International Journal of Scientific Research in Engineering & Technology
Volume6, Issue2 (March-April 2026), PP: 38-49.
https://www.doi.org/10.59256/ijsreat.20260602006
www.ijsreat.com
ISSN No: 2583-1240

A Comprehensive Survey of LLM Fine-Tuning: From Foundations to Frontier Techniques

Milind k.Patil
Syncaissa Systems Inc. USA.

To Cite this Article: *Milind k.Patil, "A Comprehensive Survey of Llm Fine-Tuning: From Foundations to Frontier Techniques", International Journal of Scientific Research in Engineering & Technology, Volume 06, Issue 02, March-April 2026, PP: 38-49.*

Abstract: *This paper presents a comprehensive technical survey of large language model (LLM) fine-tuning, spanning the complete methodological landscape from foundational techniques to frontier advances as of early 2026. We organize the field along two orthogonal axes: the training objective—what the model learns (SFT, DPO, RLHF, GRPO, ORPO, SimPO, and KTO)—and the parameter-update strategy—how weights are modified (Full Fine-Tuning, LoRA, QLoRA, DoRA, GaLore, Spectrum). We trace the theoretical evolution from classical RLHF with its three-model PPO pipeline, through the DPO reparameterization that collapsed preference learning into a single supervised objective, to the reasoning-focused GRPO/RLVR paradigm that enabled DeepSeek-R1 to achieve 71.0% Pass@1 on AIME 2024 through emergent reasoning without supervised reasoning traces. The paper further provides rigorous treatment of model merging techniques (TIES, DARE, and SLERP) that compose capabilities in weight space without gradient computation, knowledge distillation methods including cross-tokenizer and comparative approaches, Mixture-of-Experts fine-tuning with sparse routing, multimodal adaptation of vision-language models, and the modern framework ecosystem. We present formal loss functions, convergence properties, and computational complexity analysis for each method, accompanied by empirical benchmark comparisons. The survey concludes with a unified taxonomy and actionable guidance for selecting technique combinations based on compute budget, data availability, and task requirements.*

Key Words: *fine-tuning, LoRA, QLoRA, DoRA, DPO, RLHF, GRPO, RLVR, SFT, ORPO, SimPO, KTO, model merging, TIES, DARE, knowledge distillation, parameter-efficient fine-tuning, large language models, alignment, mixture of experts.*

1. INTRODUCTION AND MOTIVATION

The emergence of large language models (LLMs) with billions of parameters—GPT-4 [14], Claude, Llama-3, DeepSeek-R1 [4]—has demonstrated that pretraining on internet-scale corpora produces powerful general-purpose representations. However, the pretrained model is a starting point, not a finished artifact. Adapting these foundation models to specific domains, behavioral constraints, and deployment environments requires post-training—the family of techniques that modify learned parameters after the pretraining phase.

The post-training landscape has undergone rapid expansion since 2022. What began as a two-stage pipeline (SFT followed by RLHF) has evolved into a rich ecosystem of composable methods spanning at least five distinct dimensions: training objective, parameter-update strategy, data format, reward signal design, and model composition. Practitioners now face a combinatorial decision space that this survey aims to systematize.

Post-training is governed by two orthogonal axes: the training objective (what the model learns: SFT, DPO, RLHF, GRPO) and the parameter-update strategy (how weights change: Full, LoRA, QLoRA, DoRA). These axes are independent—any objective can be paired with any update strategy. Understanding this decomposition is the key to navigating the field.

The Post-Training Stack

We distinguish five layers of model customization, each operating at a different abstraction level with fundamentally different computational properties (Table 1).

Layer	Name	What Changes	Cost	Persistent?
1	Pretraining	All weights (from scratch)	$50M+	Yes
2	Post-Training (SFT/DPO/GRPO)	Selected or all weights	$5–$10K+	Yes
3	Serving Configuration	Zero weights	Free	Session
4	System Prompt	Zero weights (runtime)	Free	Session
5	User Prompt + In-Context Learning	Zero weights (per query)	Free	Transient

The Five Layers of LLM Customization

Only Layer 2 constitutes genuine fine-tuning—permanent modification of the model's parameter space. Layers 3–5 operate within the model's existing representational capacity without gradient-based updates. This distinction is critical: runtime prompting cannot teach the model new knowledge that lies outside the distribution of its pretrained representations, whereas fine-tuning modifies the distribution itself.

The Two-Axis Decision Matrix

	Full Fine-Tune	**LoRA**	**QLoRA**	**DoRA**
	(≥80GB)	(~16GB)	(~6GB)	(~18GB)
SFT	SFT + Full	SFT + LoRA	SFT + QLoRA	SFT + DoRA
DPO	DPO + Full	DPO + LoRA	DPO + QLoRA	DPO + DoRA
RLHF	RLHF + Full	RLHF + LoRA	RLHF + QLoRA	RLHF + DoRA
GRPO	GRPO + Full	GRPO + LoRA	GRPO + QLoRA	GRPO + DoRA
ORPO	ORPO + Full	ORPO + LoRA	ORPO + QLoRA	ORPO + DoRA
KTO	KTO + Full	KTO + LoRA	KTO + QLoRA	KTO + DoRA

The Post-Training Decision Matrix: Every Cell Is a Valid Combination

Supervised Fine-Tuning (SFT)

SFT is the canonical first post-training step, transforming a base model's next-token prediction behavior into instruction-following capability [14].

