# Fine-Tuning Large Language Models: Supervision, Instructions, and Parameter-Efficient Methods

October 25, 2025

## 1 Supervised Fine-Tuning (SFT)

### 1.1 Objective Formulation and Loss Functions

Supervised fine-tuning adapts a pretrained language model to downstream tasks using labeled examples:

- **Conditional generation:** Tasks such as summarization, translation, and code generation optimize autoregressive cross-entropy while conditioning on task-specific prompts.
- **Discriminative heads:** Classification, retrieval, or scoring tasks append lightweight projection layers, often trained with cross-entropy, contrastive, or margin-based losses.
- **Multi-task fusion:** Weighted multi-task schemes or shared decoders support diverse objectives (QA, reasoning, tool invocation) without degrading base capabilities.

Regularization techniques—label smoothing, Mixout, R-Drop, stochastic depth—mitigate overfitting in data-scarce regimes. Learning rate schedules typically feature short warmup and cosine decay, with lower rates for backbone layers to avoid catastrophic forgetting.

### 1.2 Data Engineering and Curation

High-quality labeled data are essential for successful SFT:

- **Task schema design:** Define canonical input/output formats, including context windows, delimiters, and expected reasoning style.
- **Human-in-the-loop workflows:** Combine model-generated drafts with expert review, employ double-annotation and adjudication to track agreement.
- **Augmentation and balancing:** Apply paraphrasing, counterfactual editing, or knowledge expansion to diversify samples while preserving edge cases.

Versioned datasets with metadata (annotator ID, difficulty, quality score) provide transparency and facilitate post-hoc audits of model behavior.

### 1.3 Training Protocols and Evaluation

SFT runs are usually short and seek stability:

- **Layer-wise learning rates:** Freeze or partially update lower transformer blocks while training task adapters with larger step sizes.
- **Gradient clipping and accumulation:** Stabilize updates across heterogeneous examples, especially when batch sizes are constrained by context length.
- **Comprehensive evaluation:** Track automatic metrics (accuracy, BLEU/ROUGE, perplexity) alongside manual reviews for factuality, safety, and adherence to guidelines.

# 2 Instruction Tuning (Chat Tuning)

## 2.1 Instruction Dataset Construction

Instruction tuning exposes models to varied prompts and desired responses so they follow human intent. Key dataset characteristics:

- **Task diversity:** Incorporate classification, extraction, reasoning, tool use, code generation, mathematics, and safety-sensitive instructions to enhance generalization.
- **Role conditioning:** Embed personas (teacher, lawyer, doctor) and context constraints that teach the model to modulate tone and register.
- **Alignment behaviors:** Include examples of compliance, clarification questions, refusals, and chain-of-thought reasoning.

Public resources like FLAN, Super-Natural Instructions, Self-Instruct, and OpenOrca serve as common starting points.

## 2.2 Semi-Automatic Expansion and Quality Control

Large-scale instruction datasets rely on model-assisted generation:

- **Self-Instruct pipelines:** Seed instructions prompt the model to invent new tasks; human reviewers filter and refine the outputs.
- **Adversarial and refusal prompts:** Purposefully generate unsafe, malicious, or policy-violating instructions to teach the model when and how to refuse.
- **Challenging sample mining:** Analyze failure cases to synthesize targeted follow-up instructions, improving robustness in weak areas.

Quality dashboards track average response length, citation rate, refusal accuracy, and coverage of policy domains to maintain dataset health.

## 2.3 Conversational Tuning and Context Handling

Chat tuning adapts models for multi-turn dialogue:

- **System prompts and memory:** Prepend system messages to impose behavior guidelines and maintain persistent goals across turns.
- **Dialogue compression:** Summaries, semantic caches, or retrieval-augmented history keep long conversations within context limits.
- **Safety alignment:** Reinforcement learning from human feedback (RLHF), iterative preference optimization (DPO), and red-teaming tests ensure appropriate refusals and de-escalation.

