Google Developer Group
Chandigarh University

Gemini Study Jams

Fine Tuning Basics & Supervised Fine Tuning

What will we do today?

1. What is Fine Tuning?
2. Examples of Fine Tuning
3. Why Fine Tuning?
4. When Fine Tuning?
5. When NOT Fine Tuning?
6. Types of Fine Tuning
7. What is Supervised Fine Tuning?
8. SFT Visualization
9. Quiz Time
10. Let’s Code!
11. Afterword

What is Fine Tuning?

1. Fine-tuning is the process of taking a pretrained machine-learning model and training it further on a smaller, task-specific dataset so it performs better on a particular domain or task.
2. A LLM is a behemoth with (in case of transformer architecture) **multiple overlapping attention heads**. Each task triggers specific heads to produce an output.
3. This is determined by probability. **Each completion token has a probability distribution of possible next words** that it can use.

What is Fine Tuning?

1. Fine Tuning nudges those probabilities so that

- Tokens fitting the desired style
- Tokens fitting the desired format
- Tokens fitting the desired persona
- Tokens fitting the desired task behavior receive higher probabilities.

2. For example, if we wanted the model to output riddles in the form of poems then the model’s next-token distribution given such an instruction will increasingly favor:

- A leading quotation mark
- Poetic syntax
- Line breaks
- Rhythmic cadences
- Metaphoric language

“Fine-tuning teaches the model *how you want it to behave* by repeatedly showing it *examples* of that behavior.”

Examples of Fine Tuning

1. Domain-specific Classification
   - **Medical triage classification** (e.g., “emergent / urgent / routine” based on clinical notes)
   - **Legal document issue tagging** (e.g., “improper service”, “breach of fiduciary duty”, “quiet title”)
2. Style Constrained Generation
   - **Product description templates** (“Title → 3 bullet benefits → care instructions → SKU format”)
   - **Brand-voice constrained writing** (e.g., a fashion brand’s ultra-specific tone)
3. Structured Data Extraction
   - Extracting **entities and attributes** from invoices with rare field names
   - Parsing **internal engineering logs** with weird formats

Why Fine Tuning?

1. **Makes custom behavior the default:** the model *acts* the way you want without long prompts (formatting, tone, rules).
2. **Teaches new domain knowledge:** the model actually *learns* your proprietary jargon, labels, or concepts (which prompting/RAG can only describe).
3. **Delivers high consistency:** outputs follow strict structures or templates with far fewer errors or hallucinations.
4. **Improves efficiency:** shorter prompts → lower latency and cost compared to RAG or prompt-heavy pipelines.
5. **Enforces reliable policies:** compliance, safety, or procedural rules can be baked into the model so they cannot be ignored.

When Fine Tuning?

1. **You need the model to behave in a *very specific way by default.*** If your system requires consistent formatting, tone, structure, or wording **every single time** without relying on long prompts → fine-tune.
2. **You need *high consistency* on a narrow task.** If the task is repeatable and the tolerance for variation is low (structured extraction, legal letters, template generation) → fine-tune.
3. **You need *low latency* or low cost.** Fine-tuned models can run with *tiny prompts*, unlike RAG or complex prompting chains.
4. **You need *enforceable rules or behavior.*** Compliance, safety, brand voice, internal policies that must never be violated.

When NOT Fine Tuning?

1. **When the goal is “more knowledge” or “up-to-date facts”.** Fine-tuning **cannot** reliably add new factual knowledge, especially if it changes often.
2. **When the task requires broad reasoning or general intelligence.** Fine-tuning **will not make the model “smarter.”**
3. **When you only want minor stylistic adjustments.** If the change is small and can be done with a prompt → fine-tuning is overkill.
4. **When the data is too small or noisy.** Fine-tuning with poor or tiny datasets leads to **degradation**, not improvement.
5. When prompting or function calling already works well