Formal Objective

Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ of instruction–response pairs, SFT minimizes the negative log-likelihood of target tokens conditioned on the instruction:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{i=1}^{N}\sum_{t=1}^{|y_i|} \log p_\theta(y_{i,t} \mid x_i, y_{i,<t})$$

where θ denotes model parameters, $y_{i,t}$ is the t-th token of response y_i, and conditioning on $y_{i,<t}$ denotes autoregressive generation. The loss is computed only over response tokens; instruction tokens are masked.

Data Efficiency and Scaling Properties

A key empirical finding is that SFT data quality dominates quantity. Ouyang et al. demonstrated that InstructGPT with ~13,000 carefully curated demonstrations dramatically outperformed models trained on orders of magnitude more low-quality pairs. Zhou et al. (2023) showed that as few as 1,000 high-quality examples can align a 65B model to match the instruction-following ability of models trained on 52K examples (the LIMA result). For domain-specific applications, 500–2,000 expert-written examples typically suffice, provided they cover the target task distribution and exhibit reasoning diversity.

Limitations

SFT is bounded by the *demonstration ceiling*: the model can only learn to reproduce patterns present in the training data. It cannot discover superior strategies through exploration—a fundamental limitation addressed by reinforcement learning methods (Section 4).

Preference Alignment: From RLHF to DPO and Beyond

After SFT, models follow instructions but may exhibit undesirable behaviors: verbosity, sycophancy, refusal of benign requests, or hallucination with high confidence. Preference alignment teaches the model *which* of its possible responses humans prefer.

RLHF: Reinforcement Learning from Human Feedback

The classical approach, pioneered by Ouyang et al. [14], involves three stages:

1. **Comparison collection**: Human annotators rank model outputs pairwise—$y_w \succ y_l$ given prompt x—producing a preference dataset $\mathcal{D}_{\text{pref}}$.
2. **Reward model training**: A separate neural network $r_\phi(x, y) \in \mathbb{R}$ is trained via the Bradley-Terry objective:
$$\mathcal{L}_{\text{RM}}(\phi) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_{\text{pref}}}\left[\log\sigma\left(r_\phi(x, y_w) - r_\phi(x, y_l)\right)\right]$$
3. **Policy optimization via PPO**: The LLM policy π_θ is trained to maximize reward while staying close to the reference policy π_{ref}:
$$\max_\theta \ \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[r_\phi(x, y) - \beta\,\text{KL}(\pi_\theta(\cdot\,|x) \,\|\, \pi_{\text{ref}}(\cdot\,|x))\right]$$

Computational and Stability Challenges

RLHF requires three models in GPU memory simultaneously: the active policy π_θ, the reward model r_ϕ, and the frozen reference π_{ref} (for KL regularization). This triples memory requirements. PPO [15] is notoriously sensitive to hyperparameters—learning rate, clipping ratio ε, GAE λ, number of rollout steps—and susceptible to reward hacking: the policy discovers high-reward degenerate outputs that exploit reward model artifacts without genuine quality improvement.

Model	Pretraining	SFT	Alignment	Year
GPT-4			RLHF	2023
Claude 3			Constitutional AI + RLHF	2024
Llama-3-Instruct			DPO	2024
Mistral-Instruct			DPO	2024
DeepSeek-R1			GRPO	2025
Qwen-2.5			DPO + GRPO	2025

Alignment Methods Used in Production Models

DPO: Direct Preference Optimization

DPO [3] was the key theoretical breakthrough that made preference alignment tractable. Rafailov et al. showed that the optimal policy under the RLHF objective can be expressed in closed form, enabling direct optimization without a reward model.

Theoretical Foundation

The key insight is that the optimal reward function under the KL-constrained objective admits an analytical solution:

$$r^*(x,y) = \beta \log \frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$$

Substituting into the Bradley-Terry preference model yields the DPO loss:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]$$

where β controls the strength of the KL constraint. This transforms RLHF from a three-model RL problem into a single-model supervised learning problem with a binary cross-entropy–like loss.

Computational Advantages

DPO requires only two forward passes per training step (chosen and rejected), compared to PPO's rollout sampling, reward computation, advantage estimation, and multi-epoch updates. Memory drops from 3 × to 2 × base model size (active policy + frozen reference). Training is stable without PPO's hyperparameter sensitivity.

Post-DPO Preference Methods

DPO's success catalyzed a rapid succession of improvements, each addressing a specific theoretical or practical limitation (Table 4).

Method	Year	Venue	Key Innovation
RLHF/PPO	2022	NeurIPS	Reward model + RL training; three-model pipeline
DPO	2023	NeurIPS	Reparameterizes reward; eliminates RM and RL entirely
IPO	2024	Google	Fixes DPO's unbounded implicit reward gap via regularization
KTO	2024	arXiv	Binary (unpaired) feedback via prospect-theoretic loss function
ORPO	2024	arXiv	Monolithic SFT + preference; eliminates separate SFT stage
SimPO	2024	NeurIPS	Reference-free; uses average log-prob as implicit reward; +6.4 AlpacaEval
CPO	2024	arXiv	Contrastive regularizer via KL to preferred data distribution
RainbowPO	2025	ICLR	Unified ablation framework across all DPO variants
RE-PO	2025	arXiv	Robust enhancement applicable to DPO/IPO/SimPO/CPO

Evolution of Preference Optimization Methods

ORPO: Monolithic Preference Optimization

ORPO [7] eliminates the separate SFT stage by combining the supervised language modeling objective with an odds-ratio preference penalty:

$$\mathcal{L}_{\text{ORPO}} = \mathcal{L}_{\text{SFT}} + \lambda \cdot \log\sigma\left(\log \frac{\text{odds}_\theta(y_w|x)}{\text{odds}_\theta(y_l|x)}\right)$$

where $\text{odds}_\theta(y|x) = \frac{p_\theta(y|x)}{1-p_\theta(y|x)}$. This halves total training compute and eliminates pipeline complexity. Empirically, ORPO matches or exceeds sequential SFT→DPO on multiple benchmarks.