Evaluation includes conversation success rate, turn-level coherence, safety trigger rates, and user satisfaction scores.

# 3 Prompt Templates and System Roles (ChatML, Alpaca Format)

## 3.1 Template Design Principles

Prompt templates establish the structure that models rely on:

- **Explicit role demarcation:** Formats such as ChatML use tokens like <|system|>, <|user|>, <|assistant|> to delineate participants.

- **Instruction-input-output separation:** The Alpaca format clarifies task instructions, optional inputs, and expected outputs, reducing ambiguity.
- **Style and constraint injection:** Templates embed formatting requirements, safety reminders, or style guides to steer generations.

### 3.2 Template Management Across Tasks

Maintaining consistent prompting across training and inference is crucial:

- **Optional fields:** Support tasks without inputs, as well as rich metadata (language, tone, length) for complex workflows.
- **Programmatic rendering:** Template engines convert structured data to prompts, reducing human error and enabling large-scale dataset generation.
- **Evaluation parity:** Use identical templates during fine-tuning, offline evaluation, and deployment to prevent distribution shift.

### 3.3 System Roles and Safety Guardrails

System prompts anchor model behavior:

- **Behavioral policies:** Outline priorities, factuality requirements, and safety protocols that govern interaction.
- **Tool usage instructions:** Describe available functions, input/output schemas, and error handling procedures for tool-augmented agents.
- **Persona switching:** Prepare specialized system messages for customer support, creative writing, or coding assistants to enable scenario-specific responses.

Designers must guard against user prompts that attempt to override system instructions; hierarchical prompt parsing and refusals provide defense-in-depth.

## 4 Parameter-Efficient Fine-Tuning (PEFT: LoRA, QLoRA, Prefix-Tuning)

### 4.1 LoRA and Low-Rank Adaptation

LoRA introduces trainable low-rank matrices that modulate existing weights:

- **Mechanism:** Decompose updates as $W' = W + BA$, where $A \in \mathbb{R}^{r \times d}$ and $B \in \mathbb{R}^{k \times r}$; only $A$ and $B$ are trained.
- **Benefits:** Cuts trainable parameters by orders of magnitude, enables fast task switching by swapping LoRA adapters, and works seamlessly with mixed precision.
- **Deployment modes:** Adapters can be merged into base weights for inference or loaded on demand for modular serving.

### 4.2 QLoRA and Quantization-Aware Fine-Tuning

QLoRA combines LoRA with low-bit quantization to fit large models on commodity hardware:

- **Quantization scheme:** NF4 (Normalized Float 4) representation preserves magnitude information with minimal error; double quantization reduces storage overhead.

- **Paged optimizers:** Optimizer states reside in CPU memory or NVMe, streaming chunks to GPUs to stay within VRAM limits.
- **Training setup:** 4-bit weights paired with 16-bit activations and FP16/FP32 gradient accumulation maintain stability and accuracy.

### 4.3 Prefix/Prompt Tuning and Modular Control

Prefix-based methods adapt models without modifying core weights:

- **Prefix-Tuning:** Learn key-value vectors prepended to attention layers, steering generation while freezing the backbone.
- **P-Tuning v2:** Introduce trainable virtual tokens at the embedding layer, scaling to deep architectures with parameter sharing.
- **Hybrid strategies:** Combine prefixes with LoRA adapters or router mechanisms that select task-specific prompts dynamically.

PEFT enables rapid deployment of multiple personas or domains from a single base model. Post-training evaluation should compare full fine-tuning and PEFT baselines on accuracy, robustness, and safety metrics.

## Further Reading


- Wei et al. "Finetuned Language Models Are Zero-Shot Learners." ICLR, 2022.
- Chung et al. "Scaling Instruction-Finetuned Language Models." arXiv, 2022.
- Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models." ICLR, 2022.
- Dettmers et al. "QLoRA: Efficient Finetuning of Quantized LLMs." arXiv, 2023.
- Lester et al. "The Power of Scale for Parameter-Efficient Prompt Tuning." EMNLP, 2021.