.55 (80%)	**61.33 (89%)**	49.62 (65%)	51.77 (69%)	31.23 (67%)	51.50 (75%)
		PruneNet	**71.11 (90%)**	61.09 (88%)	**58.30 (77%)**	**53.20 (71%)**	**33.53 (72%)**	**55.45 (80%)**
Phi-2	0%	Dense	79.11 (100%)	75.77 (100%)	73.83 (100%)	78.32 (100%)	54.18 (100%)	72.24 (100%)
	20%	SliceGPT	71.87 (91%)	67.80 (89%)	57.76 (78%)	58.00 (74%)	35.32 (65%)	58.15 (80%)
		PruneNet	**74.37 (94%)**	**70.80 (93%)**	**65.53 (89%)**	**74.71 (95%)**	**47.53 (88%)**	**66.59 (92%)**
	25%	SliceGPT	69.21 (88%)	65.35 (86%)	52.40 (71%)	53.7 (69%)	31.66 (58%)	54.46 (75%)
		PruneNet	**74.37 (94%)**	**68.98 (91%)**	**62.18 (84%)**	**70.54 (90%)**	**44.45 (82%)**	**64.10 (89%)**
	30%	SliceGPT	65.94 (83%)	63.14 (83%)	47.56 (64%)	53.03 (68%)	30.29 (56%)	51.99 (72%)
		PruneNet	**72.80 (92%)**	**67.48 (89%)**	**56.80 (77%)**	**67.55 (86%)**	**40.61 (75%)**	**61.05 (84%)**

Downstream performance comparison of PruneNet and SliceGPT. PruneNet consistently outperforms other methods even in the absence of recovery fine-tuning (RFT).

Importance of PruneNet for Efficient Model Pruning

Key takeaways:

A. PruneNet is highly reusable, where the compression policy learned at lower compression ratio can be used to compress model at higher compression ratio, while significantly retaining performance.

B. PruneNet is also faster than most competitive compression methods. LLaMA-2-7B model can be compressed in just 15 minutes, 50% faster than SliceGPT

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