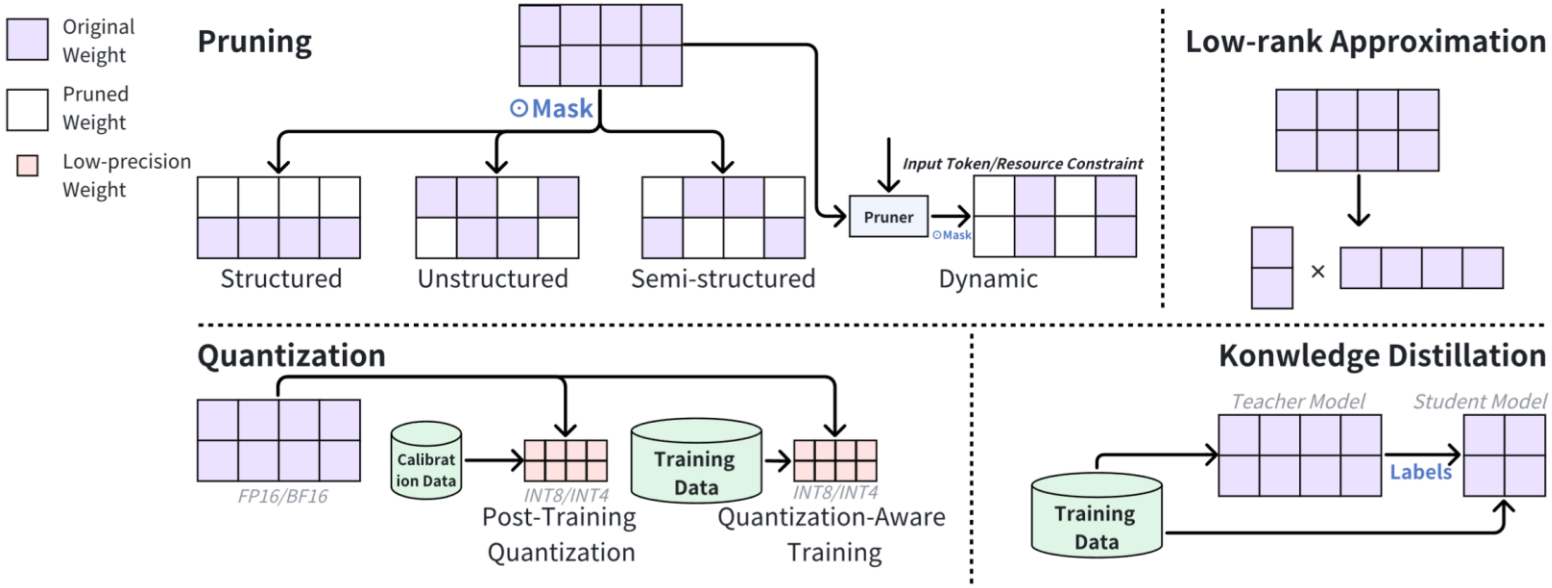
A Study and Formal Framework of the Composability of LLM Compression Techniques

Gansen HU, Zhaoguo WANG

Frontiers of Computer Science, DOI: 10.1007/s11704-025-50814-1

Problems & Goal

- Problems of single LLM compression technique:
  - Limited maximal compression ratio and degraded model accuracy
  - When combining multiple compression techniques, manually conduct experiments to determine the best combination
- Our Goal
  - Provide insights on the internal implications when combine multiple LLM compression techniques
  - Automatically determine the best combination with theoretical guarantee



Overview of LLM compression techniques. The figure illustrates the four main compression techniques: pruning, quantization, knowledge distillation,and low-rank approximation. Each technique is briefly described, highlighting its key principles and distinctions.

Main Contributions

- Contributions:
  - A comprehensive study of the composability of compression techniques for large language models, including pruning, quantization, knowledge distillation, and low-rank approximation;
  - An analysis of the interactions between these techniques and their impact on model performance, offering insights into how these techniques can be effectively combined to achieve better results;
  - A formal framework for identifying the optimal combination order of compression techniques for large language models.

Name	Compression Techniques Used				Apply Order	Max Reported Compress Ratio	Model
	Pruning	Quantization	Distilation	Low-rank			
LoRAPrune [55]	✓(St)			✓	Joint	50%	LLaMA-7B
LoRAShear [57]	✓(St)			✓	L->P	50%	LLaMA-7B
LoSparse [56]	✓(St)			✓	Joint	50%	DeBERTaV3-86M
LoSA [58]	✓(U)			✓	Joint	70%	LLaMA1~3 7B~70B
LPLR [59]		✓		✓	Joint	75%	LLaMA-7B
ZeroQuant [6]		✓	✓		Q->D	75%	GPT3-1.3B
ZeroQuant-V2 [60]		✓		✓	Q->L	87.5%	OPT-6.7~66B BLOOM-176B
ZeroQuant-FP [61]		✓		✓	Q->L	75%	LLaMA-3B~30B OPT-3B~30B
QLoRA [8]		✓		✓	Q->L	75%	LLaMA-7~65B
NutePrune [62]	✓(St)		✓	✓	P->L->D	50%	LLaMA-7B
SDMPrune [63]	✓(St)		✓		Joint	40%	LLaMA3.2-1.2B
CLM [64]	✓(St)		✓		P->D	75%	Nemotron4-15B
EfficientVLM [65]	✓(St)		✓		D->P	80%	CLIP-VIT
CPKD [66]	✓(Se)		✓		P->D	75%	BERT-110M
PruneOFA [67]	✓(U)		✓		P->D	90%	BERT-Base/Large
AST [68]	✓(Se)		✓	✓	P->D->L	90%	LLaMA2-7B
PPC-GPT [69]	✓(St)		✓		P->D	70%	LLaMA2-7B/OPT6.7B
SDD [70]	✓(St)		✓		P->D	80%	Llama3.1-8B
Minitron [71]	✓(St)		✓		P->D	80%	LLaMA3.1-8B Mistral NeMo 12B

```
# Symbolic encoding of magnitude-based pruning
# Input: weights elements , input activation samples
# Output: pruned weights
# The following code is written in a syntax
    compatible with SMT solvers
def prune_weights(w11,w12,w21,w22 ,x11,x12,x21,x22 ):
  # Find the minimum absolute weight value
  min_val = If(Abs(w11) < Abs(w12), w11, w12)
  min_val = If(Abs(w21) < min_val, w21, min_val)
  min_val = If(Abs(w22) < min_val, w22, min_val)
  # Extending to Wanda
  s11 = Abs(w11) * ((x11*x11 + x21*x21) ** 0.5)
  s12 = ...#s12,s21,s22 processed similarly
  min_score = If(s11 < s12, s11, s12)
  min_score = If(s21 < min_score, s21, min_score)
  min_score = ...  # s12, s22 omitted
return If(s11==min_score,0,s11), If(s12==min_score...)
return If(Abs(w11) <= Abs(min_val), 0, w11),
       If(Abs(w12) <= Abs(min_val), 0, w12),
       If(Abs(w21) <= Abs(min_val), 0, w21),
       If(Abs(w22) <= Abs(min_val), 0, w22)
```

Sample of studied papers and the key pseudocode of the formal framework. Left: table of some of the studied methods, check the paper for a complete list; Right: the core formal encoding logic for pruning based algorithms.