Introduction to Large Language Models

Spring 2026

LMM Model Compression

(Some slides adapted from Ralph Grishman at NYU, Yejin Choi at UWashington, N. Tomura at UDepaul, Jurafsky and Martin, CS224N, CS224, CME295 at Stanford and other resourses on the web)

LLM Compression

- **LLM Compression** refers to techniques that reduce the *size, memory footprint, and/or computational cost* of Large Language Models (LLMs) while preserving as much performance as possible. Compression is essential for **deployment on edge devices**, **cost-efficient serving**, **lower latency**, and **energy efficiency**.

- ## Why Compress LLMs?

  LLMs are expensive because they:

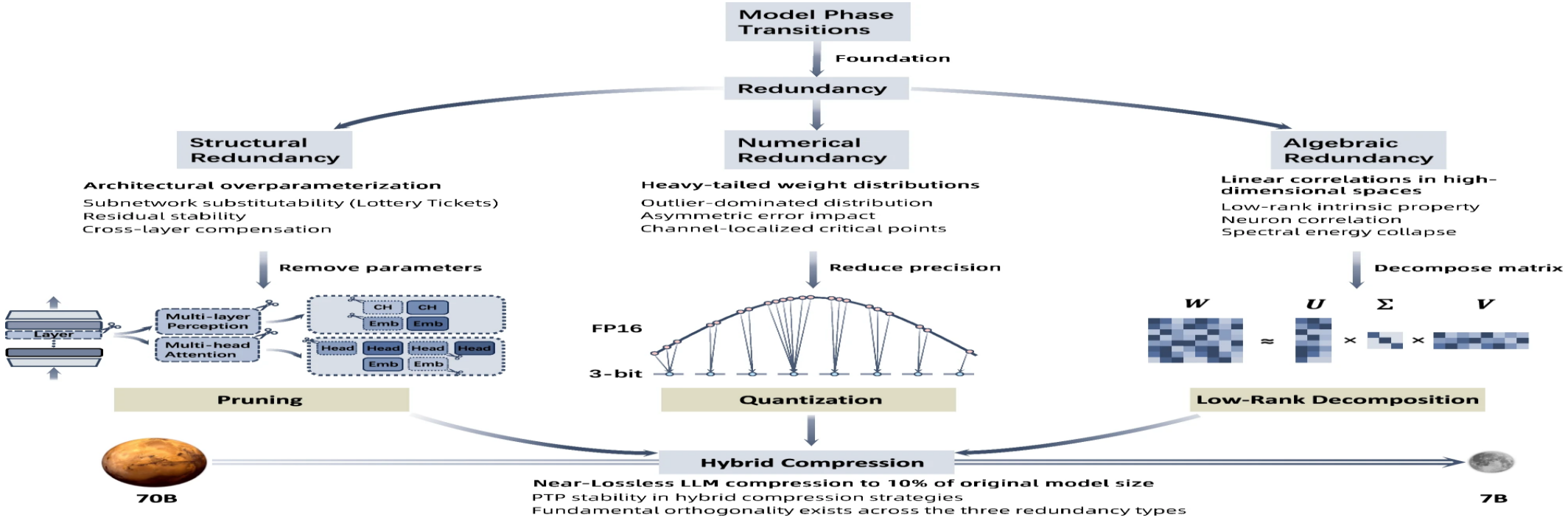
  - Contain **billions of parameters**
  - Require **large GPU/TPU memory**
  - Have **high inference latency and energy cost**

  Compression makes it possible to:

  - Run models on **consumer GPUs, CPUs, or edge devices**
  - Reduce **serving cost at scale**
  - Enable **on-device or private deployment**
  - Improve **throughput and response time**

Main LLM Compression Techniques

Method	Size Reduction	Speed Gain	Accuracy Risk
Quantization	★★★★☆	★★★★☆	Medium
Pruning	★★☆☆☆	★★☆☆☆	Medium–High
Distillation	★★★★☆	★★★☆☆	Low–Medium
Low-rank	★★☆☆☆	★★☆☆☆	Medium

