LLM Architecture, Fine-Tuning, Applications, Challenges and Future Directions

Bhavya Kumar[1] Student, B.Tech (AI&DS)
Ansh Raj Mittal[2] Student, B.Tech (AI&DS)
Satvik[3] Student, B.Tech (AI&DS)
Dr. Ritesh Kumar[4] Assistant Professor

Department of Artificial Intelligence & Data Science
Dr. Akhilesh Das Gupta Institute of Professional Studies
New Delhi, India

Abstract—This paper presents a case study of the architecture underlying modern Large Language Models (LLMs) and surveys the principal techniques used to fine-tune them for downstream and domain-specific use. Beginning with the decoder-only Transformer—token and positional embeddings, masked multi-head self-attention, position-wise feed-forward sublayers, residual connections and Layer Normalization—the paper classifies fine-tuning approaches into full fine-tuning, parameter-efficient fine-tuning (PEFT) methods (LoRA, QLoRA, adapters, prompt/prefix tuning), and alignment-oriented methods (instruction tuning, RLHF, RLAIF and DPO). The methods are compared on trainable-parameter footprint, memory requirements and typical deployment scenarios, and an illustrative case of LoRA-based domain adaptation for medical text summarisation is analysed. Practical applications in healthcare, legal analysis, customer support, software engineering, education, finance and scientific research are reviewed, followed by a critical discussion of challenges including catastrophic forgetting, hallucination, bias, computational cost, privacy and evaluation difficulty. The study is literature-based and does not report new experiments; it concludes with future directions in memory-efficient optimisation, higher-capacity low-rank methods, federated fine-tuning, improved alignment and multimodal adaptation.

Keywords—Large Language Models, Transformer Networks, Fine-Tuning, Parameter-Efficient Fine-Tuning, LoRA, QLoRA, Instruction Tuning, RLHF, Self-Attention, Domain Adaptation, Natural Language Processing.

I. Introduction

The past five years have witnessed an unprecedented shift in artificial intelligence research and practice, driven largely by Large Language Models (LLMs). Beginning with the Transformer architecture [1] and scaling through models such as GPT-3 [2] and LLaMA [3], LLMs have progressed from research prototypes to production systems embedded in search engines, coding assistants, customer service platforms and scientific research pipelines. Their defining characteristic is the ability to perform a wide range of language tasks—translation, summarisation, question answering, reasoning and code generation—using a single pre-trained model.

LLMs are pre-trained on massive, web-scale text corpora with self-supervised objectives, allowing them to acquire broad linguistic, factual and reasoning capabilities. However, a general-purpose pre-trained model is rarely optimal for a specific task, domain or organisational requirement out of the box. This gap between general pre-training and specialised deployment is bridged by fine-tuning: further training a pre-trained model on a smaller task-specific or domain-specific dataset so that its parameters are adapted to produce more accurate, relevant and controllable outputs.

As LLMs have grown from millions to hundreds of billions of parameters, fine-tuning the entire parameter set has become computationally expensive and memory-intensive. This has motivated Parameter-Efficient Fine-Tuning (PEFT) techniques such as LoRA, adapters and prompt/prefix tuning, as well as alignment-oriented approaches such as instruction tuning and Reinforcement Learning from Human Feedback (RLHF).

The study follows a descriptive, literature-based case-study methodology. Foundational works, PEFT surveys and method-defining papers were reviewed, and the findings were organised into architectural analysis, comparative technique review, case synthesis, challenge analysis and future-direction synthesis. The principal contributions of this paper are:

- A structured description of the decoder-based Transformer architecture underlying contemporary LLMs, with its attention and feed-forward formulation.
- A taxonomy of fine-tuning approaches—full, parameter-efficient and alignment-oriented—with a comparative table of trainable parameters, memory use and use cases.
- An illustrative LoRA-based domain-adaptation case with a worked trainable-parameter calculation.
- A critical review of applications, technical and ethical challenges, and emerging research directions in efficient and responsible fine-tuning.

II. Related Work

A. Parameter-Eflcient Fine-Tuning

The foundation of modern LLMs is the Transformer [1], which replaced recurrence with self-attention and enabled efficient parallel training on large corpora. BERT [4] and the GPT series [2] then demonstrated that large-scale pre-training followed by task-specific adaptation yields state-of-the-art results across NLP tasks. As models scaled into the billions of parameters, full fine-tuning became prohibitively expensive. Houlsby et al. [5] introduced adapter modules—small trainable layers inserted between frozen Transformer blocks. Lester et al. [6] proposed prompt tuning, which learns continuous soft prompts while keeping the base model frozen. Hu et al. [7] introduced Low-Rank Adaptation (LoRA), which injects trainable low-rank matrices into attention weights, sharply reducing trainable parameters while preserving performance comparable to full fine-tuning. Dettmers et al. [8] extended this with QLoRA, combining 4-bit quantisation of the base model with LoRA adapters to fine-tune 65-billion-parameter models on a single GPU. The LLM-Adapters framework [9] unified several adapter variants for LLMs.

Han et al. [10], in a comprehensive survey, categorise PEFT methods into addition-based (adapters, prefix/prompt tuning), reparameterisation-based (LoRA and its variants) and selective/sparse methods (BitFit, LISA). They note that even PEFT retains notable training memory overhead because of backpropagation, motivating zeroth-order approaches such as MeZO [16], which fine-tunes using forward passes only.