KTO: Prospect-Theoretic Alignment

KTO [8] addresses a critical data bottleneck: paired preference data (y_w, y_l) is expensive to collect, but binary signals (y is "good" or "bad") are abundant in production systems (thumbs up/down, implicit engagement metrics). KTO derives a loss from Kahneman and Tversky's prospect theory, weighting losses more heavily than gains—reflecting the empirical observation that humans are more sensitive to bad outputs than good ones.

SimPO: Reference-Free Preference Optimization

SimPO [9] removes the reference model entirely by using the average log-probability of the generated sequence as an implicit reward:

$$r_{\text{SimPO}}(x, y) = \frac{1}{|y|}\sum_{t=1}^{|y|} \log \pi_\theta(y_t|x, y_{<t}) - \gamma$$

where γ is a target reward margin. This reduces memory from $2\times$ to $1\times$ the base model and eliminates the need to maintain a frozen reference policy. SimPO outperformed DPO by 6.4 points on AlpacaEval 2 and 3.2 points on Arena-Hard.

II. THE REASONING REVOLUTION: GRPO, RLVR, AND DAPO

The most significant methodological advance since DPO emerged in early 2025 with DeepSeek-R1 [4], which demonstrated that reasoning capabilities can emerge from pure reinforcement learning without supervised reasoning demonstrations. This challenged the prevailing assumption that complex behaviors require explicit demonstration.

GRPO: Group Relative Policy Optimization

GRPO simplifies PPO by eliminating the critic (value) network entirely. For each prompt x, the model generates a group of G responses $\{y_1, \ldots, y_G\}$, each receiving a scalar reward r_i. The advantage is estimated purely from group statistics:

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_1, \ldots, r_G\})}{\text{std}(\{r_1, \ldots, r_G\})}$$

The policy gradient uses clipped importance sampling identical to PPO:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}\left[\min\left(\rho_i \hat{A}_i,\ \text{clip}(\rho_i, 1-\varepsilon, 1+\varepsilon)\hat{A}_i\right)\right] + \beta\, \text{KL}(\pi_\theta \,\|\, \pi_{\text{ref}})$$

where $\rho_i = \frac{\pi_\theta(y_i|x)}{\pi_{\text{old}}(y_i|x)}$ is the importance ratio. By removing the critic network, GRPO eliminates both its memory cost and the well-documented instability of value function estimation in language model RL.

SFT is bounded by the demonstration ceiling: it can only reproduce patterns present in training data. GRPO enables exploration beyond the data distribution. DeepSeek-R1's Pass@1 on AIME 2024 improved from 15.6% to 71.0%—the model discovered reasoning strategies not present in any training example. This distinction—imitation vs. discovery—is the fundamental reason RL methods remain essential despite the simplicity of supervised alternatives.

RLVR: Reinforcement Learning with Verifiable Rewards

RLVR represents a 2025 paradigm shift in reward signal design. Instead of human preference labels or learned reward models, RLVR uses programmatic verification: mathematical proofs can be checked against known answers, code can be executed against test suites, logical statements can be formally verified, and structured outputs can be validated against schemas. This produces a perfect, noise-free binary reward signal at zero annotation cost.

The key theoretical advantage is that verifiable rewards eliminate both the reward model's approximation error and the annotation noise inherent in human preferences. The practical advantage is unlimited scaling: reward computation is embarrassingly parallel and has zero marginal cost.

DAPO: Decoupled Clip and Dynamic Sampling

ByteDance's DAPO [13] improves GRPO through four algorithmic refinements:

1. **Decoupled clipping**: Separate ε values for the upper and lower importance ratio bounds, allowing finer control over exploration vs. exploitation.

2. **Dynamic sampling**: Filters out prompts where all G responses receive identical rewards (zero gradient signal), improving sample efficiency.
3. **Token-level KL penalty**: Applies KL regularization per-token rather than per-sequence, preventing distribution collapse in long chain-of-thought generations.
4. **Overlong reward shaping**: Penalizes responses exceeding the context window to prevent truncation artifacts.

DAPO outperformed DeepSeek-R1-Zero on AIME 2024 (50 vs. 47 points) using 50% fewer training steps, demonstrating that the GRPO framework has significant room for algorithmic improvement.

III. PARAMETER-EFFICIENT FINE-TUNING (PEFT) METHODS

Full fine-tuning of a model with d-dimensional weight matrices requires storing full gradients and optimizer states, resulting in memory consumption of approximately $16d$ bytes per parameter (weights + gradients + Adam first/second moments in mixed precision). For an 8B-parameter model, this exceeds 80GB. Parameter-efficient methods reduce trainable parameters by 97–99.9% while retaining 93–99% of full fine-tuning quality.

LoRA: Low-Rank Adaptation

LoRA [1] is the foundational PEFT method. It freezes the pretrained weight matrix $W_0 \in \mathbb{R}^{d\times k}$ and injects a trainable low-rank decomposition:

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and $r \ll \min(d, k)$. Matrix A is initialized with random Gaussian entries; B is initialized to zero so that $\Delta W = BA = 0$ at the start of training, ensuring the model begins from the pretrained checkpoint.