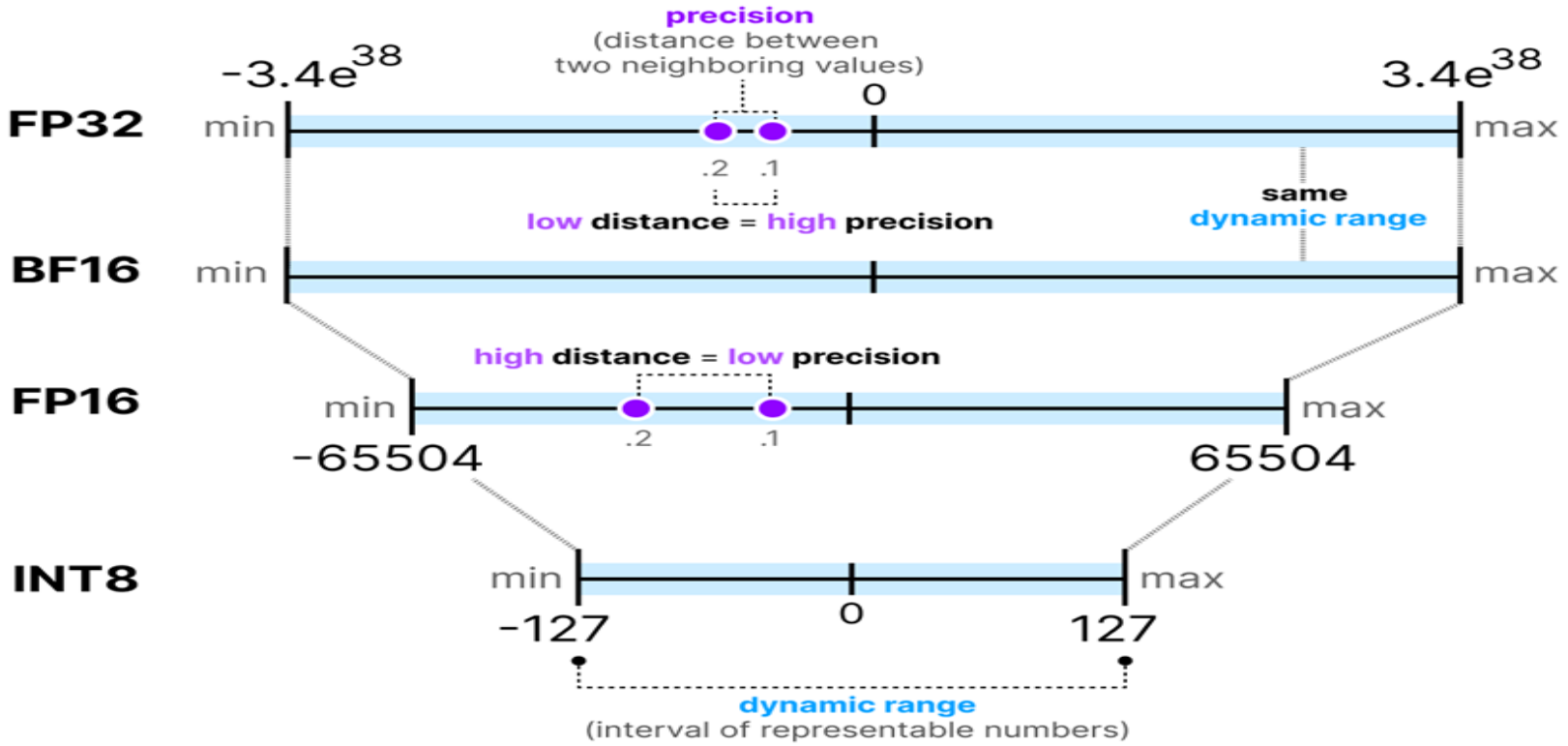


Quantization

Idea: Represent weights and activations with **lower precision** numbers.

Common formats

- FP16 / BF16 → baseline
- INT8 → minimal accuracy loss
- INT4 / INT3 → aggressive, often needs careful calibration



https://www.redhat.com/en/blog/llm-compression-and-optimization-cheaper-inference-fewer-hardware-resources

Quantization

Type	Description	Typical Precision
Post-Training Quantization (PTQ)	Applied after training	INT8, INT4
Quantization-Aware Training (QAT)	Model trained with quantization in mind	INT8, INT4
Weight-only Quantization	Activations remain FP	INT8–INT2

- **PTQ quantizes a fully trained model *after* training**, without updating weights via backpropagation. The model is trained in high precision (FP32/FP16), then converted to lower precision for inference.
- **QAT simulates quantization *during training*** so the model learns to be robust to low-precision arithmetic. Quantization noise is injected while training.
- **In WOQ, Only the model weights are quantized**, while activations remain FP16/FP32. This is the **most popular approach for LLM inference today**.

