COMS4995W32
Applied Machine Learning

Dr. Spencer W. Luo

Columbia University | Fall 2025

LLM Pre-training & Fine-tuning

Agenda

- Architecture recap
- LLM Pre-training
- LLM Fine-tuning
- Industry trends

Encoder vs Decoder

Decoder-only Family – Language Model (LM) 💬

Mask: Causal → token t can observe only tokens < t

Objective: Predict next token → $P(\text{token}_{□} \mid \text{tokens}_{(1\ldots□-1)})$

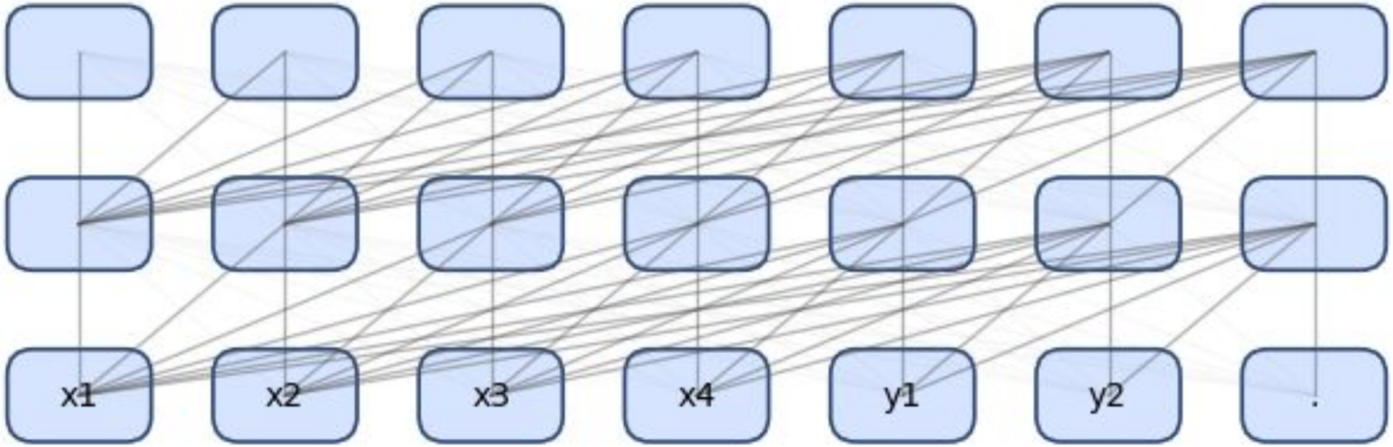
Key: “Generates” but does **not** encode full context

Examples:

Chat-GPT, LLaMA, Gemini, Mistral….

Decoder-only

causal self-attention



Encoder-only Family – Masked Language Model (MLM)🔍

Mask: Full attention → all tokens can attend bidirectionally

Objective: Predict randomly masked tokens using full context

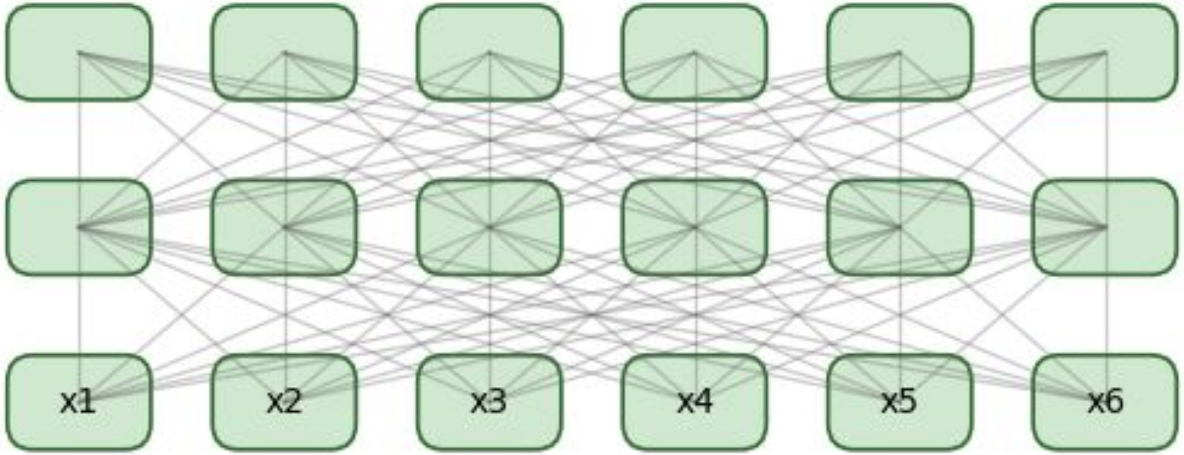
Key: “Understands” but does **not** generate