Complexity Analysis

For a weight matrix $W_0 \in \mathbb{R}^{d\times d}$ with rank r:

- **Full fine-tuning:** d^2 trainable parameters, $\mathcal{O}(d^2)$ memory
- **LoRA:** $2dr$ trainable parameters, $\mathcal{O}(dr)$ memory
- **Reduction factor:** $\frac{d}{2r}$ (e.g., $d = 4096, r = 16$ yields 128 × reduction)

At inference, the adapter can be merged: $W = W_0 + BA$, adding *zero latency*. Multiple LoRA adapters can be hot-swapped on a single base model deployment.

Method	GPU VRAM	Trainable %	Quality Retention	Adapter Size
Full Fine-Tune	~80GB	100%	100% (baseline)	16GB
LoRA ($r = 16$)	~16GB	~0.1%	~97%	~50MB
QLoRA ($r = 16$)	~6GB	~0.1%	~95%	~50MB
DoRA ($r = 16$)	~18GB	~0.15%	~98%	~60MB
GaLore	~16GB	100%*	~99%	N/A

Resource Requirements by Parameter-Update Method (8B Model)

*GaLore updates all parameters but projects gradients to low-rank, achieving full-rank updates with low-rank memory.

QLoRA: Quantized LoRA

QLoRA [2] combines LoRA with 4-bit NormalFloat (NF4) quantization of the frozen base model:

$$h = \text{Dequant}(W_{\text{NF4}}) \cdot x + BA \cdot x$$

NF4 quantization is information-theoretically optimal for normally distributed weights, preserving more information per bit than uniform or symmetric quantization schemes. The LoRA adapters remain in BFloat16 precision, maintaining gradient fidelity. QLoRA introduced *double quantization*—quantizing the quantization constants themselves—and *paged optimizers* that offload optimizer states to CPU RAM during GPU memory pressure.

DoRA: Weight-Decomposed Low-Rank Adaptation

DoRA [5] observes that full fine-tuning and LoRA exhibit fundamentally different weight update patterns. Full fine-tuning tends to make large directional changes with small magnitude changes, while LoRA conflates both. DoRA decomposes the update:

$$W' = m \cdot \frac{V + BA}{\| V + BA \|_c}$$

where $W_0 = m \cdot \frac{V}{\|V\|_c}$ (magnitude–direction decomposition), $m \in \mathbb{R}^{1\times k}$ is a learnable magnitude vector, and LoRA is applied only to the directional component V. The column-wise normalization $\|\cdot\|_c$ ensures the directional component has unit norm per column. DoRA consistently outperforms LoRA by 1–3% across commonsense reasoning, visual instruction tuning, and mathematical reasoning benchmarks.

GaLore: Gradient Low-Rank Projection

GaLore [6] takes a fundamentally different approach: rather than constraining the *weight update* to be low-rank, it projects *gradients* into a low-rank subspace:

$$\tilde{G}_t = P_t^\top G_t, \quad \theta_{t+1} = \theta_t - \eta \cdot P_t \cdot \text{Adam}(\tilde{G}_t)$$

where $P_t \in \mathbb{R}^{d \times r}$ is a projection matrix updated periodically via SVD of the full gradient. This enables full-parameter learning—all weights receive updates—with LoRA-level memory consumption. GaLore demonstrated that a 7B model can be pretrained on a single 24GB consumer GPU without model parallelism, CPU offloading, or checkpoint sharding.

Spectrum: SNR-Based Layer Selection

Spectrum (2024) applies Signal-to-Noise Ratio analysis from Random Matrix Theory to identify layers with the highest information density. It computes the SNR of each layer's weight matrix by analyzing its singular value spectrum relative to the Marchenko-Pastur distribution (the theoretical spectrum of random matrices). Layers whose spectrum significantly deviates from the random baseline carry more learned information and benefit most from fine-tuning. Spectrum freezes the bottom 50–75% of layers by SNR, achieving 23% memory reduction and 37% faster training with no statistically significant quality loss.

NEFTune: Noisy Embedding Fine-Tuning

NEFTune [10] adds uniform random noise to embedding vectors during SFT:

$$\tilde{e} = e + \frac{\alpha}{\sqrt{Ld}} \cdot \mathcal{U}(-1,1)^d$$

where L is sequence length, d is embedding dimension, and α controls noise magnitude. Despite its simplicity, NEFTune boosted LLaMA-2-7B on AlpacaEval from 29.79% to 64.69%. The mechanism is hypothesized to act as a regularizer preventing overfitting to specific token sequences, encouraging the model to learn more generalizable instruction-following patterns.

Model Merging: Composition Without Gradient Computation

Model merging composes capabilities from multiple fine-tuned models without any gradient computation or training data. The key insight is that fine-tuned models sharing a common pretrained ancestor occupy nearby regions in weight space, and interpolation in this space often produces models that inherit capabilities from both parents. As of 2025, merged models represent 34% of the top 100 entries on the Open LLM Leaderboard.

Formal Framework

Given a base model θ_{base} and K fine-tuned variants$\{\theta_k\}_{k=1}^K$, define *task vectors* $\tau_k = \theta_k - \theta_{\text{base}}$. Merging operates on these task vectors.