SFT

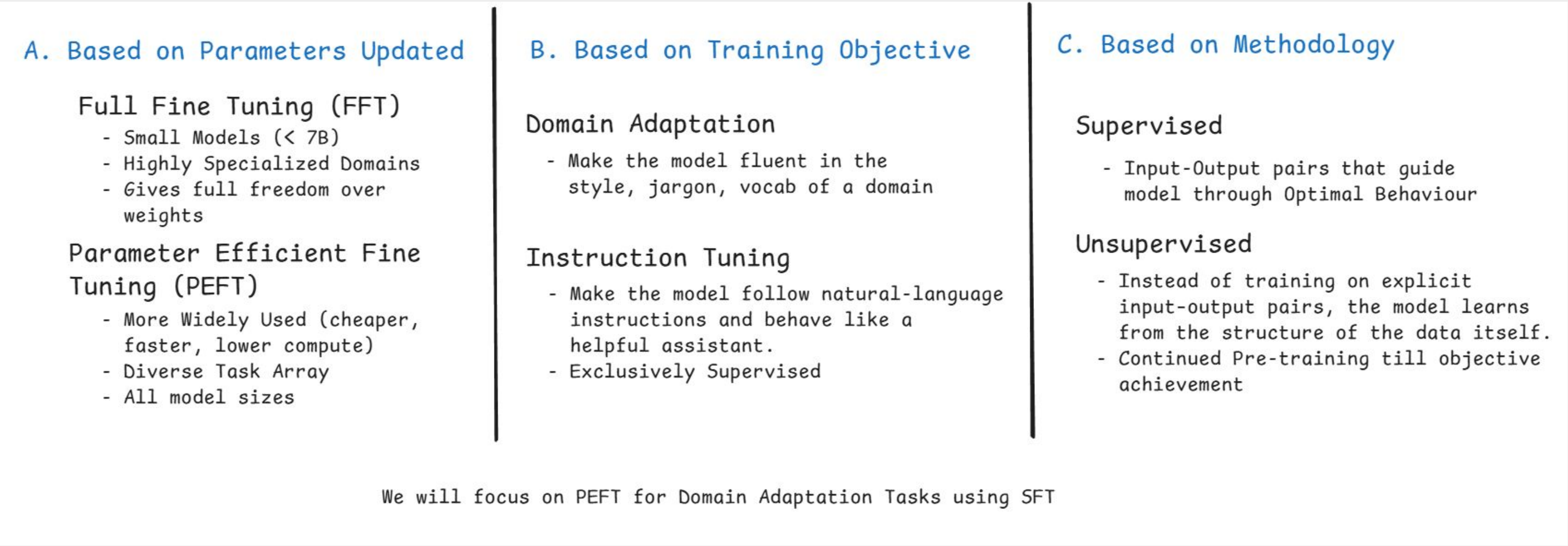
Supervised Fine Tuning is the process of taking a pre-trained model (e.g., GPT, LLaMA, BERT) and **fine-tuning it on labeled examples**.

SFT is used to:

- teach a model new skills
- align behavior to instructions
- adapt to a specific domain (law, medicine, finance, coding, etc.)
- refine tone, style, or personality

SFT is typically the *first* stage in modern model training pipelines (e.g., pre training → SFT → RLHF/DPO → iterative alignment/training).

Types of Fine Tuning



A. Based on Parameters Updated

Full Fine Tuning (FFT)

- Small Models (< 7B)
- Highly Specialized Domains
- Gives full freedom over weights

Parameter Efficient Fine Tuning (PEFT)

- More Widely Used (cheaper, faster, lower compute)
- Diverse Task Array
- All model sizes

B. Based on Training Objective

Domain Adaptation

- Make the model fluent in the style, jargon, vocab of a domain

Instruction Tuning

- Make the model follow natural-language instructions and behave like a helpful assistant.
- Exclusively Supervised

C. Based on Methodology

Supervised

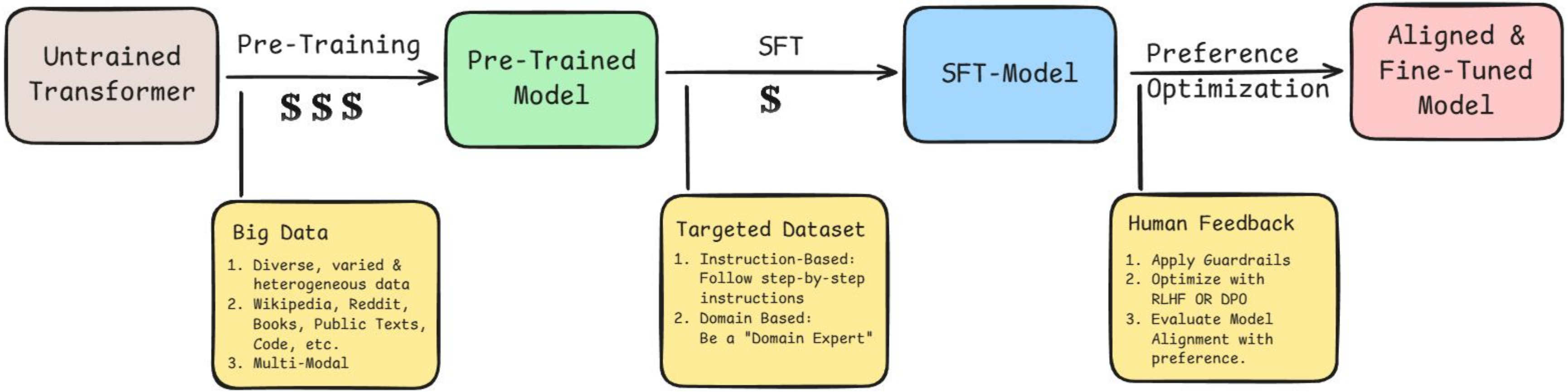
- Input-Output pairs that guide model through Optimal Behaviour

Unsupervised

- Instead of training on explicit input-output pairs, the model learns from the structure of the data itself.
- Continued Pre-training till objective achievement

We will focus on PEFT for Domain Adaptation Tasks using SFT

SFT Visualization



Big Data

1. Diverse, varied & heterogeneous data
2. Wikipedia, Reddit, Books, Public Texts, Code, etc.
3. Multi-Modal

Targeted Dataset

1. Instruction-Based: Follow step-by-step instructions
2. Domain Based: Be a "Domain Expert"

Human Feedback

1. Apply Guardrails
2. Optimize with RLHF OR DPO
3. Evaluate Model Alignment with preference.

Quiz Time!

Let’s Code!

Doubts?

Ask away!

Google Developer Group
Chandigarh University

Thank You!

Fine Tuning Basics & Supervised Fine Tuning - End