🔑 Core Fine-Tuning Approaches

1. Full Fine-Tuning

- What it is: Update *all* parameters of the base model on your domain dataset.
- Pros: Maximum adaptation, can deeply specialize the model.
- Cons: Extremely compute-heavy (billions of parameters), risk of catastrophic forgetting.
- Use case: When you have huge domain-specific data (e.g., medical corpora, legal texts).

---

2. Feature Extraction (Classifier Head Training)

- What it is: Freeze the base model, train only a small classifier layer on top.
- Pros: Lightweight, fast, avoids overfitting.
- Cons: Limited adaptation, mostly useful for classification tasks.
- Use case: Sentiment analysis, spam detection, intent classification.

⚡ Parameter-Efficient Fine-Tuning (PEFT)

These are the most practical techniques today because they adapt LLMs without retraining billions of weights.

3. LoRA (Low-Rank Adaptation)

- How it works: Inject trainable low-rank matrices into attention layers.
- Pros: Updates <1% of parameters, very efficient.
- Cons: Slightly less expressive than full fine-tuning.
- Use case: Enterprise chatbots, policy Q&A, domain-specific assistants.

---

4. Adapters

- How it works: Add small neural modules ("adapter layers") between transformer blocks.
- Pros: Modular—can swap adapters for different domains.
- Cons: Slightly higher inference latency.
- Use case: Multi-domain systems (finance adapter, healthcare adapter, etc.).

---

5. Prefix Tuning / Prompt Tuning

- How it works: Learn a set of "soft prompts" (trainable embeddings) prepended to inputs.
- Pros: Extremely lightweight, only a few thousand parameters.
- Cons: Works best for narrow tasks, less generalization.
- Use case: Task-specific bots (e.g., customer support for one product line).

---

6. BitFit

- How it works: Fine-tune only bias terms in the model.
- Pros: Very simple, tiny parameter updates.
- Cons: Limited capacity, weaker than LoRA/adapters.
- Use case: Quick adaptation when compute is very constrained.

🧭 Advanced Alignment Techniques

7. Instruction Fine-Tuning

- How it works: Train on prompt–response pairs to make the model follow instructions better.
- Pros: Improves usability, aligns with human expectations.
- Cons: Needs high-quality curated datasets.
- Use case: General-purpose assistants, enterprise copilots.

---

8. RLHF (Reinforcement Learning from Human Feedback)

- How it works:
    1. Collect human preference data.
    2. Train a reward model.
    3. Optimize the LLM to maximize reward.
- Pros: Aligns outputs with human values/preferences.
- Cons: Expensive, requires human annotators.
- Use case: Safety-critical systems, public-facing assistants.

---

9. DPO (Direct Preference Optimization)

- How it works: Directly optimize on preference pairs without training a separate reward model.
- Pros: Simpler than RLHF, less compute.
- Cons: Still requires preference data.
- Use case: Aligning enterprise bots with compliance rules.

📊 Comparison Table

Technique	Params Updated	Efficiency	Best Use Case
Full Fine-Tuning	100%	❌ Heavy	Deep domain specialization

Technique	Params Updated	Efficiency	Best Use Case
Feature Extraction	Few (head)	✅ Light	Classification tasks
LoRA	<1%	✅✅ High	Domain adaptation
Adapters	Small modules	✅ High	Multi-domain systems
Prefix/Prompt Tuning	Few thousand	✅✅ Very High	Narrow tasks
BitFit	Bias terms	✅✅ Very High	Quick adaptation
Instruction Tuning	Varies	✅ Medium	General assistants
RLHF	Many	❌ Heavy	Safety alignment
DPO	Moderate	✅ Medium	Compliance alignment

🚀 Enterprise Perspective (Your Context)

- LoRA + RAG → Best combo for enterprise AI. LoRA adapts tone/style, RAG ensures factual grounding.
- Adapters → Useful if you want modular AI agents (finance adapter, HR adapter, etc.).
- Instruction tuning → Critical for Salesforce Agentforce prep, since enterprise copilots must follow structured instructions.
- RLHF/DPO → Needed when deploying customer-facing assistants with compliance and safety requirements.