Method	Description
Linear / SLERP	Spherical Linear Interpolation: $\theta_{\text{merged}} = \frac{\sin((1-t)\Omega)}{\sin\Omega}\theta_1 + \frac{\sin(t\Omega)}{\sin\Omega}\theta_2$ where $\Omega = \arccos(\hat{\theta}_1 \cdot \hat{\theta}_2)$. Maintains constant angular velocity in weight space.
TIES-Merging [12]	Three steps: (1) *Trim*: zero out small-magnitude changes; (2) *Elect sign*: resolve sign conflicts via majority vote; (3) *Merge*: average only sign-consistent parameters.
DARE [11]	Drop And REscale: randomly reset fraction p of fine-tuned deltas to zero, rescale remaining by $1/(1-p)$. Effective even at $p = 0.9$–0.99, suggesting extreme redundancy in task vectors.
Task Arithmetic	$\theta_{\text{merged}} = \theta_{\text{base}} + \sum_k \lambda_k \tau_k$. Addition composes capabilities; subtraction negates them (e.g., removing toxic behaviors).
Model Soup	Average weights across multiple training checkpoints or hyperparameter configurations. Improves robustness via implicit ensembling.

Model Merging Methods: Formal Descriptions

MergeKit

MergeKit (Arcee AI) is the standard open-source toolkit for model merging, supporting all methods above with lazy tensor loading for memory efficiency. It runs on CPU or with as little as 8GB VRAM. A typical merge completes in minutes—compared to hours or days for fine-tuning.

Model merging enables capability composition at zero training cost. A domain-specialized model can be merged with a code-generation model and a reasoning-tuned model to produce a combined specialist. This is particularly powerful when combined with fine-tuning: train individual experts on different data slices, then merge them. DARE's finding that 90–99% of task vector parameters can be dropped suggests that fine-tuned knowledge is encoded in a very sparse subset of weight changes.

Knowledge Distillation

Knowledge distillation transfers capabilities from a large "teacher" model T to a smaller "student" model S, enabling deployment on resource-constrained hardware while preserving a significant fraction of the teacher's performance.

Classical Distillation

The student is trained to match the teacher's softened output distribution:

$$\mathcal{L}_{\text{KD}} = \alpha \cdot \mathcal{L}_{\text{CE}}(y, \hat{y}_S) + (1-\alpha) \cdot T^2 \cdot \text{KL}\left(\text{softmax}\left(\frac{z_T}{\tau}\right) \parallel \text{softmax}\left(\frac{z_S}{\tau}\right)\right)$$

where z_T, z_S are teacher and student logits, τ is the temperature (higher values produce softer distributions that reveal more inter-class structure), and α balances the hard-label cross-entropy loss with the soft-label KL divergence. The T^2 scaling factor compensates for the gradient magnitude reduction caused by temperature scaling.

Reasoning Distillation: The DeepSeek-R1 Approach

DeepSeek demonstrated that reasoning capabilities trained via GRPO in a 671B-parameter model can be distilled into models ranging from 1.5B to 70B parameters. The distillation data consists of the teacher's *reasoning traces*—the full chain-of-thought, including intermediate steps, backtracking, and self-correction. The student learns not just the final answer but the *process* of arriving at it. Distilled models retained 85–95% of the teacher's reasoning performance at 10–100× fewer parameters.

Cross-Tokenizer Distillation

A practical barrier to distillation is that teacher and student models often use incompatible tokenizers (different vocabularies, different subword segmentation algorithms). Cross-tokenizer distillation (NeurIPS 2025) addresses this by projecting both models' token distributions into a shared semantic space before computing the KL divergence. This enables distillation across model families (e.g., Llama → Mistral, GPT → Qwen).

Comparative Knowledge Distillation (CKD)

CKD (2024) trains the student to mimic the teacher's *relative preferences* between response pairs rather than absolute probabilities. From N teacher inferences, CKD constructs up to $\binom{N}{2}$ pairwise comparisons, quadratically amplifying the training signal without additional teacher API calls. This approach is particularly effective when teacher access is expensive (e.g., proprietary API models).

Mixture of Experts (MoE) Fine-Tuning

Mixture of Experts models (Mixtral, DeepSeek-V3, Grok) route each token to a subset of k out of E expert sub-networks via a learned gating function $g(x) = \text{TopK}(\text{softmax}(W_g \cdot x))$. Fine-tuning MoE models presents unique challenges: expert specialization must be preserved, router distributions must remain balanced, and the sparse activation pattern complicates gradient estimation.

LoRA-Based Expert Injection

MoELoRA, MixLoRA, and LoRAMoE inject LoRA adapters as lightweight experts into pretrained dense or MoE models. TT-LoRA MoE combines tensor-train decomposition with sparse MoE routing for extreme parameter efficiency, matching full fine-tuning quality while updating $<1\%$ of parameters.

Branch-Train-MiX (BTX)

BTX parallelizes expert training: (1) initialize E copies of a seed model, (2) independently fine-tune each copy on a different data domain (embarrassingly parallel), (3) extract FFN layers as experts and combine with shared attention layers, (4) train the router with a brief MoE fine-tuning phase. This amortizes the cost of multi-domain expertise across parallel training runs.

Factor	Guidance
Batch size	Use smaller batches than dense models; large batches dilute expert gradients
Learning rate	Higher than dense equivalents; sparse activation means fewer gradient updates per parameter
Router entropy	Monitor continuously; low entropy indicates expert collapse (all tokens routed to same experts)
Load balancing	Apply auxiliary loss to prevent expert starvation ($\mathcal{L}_{\text{aux}} = E \cdot \sum_{e=1}^{E} f_e \cdot p_e$)

MoE-Specific Fine-Tuning Considerations

Multimodal Fine-Tuning

Vision-Language Model (VLM) fine-tuning has become mainstream, with open-source models (Qwen2.5-VL-72B, InternVL3-78B, LLaVA-OneVision) approaching proprietary model performance on standard benchmarks.