Examples:

BERT, RoBERTa, SpanBERT, ALBERT……

Encoder-only

bi-directional self-attention



Large Language Models (LLMs) 🤖

LLMs are:

- Transformer-based models
- Trained on massive text corpora - O(B) tokens scale
- Learn general-purpose language understanding and generation abilities

[Scaling Law] Model parameters, data, and compute → emergent intelligence

Key Characteristics:

- Trained with self-supervised next-token prediction
- Exhibit few-shot / zero-shot capabilities
- Can be further adapted via fine-tuning to many downstream tasks

From Architecture → Paradigm ⚙️

Old View: “Architecture Defines Intelligence”

- Researchers focused on building smarter structures (LSTM, Transformer)
- But even elegant designs failed without scale and rich data

Modern View: “Data + Algorithm Define Capability”

- The same Transformer backbone can become BERT, GPT, or T5, depending on the algorithmic design (PT, SFT and RL) and massive data

From Architecture → Paradigm ⚙️

Key Insight:

- Architecture shapes foundation, but data and algorithm shape behavior
- The training paradigm now matters more than structural novelty

Takeaway:

Modern breakthroughs (ChatGPT, Gemini) come not from new layers, but from how we teach and scale the same architecture, with better data

Analogy: Human Learning 🧍‍♀️

Stage	Human	Model (LLM)
Pre-training	Learning from reading, observation, and experience about the world	Trained on massive text corpus - learns language, facts, and world knowledge
Fine-tuning	Getting job-specific/domain training (e.g., becoming an engineer etc.)	Task-specific SFT on labeled data - adapts to domains or strengthens reasoning
Reinforcement Learning	Socialization and feedback - learning norms, politeness, ethics from others	Preference tuning (RLxF) - adjusts to ANY preference signals

Flow: Foundation → Specialization → Alignment

LLM Pre-training

What Is Pre-training? 📘

Train on a massive unlabeled text corpus

- Billions of words from books, and web pages - no manual labeling needed
- The model learns patterns of syntax, and world knowledge implicitly

Self-supervised objective:

- Does not need label
- Predict masked (Encoder-only) or next tokens (Decoder-only) from context
- The task creates its own supervision signal by hiding or shifting tokens

Pre-training Examples

Example 1 - Masked Language Modeling (BERT-style)

Input:

"The cat [MASK_1] on the [MASK_2]."

Target:

"The cat sat on the mat."

👉 The model must use bidirectional context to predict the masked token

Pre-training Examples

Example 2 - Next Token Prediction (GPT-style)

Input:

"The cat sat on the ?"

Target:

Next token = mat

👉 The model predicts the next word given all previous ones (causal connection)

What Is Pre-training? 📘

Outcome:

The model learns a general-purpose representation of language

- capturing grammar, meaning, and relationships between entities

Result:

The pre-trained model becomes:

- a reusable foundation
- easily adapted to many downstream tasks (classification, summarization, etc.) through fine-tuning

Why Self-Supervised? 🪄

Training signal comes directly from the data itself

Every word, punctuation, or phrase provides a new learning opportunity

This makes the world knowledge effectively one massive training corpus

Scales effortlessly:

Since labeling is automatic, models can train on trillions of tokens across diverse domains (books, code, social media etc)