Feature	PTQ	QAT	Weight-Only
Retraining needed	❌	✅	❌
Training cost	None	High	None
Inference speedup	Medium–High	High	Medium
Accuracy at INT4	Medium	**Best**	Good
Popular for LLMs	✅ ✅ ✅	❌	✅ ✅ ✅
Best hardware	GPU / CPU	Edge / ASIC	GPU

Pruning

Idea: Remove unnecessary parameters or structures.

Types

- **Unstructured pruning**: Remove individual weights
- **Structured pruning**: Remove entire neurons, heads, or layers

Examples:

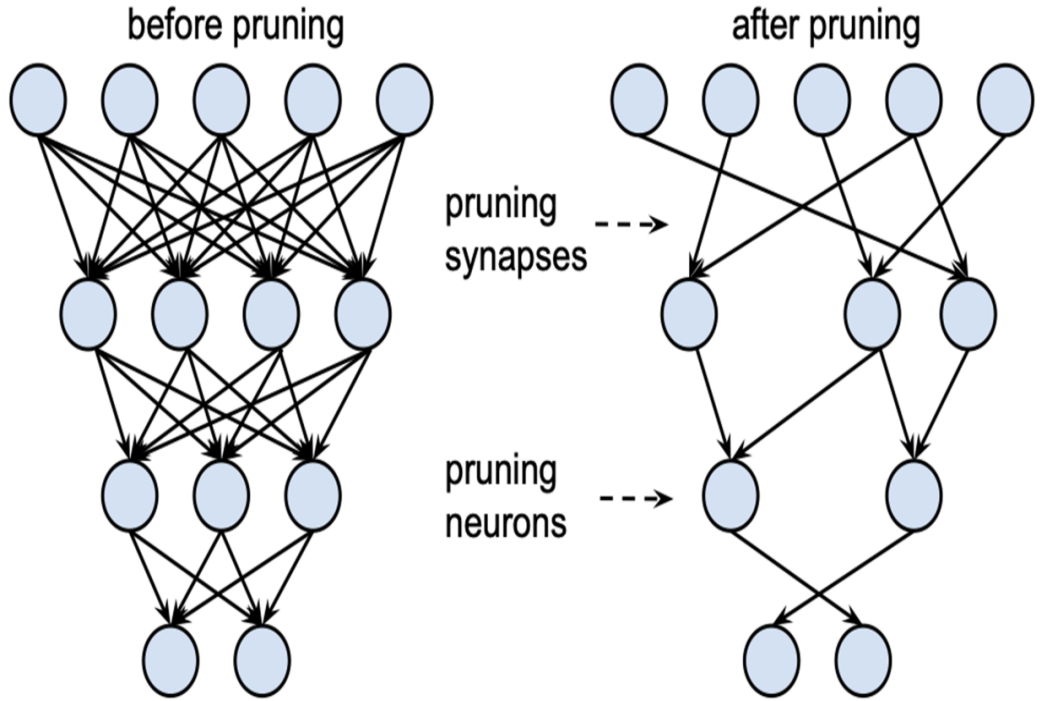
- Attention head pruning
- Feed-forward neuron pruning
- Layer dropping

✅ Pros:

- Theoretical sparsity gains
- Can reduce compute

⚠️ Cons:

- Unstructured sparsity is hard to exploit o real hardware
- May require retraining or fine-tuning



Knowledge Distillation

- **Knowledge Distillation (KD)** is a model compression and training technique in machine learning where a **smaller model (student)** is trained to replicate the behavior of a **larger, more powerful model (teacher)**. The teacher transfers its knowledge not just through hard labels, but through **soft probability distributions**, which encode richer information about class relationships and decision boundaries.
-

  Idea: Train a *smaller "student" model* to mimic a larger *"teacher" model*.