Architectural Patterns

Three paradigms dominate:

1. **Single-stage**: Joint training across all modalities with a unified next-token prediction objective over interleaved image and text tokens.
2. **Two-stage**: (a) Alignment pretraining freezes the LLM and trains only the vision encoder projection layer on large-scale image–caption pairs; (b) instruction tuning unfreezes the LLM and trains on multimodal instruction-following data.

3. **Direct adaptation**: Apply LoRA/QLoRA to a pretrained VLM on downstream tasks, updating only adapter parameters while freezing both the vision encoder and the language model.

Training Challenges

VIRAL regularization (2024) addresses a common failure mode: during fine-tuning, VLMs discard fine-grained visual attributes (color, texture, spatial relationships) in favor of high-level semantic features. VIRAL applies a regularization term that preserves attention to visual details. Catastrophic forgetting of the language model's text-only capabilities during multimodal training is mitigated through data mixing (interleaving text-only and multimodal training examples).

Synthetic Data and Data Quality

Data quality has emerged as the dominant factor in fine-tuning outcomes, consistently outweighing data quantity, model size, and training compute across empirical studies.

Synthetic Data Generation Pipelines

Modern fine-tuning pipelines increasingly rely on synthetic data generated by stronger teacher models:

- **Self-Instruct** [16]: Bootstraps from a small seed set of instructions, using the model itself to generate new instructions, inputs, and outputs, filtered by heuristic quality checks.
- **Evol-Instruct**: Iteratively evolves simple instructions into complex, multi-step variants by applying transformation operators (deepen, widen, add constraints, increase reasoning steps).
- **Distilabel**: Argilla's framework for constructing multi-step synthetic data generation pipelines with LLM-as-judge quality filtering. Supports structured output generation, preference pair construction, and contamination detection.

Quality Filtering

Empirical evidence strongly supports aggressive quality filtering. Key approaches include:

- **LLM-as-judge**: Use a stronger model (e.g., GPT-4, Claude) to score and filter generated examples.
- **Decontamination**: Remove examples that overlap with evaluation benchmarks to prevent artificial benchmark inflation.
- **Diversity maximization**: Ensure coverage of the target task distribution via embedding-based clustering and stratified sampling.
- **Difficulty calibration**: Include a range of difficulty levels; excessively easy examples provide little learning signal.

IV. THE MODERN FRAMEWORK ECOSYSTEM

The fine-tuning framework landscape has expanded dramatically beyond the foundational PEFT library. Table 8 compares the major options as of early 2026.

	Axolotl	**Unsloth**	**LLaMA-Factory**	**TRL**	**TorchTune**
Type	CLI framework	Optimized kernel runtime	CLI + Web UI	Python library	PyTorch-native
Speed	1×	2–5×	1–2×	1×	1×
VRAM	Standard	70–80% less	Standard	Standard	Standard
Stars	8K+	50K+	40K+	15K+	5K+
SFT					
DPO					
GRPO	Partial				Partial
Multimodal	Partial				

Fine-Tuning Framework Comparison (2026)

Additional frameworks include **ms-swift** (ModelScope's framework supporting 600+ LLMs and 300+ multimodal models with SFT/DPO/GRPO) and **PEFT** (HuggingFace's foundational library implementing LoRA, QLoRA, IA3, prefix tuning, and other adapter methods).

Unsloth achieves its 2–5× speedup through custom Triton kernels for attention, cross-entropy, and RoPE computation—not through algorithmic approximation. This means Unsloth produces *bit-identical* results to standard training, making the speedup a pure engineering gain with no quality tradeoff. For compute-constrained practitioners, Unsloth is the highest-impact single change to the training pipeline.

Quantization for Deployment

Quantization bridges fine-tuning and deployment, compressing models for inference on constrained hardware. The fine-tune-to-deploy pipeline typically follows: train with LoRA/QLoRA → merge adapters → quantize → serve.

Method	Bits	Quality Retention	Key Property
FP16/BF16	16	100% (baseline)	Standard training/inference precision
GPTQ	4	95–98%	Post-training; calibration-based
AWQ	4	96–99%	Activation-aware channel scaling
GGUF	2–8	Varies	llama.cpp format; portable; mixed quantization
AQLM	2–3	90–95%	Multi-codebook additive quantization
EfficientQAT	4	97–99%	Quantization-aware training (ACL 2025)

Quantization Methods for LLM Deployment

Practical Pipeline: End-to-End Domain Specialization

To ground these techniques, we present a reference pipeline for building a domain-specialized agent from a pretrained foundation model.