Pre-training Data 📚

High-Quality Curated Web Data

- StackExchange / Reddit / Hacker News / ArXiv / PubMed - filtered for long-form reasoning, technical correctness, and linguistic quality
- C4 (Colossal Clean Crawled Corpus) - curated Common Crawl derivative used in T5
- RefinedWeb / Dolma / RedPajama / Falcon RefinedWeb - open-source large-scale cleaned web corpora

Pre-training Data 📚

Instructional / Human-Curated Data

- Wikipedia discussions, StackOverflow Q&A, instructional forums — naturally structured in “question → answer” form
- Open-sourced QA datasets: Natural Questions, SQuAD, GSM8K, MMLU-style evaluation corpora

Educational and Academic Texts

- Textbooks, lecture notes, academic papers (via S2ORC or ArXiv) - help models internalize reasoning and math syntax
- Project Gutenberg + scientific book scans - classic long-form, well-structured writing

Pre-training Data 📚

Code and Technical Content

- GitHub, StackOverflow, Jupyter Notebooks, Competitive Programming datasets

Dialogue & Conversational Corpora

- OpenSubtitles, MultiWOZ, OpenOrca - to enhance conversational fluency

Multilingual and Multimodal Text

- Multilingual datasets: Wikipedia in 100+ languages - encourage cross-lingual generalization
- Multimodal text: image captions, audio transcripts, or paired HTML/context

Pre-training Data 📚

Synthetic & Augmented Data (increasingly common)

Self-generated text from earlier LLMs (self-play, bootstrapping, or distillation)

Data augmentation via back-translation, paraphrasing, or chain-of-thought synthesis

Steps: Data cleaning → Deduplication → Quality filtering

Goals: Diversity, quality, and safety → Teach LLMs the basic knowledge

Pre-training Data Cleaning Pipeline 🧹

Deduplication

- Remove exact and near-duplicate documents using MinHash or SimHash

Content Filtering

- Exclude boilerplate (HTML templates, navigation bars, ads)

Toxicity & Safety Filtering

- Use classifiers to detect hate, harassment, adult, or violent content
- Combine rule-based filters with model-based toxicity scoring

Pre-training Data Cleaning Pipeline 🧹

Language & Format Detection

- Identify language via FastText or Compact Language Detector (CLD3)
- Keep target languages; drop mixed or unrecognized ones

Quality Scoring & Sampling

- Train classifiers or use perplexity thresholds to rank document quality.
- Sample more from high-quality sources (books, Wikipedia) than noisy web

Normalization & Finalization

- Lowercase normalization, punctuation cleanup, HTML stripping
- Mask or remove personally identifiable information (PII)