The student learns from:

- Soft logits
- Hidden states
- Attention maps

Examples:

- DistilBERT ← BERT
- TinyLLaMA ← LLaMA-style teachers

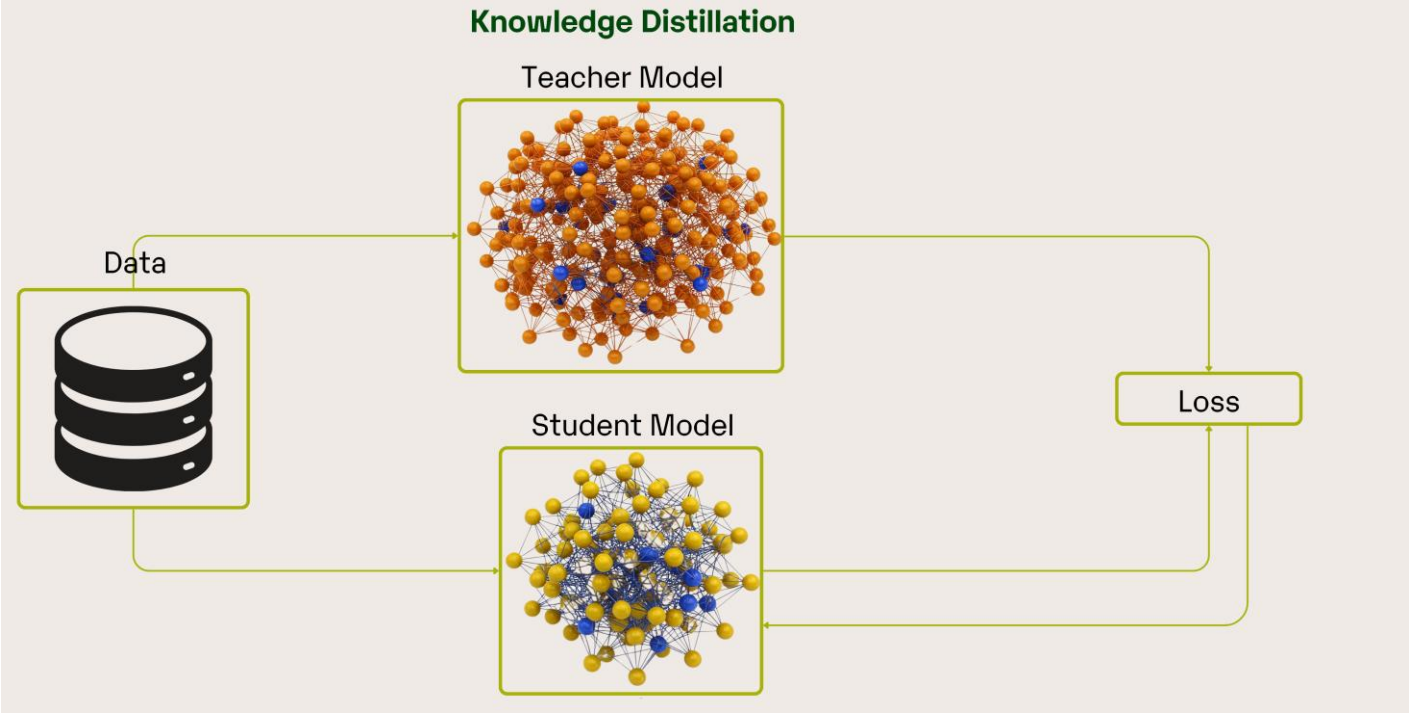
✅ Pros:

- Strong performance-to-size ratio
- Clean, dense models (hardware-friendly)

⚠️ Cons:

- Requires retraining

Student is limited by teacher's capabilities



Knowledge Distillation

How Knowledge Distillation Works

1. Teacher Model

- A large, high-accuracy model (e.g., LLaMA-70B, BERT-Large)
- Expensive to run and deploy

2. Student Model

- Smaller, cheaper model (e.g., 7B, 3B, or even 1B parameters)
- Designed for efficiency

3. Distillation Loss

Typically a combination of:

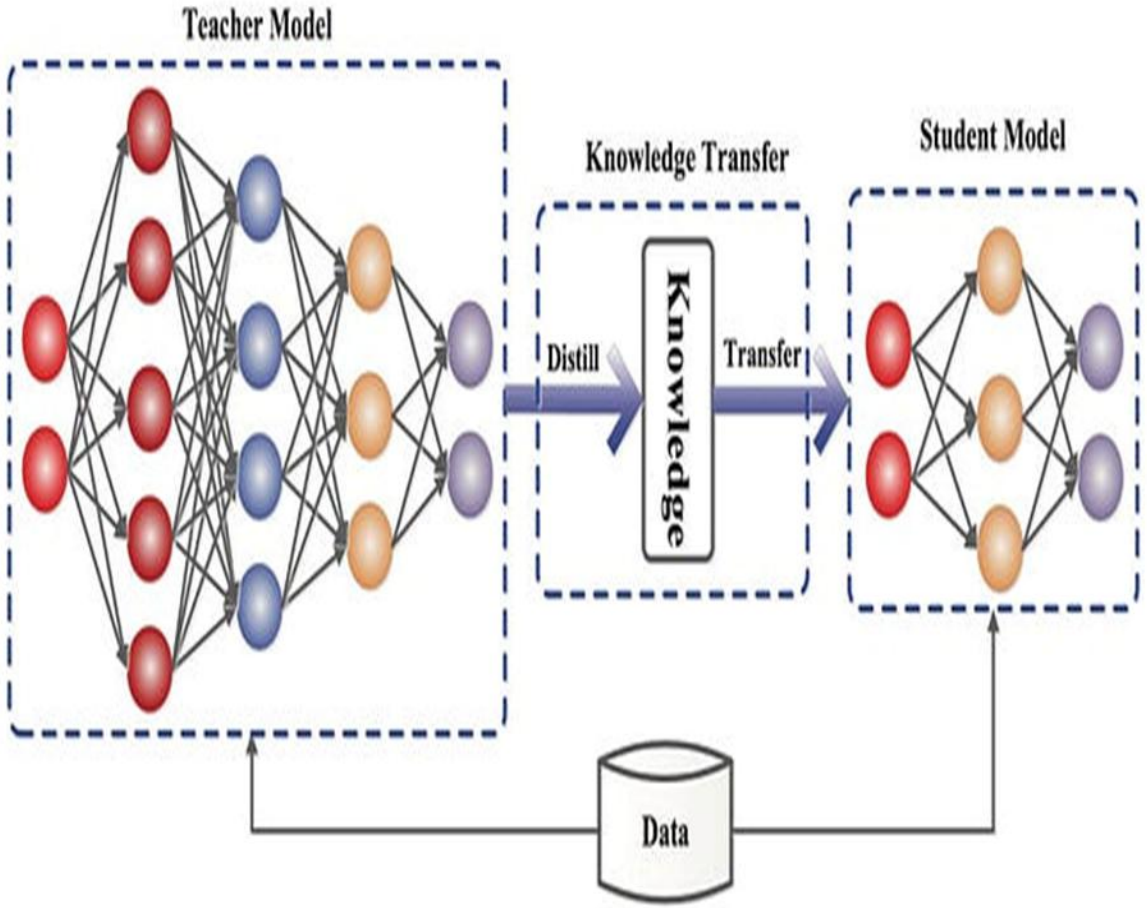
- **Hard loss**: standard cross-entropy with ground-truth labels
- **Soft loss**: divergence between teacher and student output distributions

A common formulation:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{CE}(y, \hat{y}_s) \; + \; (1 - \alpha) \cdot T^2 \cdot KL(\hat{y}_t^T \; \| \; \hat{y}_s^T)$$

Where:

- T = temperature (softens probabilities)
- KL = Kullback–Leibler divergence



Low-Rank Factorization

Idea: Approximate large weight matrices using low-rank decompositions.

Techniques:

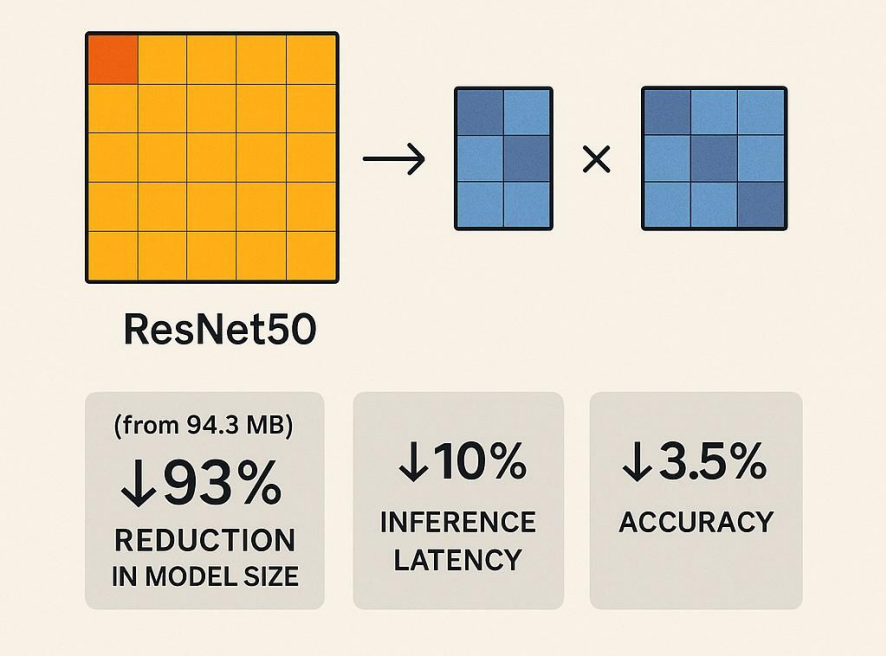
- SVD
- LoRA-style low-rank matrices (for parameter-efficient fine-tuning)
- Tensor decomposition

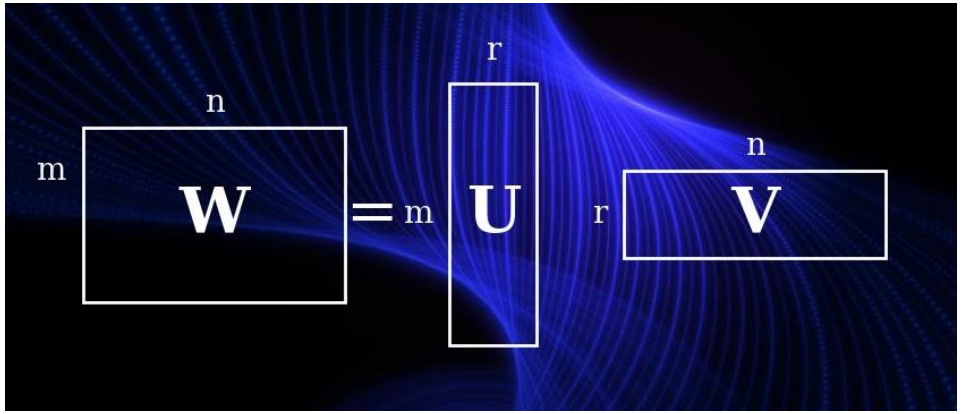
✅ Pros:

- Reduces parameters and compute
- Can be combined with fine-tuning

⚠️ Cons:

- Approximation error
- Optimal rank selection is nontrivial





https://arikpoz.github.io/posts/2025-04-29-low-rank-factorization-in-pytorch-compressing-neural-networks-with-linear-algebra/

Parameter Efficient Fine Tuning (PEFT) vs LLM Compression

Parameter-Efficient Fine-Tuning (PEFT)

- **Primary goal:**
  ➡️ *Adapt a model to a new task with minimal training cost*
- PEFT targets **training efficiency**:
- Avoid full fine-tuning
- Reduce GPU memory during training
- Enable rapid domain adaptation
- You apply PEFT **during training / adaptation**.

LLM Compression

- **Primary goal:**
  ➡️ *Make a model cheaper and faster to run*
- Compression targets **inference efficiency**:
- Reduce memory footprint
- Reduce latency
- Reduce energy and serving cost
- Enable deployment on limited hardware
- You usually apply compression **after training**.

Combined Strategies (Most Effective in Practice)

Real systems **combine multiple methods**:
Example deployment stack:

```
Distilled model
→ INT4 weight-only quantization
→ Structured pruning
→ KV-cache compression
```

Method	Size Reduction	Speed Gain	Accuracy Risk
Quantization	★★★★☆	★★★★☆	Medium
Pruning	★★☆☆☆	★★☆☆☆	Medium–High
Distillation	★★★★☆	★★★☆☆	Low–Medium
Low-rank	★★☆☆☆	★★☆☆☆	Medium

Summary

LLM compression is not one technique, but a toolbox. The best approach depends on:

- Target hardware
- Acceptable accuracy loss
- Latency and cost constraints
- Whether retraining is possible

In practice, **quantization + distillation** gives the best ROI for most deployments.