Architecture

Step	Phase	Objective	Method	Data
1	Base model selection	–	–	Llama-3-8B / Qwen-2.5-7B
2	Domain knowledge	SFT	QLoRA ($r = 16$)	1,000–5,000 expert examples
3	Behavior alignment	ORPO or DPO	LoRA ($r = 8$)	200–1,000 preference pairs
4	(Optional) Reasoning	GRPO	LoRA ($r = 16$)	Verifiable task prompts
5	Adapter merging	–	–	merge-lora
6	(Optional) Model merge	TIES/DARE	–	Complementary experts
7	Quantization	–	–	GGUF Q4_K_M
8	Deployment	–	–	Ollama / vLLM / TGI

Reference Fine-Tuning Pipeline

Configuration Examples

```
base_model: meta-llama/Meta-Llama-3-8B
model_type: LlamaForCausalLM
load_in_4bit: true # QLoRA: NF4 quantization
adapter: lora
lora_r: 16
lora_alpha: 32
lora_dropout: 0.05
lora_target_modules: [q_proj, v_proj, k_proj, o_proj]
datasets:
  - path: domain_sft.jsonl
    type: alpaca
sequence_len: 4096
micro_batch_size: 2
gradient_accumulation_steps: 8 # effective batch = 16
num_epochs: 3
learning_rate: 2e-4
warmup_ratio: 0.03
lr_scheduler: cosine
neftune_noise_alpha: 5 # NEFTune regularization
output_dir: ./sft-model
```

```
base_model: meta-llama/Meta-Llama-3-8B
adapter: lora
lora_r: 16
lora_alpha: 32
datasets:
  - path: domain_orpo.jsonl
    type: orpo
orpo_alpha: 0.1 # odds-ratio weight
output_dir: ./orpo-model
```

Deployment

```
# Train (single command)
axolotl train sft_config.yml

# Merge LoRA into base model
axolotl merge-lora sft_config.yml

# Convert to GGUF for portable deployment
python convert_hf_to_gguf.py ./merged-model --outtype q4_k_m

# Serve via Ollama
ollama create domain-expert -f Modelfile
ollama run domain-expert
```

Cost Analysis

Configuration	Hardware	Time	Cost
SFT + QLoRA (8B)	RTX 4090 (24GB)	2–4 hrs	$0 (own hardware)
SFT + QLoRA (8B)	Cloud A100	1–2 hrs	$2–5
SFT + DPO (8B)	Cloud A100	3–6 hrs	$5–20
SFT + GRPO (8B)	Cloud A100	12–48 hrs	$50–200
SFT + DPO (70B)	4× A100 80GB	24–72 hrs	$200–800
Full fine-tune (8B)	A100 80GB	24–48 hrs	$50–200
Enterprise RLHF	H100 cluster	Weeks	$10,000+

Estimated Fine-Tuning Costs by Configuration

Unified Taxonomy

Table 12 presents the complete taxonomy of post-training techniques organized by functional category.

Category	Technique	Function	Key Property
Knowledge	SFT	Domain knowledge injection	Supervised; input→output pairs
	DPO	Behavioral alignment	Pairwise preferences; no RM
	RLHF	Behavioral alignment	Reward model + PPO
	ORPO	Joint SFT + alignment	Monolithic single-stage
	KTO	Behavioral alignment	Binary (unpaired) feedback
	SimPO	Behavioral alignment	Reference-free; length-normalized
	GRPO	Emergent reasoning	Group-relative advantage; no critic
	RLVR	Verified reasoning	Programmatic reward signals
	DAPO	Improved GRPO	Decoupled clipping + dynamic sampling
	LoRA	Low-rank weight adapters	Frozen base + trainable BA
	QLoRA	LoRA + 4-bit base	NF4 quantization
	DoRA	Magnitude/direction split	Column-wise decomposition
	GaLore	Gradient projection	Full-rank update, low-rank memory
	Spectrum	Layer selection by SNR	Random Matrix Theory
	Model Merging	Weight-space composition	Zero gradient computation
	Distillation	Model compression	Teacher→student transfer
	Quantization	Bit-width reduction	Deployment optimization

Unified Taxonomy of Post-Training Techniques

Open Problems and Future Directions

1. **Unified training objectives**: ORPO demonstrated that SFT and preference alignment can be collapsed into a single stage. Extending this unification to include reasoning enhancement (GRPO) would yield a single-stage pipeline from base model to fully aligned reasoner.
2. **RLVR beyond math and code**: Verifiable rewards currently require tasks with checkable answers. Extending programmatic verification to open-ended domains—summarization quality, factual accuracy, stylistic adherence—is an open challenge. Hybrid approaches combining verifiable sub-rewards with learned critics are a promising direction.
3. **Theoretical foundations of model merging**: DARE's finding that 90–99% of task vector parameters can be dropped without quality loss lacks a satisfactory theoretical explanation. Understanding the geometry of fine-tuned weight spaces could yield principled merging algorithms with provable guarantees.
4. **Continual and incremental fine-tuning**: Current methods assume a fixed training dataset. Techniques for incremental adaptation—adding new capabilities without catastrophic forgetting of previous fine-tuning—remain underdeveloped for LLMs at scale.
5. **On-device fine-tuning**: QLoRA and Spectrum point toward fine-tuning on edge devices. Achieving practical on-device adaptation for mobile and embedded deployment would fundamentally change the economics of model personalization.
6. **Efficient MoE adaptation**: As sparse MoE architectures become dominant in frontier models, developing parameter-efficient methods that respect expert specialization boundaries—rather than applying LoRA uniformly—is an active research area.
7. **Automated pipeline selection**: Given the combinatorial space of technique combinations (Table 2), automated methods for selecting optimal pipeline configurations based on data characteristics, compute budget, and task requirements would significantly reduce practitioner burden.

V. CONCLUSION

The LLM post-training landscape has evolved from a linear two-stage pipeline (SFT → RLHF) into a rich combinatorial space of composable techniques spanning knowledge injection, preference alignment, reasoning enhancement, parameter-efficient adaptation, model composition, and deployment optimization. The two-axis decomposition—training objective × parameter-update strategy—provides a principled framework for navigating this complexity.