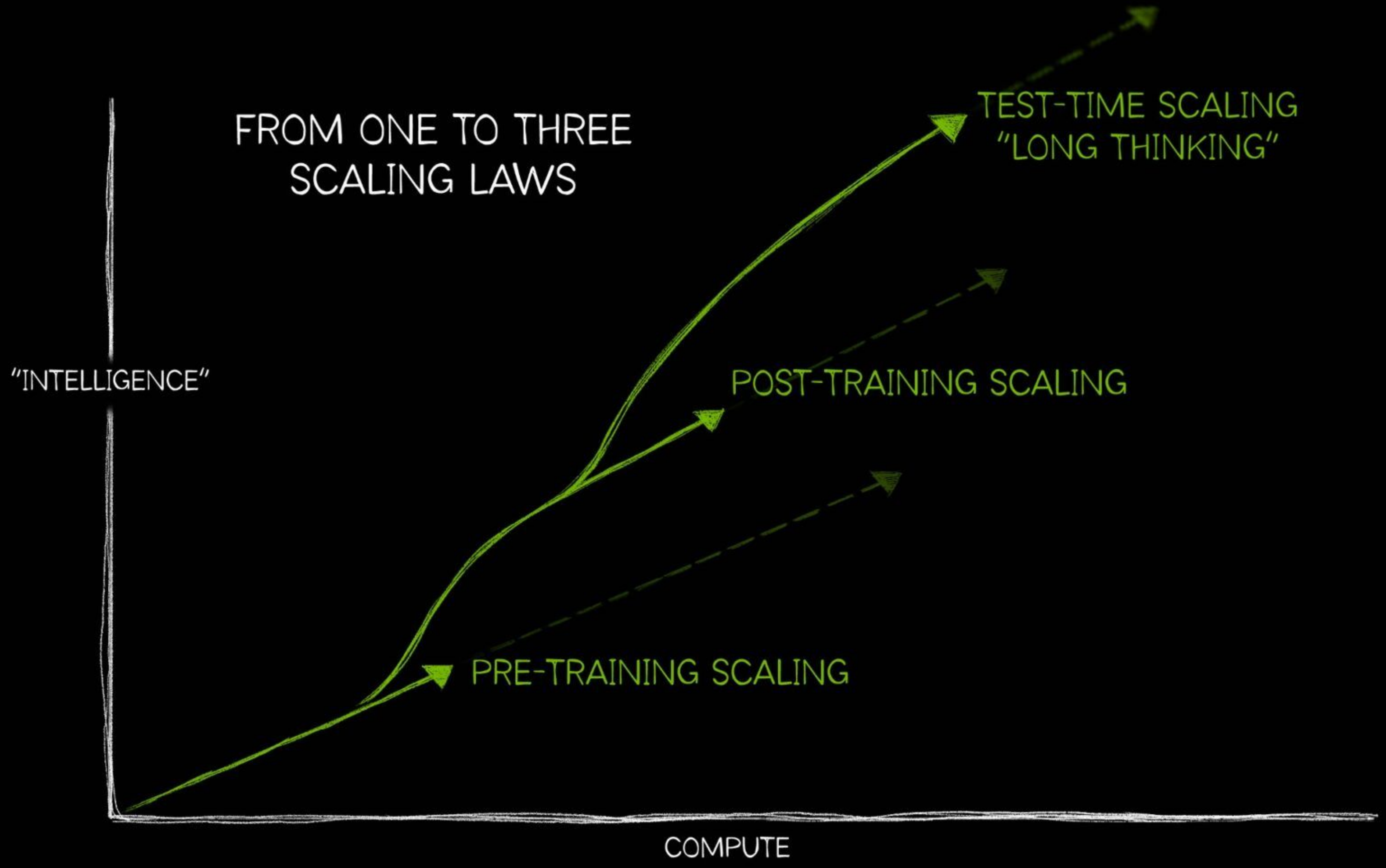
Scaling Laws 📈

Model performance improves predictably as model size, data, and compute increase - following a power-law trend

Patterns observed:

- More data → smoother loss curves, better generalization
- Larger models → capture richer structure, longer reasoning chains
- Emergent abilities appear only after certain scale thresholds
- But returns diminish - doubling data ≠ doubling capability

FROM ONE TO THREE SCALING LAWS



Why Next-Token Prediction Matters 🤔

The next token prediction task is extremely simple

Yet applied at massive scale the model implicitly learns many tasks

Grammar: “In my free time, I like to {run, banana}” → predicting “run”

World knowledge: “The capital of Japan is {Copenhagen, Tokyo}” → predicting “Tokyo”

Sentiment: “I was laughing the entire time, the movie was {good, bad}” → predicting “good”

Reasoning & Logic: “If it rains, the ground will be {wet, dry}.” → wet

Math & Symbolic Reasoning: “2 + 3 = {5, 6}” → 5

What the Model Learns 🧠

Syntax / Semantics / Factual knowledge / Basic reasoning capabilities / ……

Each next-token prediction is like solving a micro-task

- fill in a blank, finish a phrase, complete a thought

Across billions of sentences, the model encounters millions of implicit tasks - translation, analogy, classification, reasoning, summarization

The model does NOT memorize text; it models probability of the next token

- learning which continuations are plausible, logical, and coherent

Emergent capability 🌍

At sufficient scale, next-token prediction yields general intelligence-like behavior

- models can perform tasks they were never explicitly trained for, by transfer what they have learned from languages

This phenomenon is often summarized as:

- Pre-training teaches basic knowledge
- Supervised Fine-tuning enhance reasoning capabilities
- Reinforcement Learning aligns it

General Artificial Intelligence Step 0 🧠

Transformer + next-token objective + large data

→ General-purpose language model

Emergent abilities appear once enough scale + data + training

LLM Supervised Fine-tuning

From Pre-training to Fine-tuning 🌉

Pre-training: Learns general world knowledge from massive unlabeled text

- Models know how to speak but not what to do

Fine-tuning: Teaches behavior and task goals using labeled examples

- Bridge: Moves from “foundation knowledge” → “actionable skills”

Supervised Fine-tuning (SFT) 🧭

Train a pre-trained LLM on (prompt, response) pairs for a target task/behavior

Goal: Align the model to follow instructions and produce useful, reliable outputs

Setup:

- Freeze nothing (Full SFT) or lightly freeze (PET)
- Minimize token-level cross-entropy on target responses

Data: High-quality, diverse instructions with clear, verifiable targets

What Does the SFT Loss Do? 🤔

Cross-entropy loss = negative log probability of the correct next token

$$L = -\sum \log P(\text{correct\_token} \mid \text{prompt}, \text{previous\_tokens})$$

Each token in the reference answer contributes to the loss

If the model assigns low probability to a correct token

- higher loss → stronger learning signal

Standard SFT Paradigm

Example:

Prompt: “Summarize the paragraph.”

Response: “Climate change is harming coral reefs.”

Model learns to:

- Follow instructions
- Generate fluent, polite, and relevant text

Limitation: Learns what to say, **not** how to think

Why Modern LLMs Are “Thinking Models” 🧠

From Pattern Matching → Structured Reasoning

- Early LLMs mimicked language fluency
- Newer models learn to analyze, plan, and reason across multiple steps

Training Evolution

- Pre-training on vast, diverse data teaches latent world models
- SFT and RL explicitly train reasoning traces (“think step by step”)

Behavior Shift

- Models now explain reasons before answering
- Reasoning improves consistency, interpretability, and self-correction

Thinking SFT is Step 1

Standard SFT = aligns immediate behavior (instruction-following)

Thinking SFT = aligns reasoning (how the model reaches conclusions)

Without reasoning signals, models learn to “answer fast” instead of “think slow”

Goal: expose the model to step-by-step problem solving, not just end results

💡 We are teaching the model to show its work - not just the answer

What Are “Thinking Tasks”?

Tasks where intermediate reasoning is essential, not optional

Example 1 - Math / Logic

Prompt: “What is 27 × 14?”

Reasoning: “27 × 10 = 270, 27 × 4 = 108, 270 + 108 = 378.”

Answer: “378.”

Example 2 - Everyday Reasoning

Prompt: “Why do we wear jackets in winter?”

Reasoning: “Because jackets trap body heat, keep us warm when the air is cold.”

Answer: “To stay warm.”

Example 1 – Plain SFT 🧱

Prompt: “Solve: 12 + 7 = ?”

Response: “19.”

Loss is computed only on the final tokens “1”, “9”, “.”

→ The model learns to produce correct answers and follow format

→ It does not learn how to reason

Example 2 – Reasoning-augmented SFT

Prompt: “Solve: 12 + 7 = ?”

Response: “Let’s think step by step. 12 + 7 = 19. Answer: 19.”

Now the loss covers every token, including

“Let’s”, “add”, “step”, “by”, “step”, “12”, “+”, “7”, “=”, “19”…

→ The model learns that generating intermediate reasoning steps reduces the overall loss

→ The behavior “explain before answering” becomes part of training

Example 3 – Weighting by Reasoning Quality

Prompt: “Solve: 12 + 7 = ?”