Three developments stand out as inflection points. First, DPO's reparameterization (2023) collapsed the three-model RLHF pipeline into a single supervised objective, reducing the barrier to preference alignment by an order of magnitude in both compute and implementation complexity. Second, GRPO and RLVR (2025) demonstrated that models can develop reasoning capabilities beyond their training data through guided exploration with verifiable rewards—a capability fundamentally inaccessible to supervised methods. Third, model merging introduced capability composition without gradient computation, opening a new axis of model development orthogonal to training.

Combined with dramatic efficiency improvements—QLoRA's 6GB fine-tuning, Unsloth's 2–5× speedup, NEFTune's zero-cost quality boost—these advances have democratized post-training. A practitioner with a single consumer GPU and a curated dataset of 1,000 examples can now produce a domain-specialized model that exceeds general-purpose commercial models on targeted tasks. The recommended pipeline for most applications is: curate high-quality data → SFT with QLoRA + NEFTune → preference alignment with ORPO or KTO → optional model merging → quantize to GGUF → deploy.

The central tension in post-training is between imitation (SFT: reproduce demonstrations) and discovery (GRPO/RLVR: find solutions through exploration). The most capable models combine both: SFT provides the foundation of knowledge and instruction-following, while RL-based methods push capabilities beyond the ceiling of available demonstrations. Understanding when each paradigm is appropriate—and how to compose them—is the key competency for practitioners navigating the modern fine-tuning landscape.

References

1. [ref:lora] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). "LoRA: Low-Rank Adaptation of Large Language Models." *ICLR 2022*. https://arxiv.org/abs/2106.09685
2. [ref:qlora] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). "QLoRA: Efficient Finetuning of Quantized Language Models." *NeurIPS 2023*. https://arxiv.org/abs/2305.14314
3. [ref:dpo] Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. (2023). "Direct Preference Optimization: Your Language Model is Secretly a Reward Model." *NeurIPS 2023*. https://arxiv.org/abs/2305.18290
4. [ref:deepseek] DeepSeek-AI. (2025). "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning." *Nature*. https://arxiv.org/abs/2501.12948
5. [ref:dora] Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. (2024). "DoRA: Weight-Decomposed Low-Rank Adaptation." *ICLR 2025*. https://arxiv.org/abs/2402.09353
6. [ref:galore] Zhao, J., Zhang, Z., Chen, B., Wang, Z., Anandkumar, A., and Tian, Y. (2024). "GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection." *ICML 2024*. https://arxiv.org/abs/2403.03507
7. [ref:orpo] Hong, J., Lee, N., and Thorne, J. (2024). "ORPO: Monolithic Preference Optimization without Reference Model." https://arxiv.org/abs/2403.07691
8. [ref:kto] Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. (2024). "KTO: Model Alignment as Prospect Theoretic Optimization." https://arxiv.org/abs/2402.01306
9. [ref:simpo] Meng, Y., Xia, M., and Chen, D. (2024). "SimPO: Simple Preference Optimization with a Reference-Free Reward." *NeurIPS 2024*. https://arxiv.org/abs/2405.14734

10. [ref:neftune] Jain, N., Chiang, P.-y., Wen, Y., Kirchenbauer, J., Chu, H.-M., Somepalli, G., Bartoldson, B. R., Kailkhura, B., Schwarzschild, A., Saha, A., Goldblum, M., Geiping, J., and Goldstein, T. (2024). "NEFTune: Noisy Embeddings Improve Instruction Finetuning." *ICLR 2024*. https://arxiv.org/abs/2310.05914
11. [ref:dare] Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. (2024). "Language Model Surgery: Efficient Knowledge Unlearning and Editing via Weight Disentanglement." https://arxiv.org/abs/2311.03099
12. [ref:ties] Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. (2023). "TIES-Merging: Resolving Interference When Merging Models." *NeurIPS 2023*. https://arxiv.org/abs/2306.01708
13. [ref:dapo] Yu, Q., et al. (2025). "DAPO: An Open-Source LLM Reinforcement Learning System." https://arxiv.org/abs/2503.14476
14. [ref:instructgpt] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). "Training Language Models to Follow Instructions with Human Feedback." *NeurIPS 2022*. https://arxiv.org/abs/2203.02155
15. [ref:ppo] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). "Proximal Policy Optimization Algorithms." https://arxiv.org/abs/1707.06347
16. [ref:selfinstruct] Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. (2023). "Self-Instruct: Aligning Language Models with Self-Generated Instructions." *ACL 2023*. https://arxiv.org/abs/2212.10560
17. [ref:lima] Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., Yu, L., Zhang, S., Ghosh, G., Lewis, M., Zettlemoyer, L., and Levy, O. (2023). "LIMA: Less Is More for Alignment." *NeurIPS 2023*. https://arxiv.org/abs/2305.11206
18. HuggingFace PEFT Library. https://github.com/huggingface/peft
19. Axolotl Fine-Tuning Framework. https://github.com/axolotl-ai-cloud/axolotl
20. Unsloth. https://github.com/unslothai/unsloth
21. LLaMA-Factory. https://github.com/hiyouga/LLaMA-Factory
22. TRL: Transformer Reinforcement Learning. https://github.com/huggingface/trl
23. MergeKit. https://github.com/arcee-ai/mergekit
24. Ollama. https://github.com/ollama/ollama
25. TorchTune. https://github.com/pytorch/torchtune

Acknowledgments

The author extends sincere gratitude to family, friends, and colleagues for their continuous encouragement and support. Their willingness to assume some of the routine responsibilities provided the time and space essential for this research

Syncaissa Systems Inc. syncaissa@outlook.com

.