Sample	Text	Correct?
1	“I think 12 + 10 = 22”	❌
2	“12 + 7 shall be 19”	✅

Loss can be weighted across samples:

$$L = w_1 \times \mathrm{CE}(\mathrm{sample}_1) + w_2 \times \mathrm{CE}(\mathrm{sample}_2)$$

Only correct or verified samples receive higher weight

→ The model learns which reasoning paths are trustworthy

Where to Source Thinking Data ⚙️

Math / Logic: GSM8K, MATH, AQUA-RAT, SVAMP

Reasoning QA: StrategyQA, HotpotQA, OpenBookQA

Step-by-step: CoT (Chain-of-Thought) traces from Flan, CotHub, OpenOrca

Code Reasoning: CodeContests, LeetCode-HF, DeepMind MathCode datasets

Human-curated: Collect instructor-written rationales or worked solutions

Synthetic Generation: Use an existing LLM to produce reasoning traces

(“Let’s think step by step…” → label + verify)

Filter by correctness or self-consistency

How to Curate and Format 📚

Keep structure consistent

- Prompt: [problem]
- Reasoning: [chain of thought]
- Answer: [final result]

Keep reasoning concise

- 2–6 reasoning steps preferred
- Each step should be verifiable (not vague “I think maybe...”)

How to Curate and Format 📚

Keep correctness

- Human or model verification for every rationale
- Discard hallucinated or logically invalid traces

Keep diversity

- Mix numeric, symbolic, commonsense, procedural reasoning
- Include both short and long-context tasks

Beyond Single-task SFT 🌐

Multi-task fine-tuning: Train on many instruction types simultaneously

Mixture of formats: QA + translation + summarization + more → boosts skills

Instruction format standardization: Convert all tasks to prompt-response pairs

Results: Better zero-shot transfer and alignment readiness

Industry trends

Looking Ahead

The pre-training + fine-tuning paradigm keeps evolving rapidly

New research improves efficiency, scalability, and capability

Understanding these trends helps you stay ahead of the LLM curve

Focus today:

- Efficiency
- Multimodality
- Domain adaptation

Parameter-efficient Fine-tuning (PEFT) ⚙️

Challenge: Full fine-tuning = billions of parameters → expensive & slow

Idea: Freeze the base model; only train small additional modules

Goal: Reduce compute + memory footprint while keeping performance

Result: Same model reused across many downstream tasks efficiently

LoRA: Low-Rank Adaptation 💾

Key idea: Instead of updating full weight matrix W, learn 2 small matrices A & B s.t.

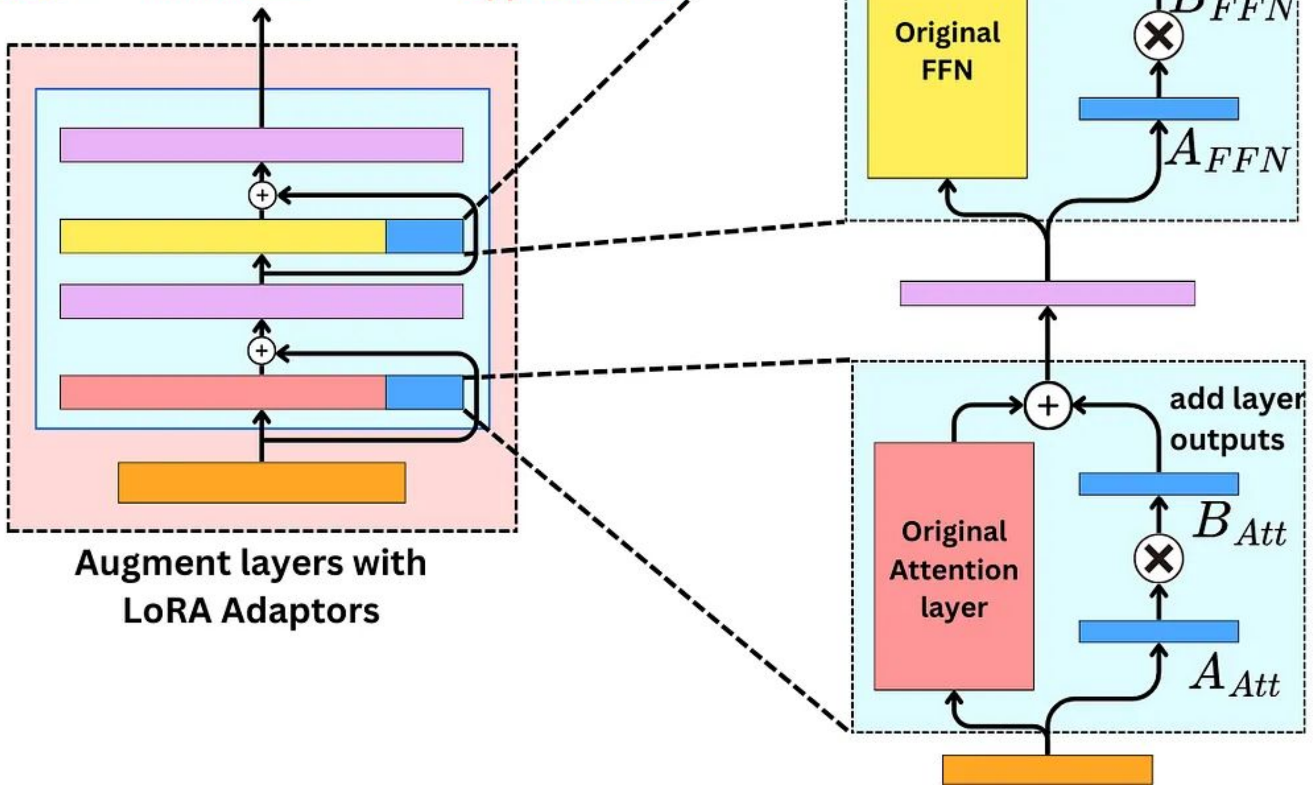
$$\Delta W = A \times B^T \text{ (low-rank update)}$$

Benefits:

- ~1–2% trainable parameters
- Easy to merge/unmerge with base model
- Great for multi-task learning and low-data scenarios

Forward pass

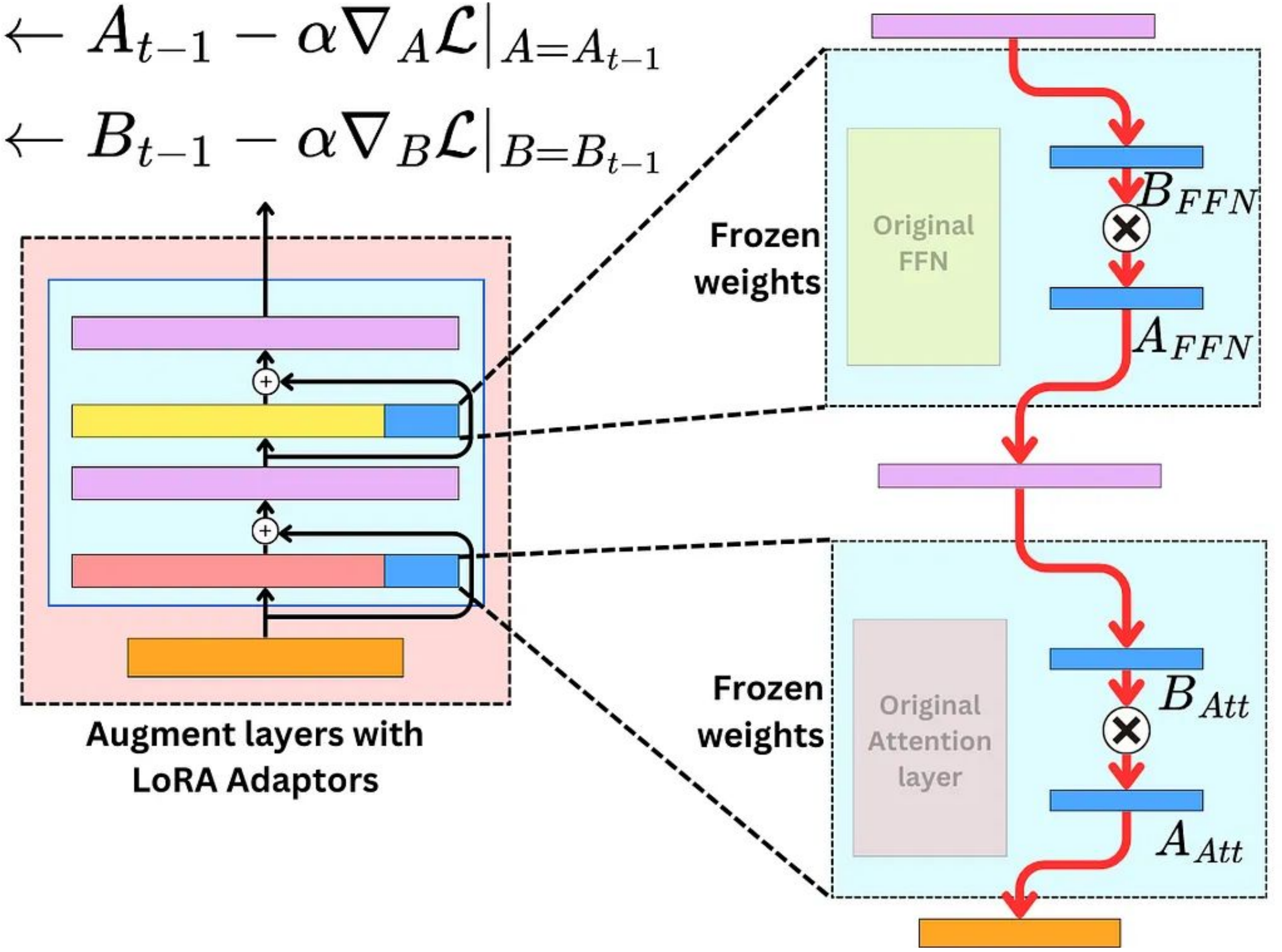
$$\theta_F = \underbrace{\boxed{\theta_T}}_{\text{The trained model}} + \underbrace{\boxed{\Delta W}}_{\text{The fine-tuning data}} \simeq \theta_T + \underbrace{\boxed{B^T A}}_{\text{Low-Rank approximation}}$$



$$A_t \leftarrow A_{t-1} - \alpha \nabla_A \mathcal{L}|_{A=A_{t-1}}$$

$$B_t \leftarrow B_{t-1} - \alpha \nabla_B \mathcal{L}|_{B=B_{t-1}}$$

Backward pass



QLoRA: Quantized LoRA ⚡

Problem: Even LoRA can be memory-heavy for >30B models

Quantize frozen/trainable weights to 4-bit precision → only train LoRA layers

Outcome: Fine-tune 65B models on a single 48GB GPU 😺

Used in: almost all open-source chat models

Multimodal Pre-training 🖼️

Text + Image → CLIP

Contrastive objective aligns visual & textual embeddings

Text + Vision + Audio → Flamingo, Gemini, ChatGPT

Cross-modal attention for unified reasoning

Why It Matters:

Language models are expanding beyond text - “seeing” and “hearing”

Ongoing Effort: Unified foundation models across modalities

Domain Adaptation 🌐

Tailor models to specialized domains (e.g., law, finance, biomedicine)

Approach: Further tuning (SFT/RL) on domain-specific corpora

Examples:

- CodeLlama → trained on github data to deal with coding problems
- BloombergGPT → trained on financial news, and market data to handle finance-specific tasks (analysis, risk, compliance)
- Med-PaLM / Meditron → fine-tuned on medical QA datasets for clinical reasoning

Impact: Improves factual accuracy and domain reasoning

Summary

Pre-training: foundation of general intelligence

SFT: teaches forcing behavior and step 1 of thinking LLMs

Emerging trends: efficiency, multimodality, domain focus

Mindset shift: Foundation models are not endpoints - but platforms

Next Lecture: Reinforcement Learning for “AGI” 🤔