Later work refines LoRA itself. DoRA [11] decomposes pre-trained weights into magnitude and direction components and applies low-rank updates to the directional component; a separately named DoRA [12] instead allocates rank dynamically across layers. LoRA+ [13] shows that using different learning rates for the two LoRA matrices accelerates convergence. Quantisation-aware variants such as LoftQ [14] and QA-LoRA [15] integrate low-bit quantisation into the adapter initialisation or training process, narrowing the gap to full-precision fine-tuning. GaLore [17] reduces optimiser memory through low-rank gradient projection. Federated fine-tuning research further combines

TABLE I
SUMMARY OF KEY FINE-TUNING LITERATURE

Work	Year	Key Contribution
Houlsby et al. [5]	2019	Adapter modules inserted between frozen Transformer layers
Lester et al. [6]	2021	Prompt tuning with learned continuous soft prompts
Hu et al. [7]	2022	LoRA: trainable low-rank updates to frozen weights
Ouyang et al. [18]	2022	Instruction tuning with RLHF (Instruct-GPT)
Dettmers et al. [8]	2023	QLoRA: 4-bit base model with LoRA adapters
Han et al. [10]	2024	Comprehensive PEFT survey and tax-onomy
Liu et al. [11]	2024	DoRA: weight decomposition into mag-nitude and direction
Hayou et al. [13]	2024	LoRA+: separate learning rates for LoRA matrices
Malladi et al. [16]	2023	MeZO: fine-tuning with forward passes only

PEFT with federated learning so that LLMs can be adapted across distributed, privacy-sensitive data without centralising raw data.

B. Alignment-Oriented Fine-Tuning

Ouyang et al. [18] showed that instruction tuning combined with RLHF substantially improves the helpfulness, harmlessness and honesty of LLM outputs, forming the basis of InstructGPT and ChatGPT. Subsequent work explores Reinforcement Learning from AI Feedback (RLAIF) and Direct Preference Optimisation (DPO) [19] as computationally lighter alternatives to classical RLHF pipelines. Overall, the literature shows a trajectory from "how do we adapt this model" toward "how do we adapt it efficiently, safely and at scale."

III. LLM ARCHITECTURE

A. Decoder-Only Transformer

Most contemporary LLMs (GPT family, LLaMA, Mistral, Claude) are decoder-only Transformers. As shown in Fig. 1, the architecture consists of a token and positional embedding layer followed by a stack of □ identical decoder blocks. Each block contains a masked multi-head self-attention sublayer and a position-wise feed-forward sublayer, with residual connections and Layer Normalization around each. A final linear projection followed by softmax yields a probability distribution over the vocabulary for next-token prediction.

B. Self-Attention and Multi-Head Attention

Self-attention allows each token to attend to every earlier token in the sequence. Given an input □, learned linear projections produce queries □, keys □ and values □ , and attention is computed as scaled dot-product attention:

$$\text{Attention}(□, □, □) = \text{softmax}\left(\frac{□□^{□}}{\sqrt{□_{□}}}\right)□. \quad (1)$$

where $□_{□}$ is the key dimension; the $1/\sqrt{□_{□}}$ factor prevents large dot products from saturating the softmax. In decoder models a causal mask blocks attention to future positions. Multi-head attention runs *h* such operations in parallel and concatenates the results:

$$\text{MultiHead}(□, □, □) = \text{Concat}(h□□□_1, \ldots, h□□□_h)□^{□}, \quad (2)$$

allowing the model to capture different types of relationships simultaneously.

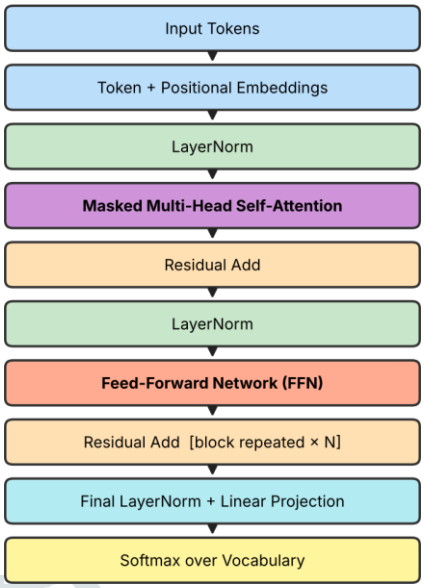


Fig. 1. Decoder-based LLM architecture—embeddings, repeated attention and feed-forward blocks with residual connections, and a softmax output over the vocabulary.

C. Residual Connections, Normalization and Feed-Forward Sublayer

Each sublayer is wrapped with a residual connection and Layer Normalization, which stabilises activations and gradient flow in deep stacks. The feed-forward sublayer applies two linear transformations with a non-linearity to each position independently, expanding and then restoring the hidden dimension. Stacking dozens to hundreds of such layers, with models ranging from a few billion to over a trillion parameters, gives LLMs their capacity for in-context learning, reasoning and generation.

D. Pre-Training Objective

Pre-training minimises the autoregressive next-token cross-entropy over a text corpus:

$$□ = -\sum \log □(□_{□} \mid □_1, \ldots, □_{□-1}). \quad (3)$$

This self-supervised objective requires no manual labels, which is what allows pre-training at web scale before any task-specific adaptation.

IV. FINE-TUNING TECHNIQUES

A. Full Fine-Tuning

Fig. 2 shows the taxonomy of approaches. Full fine-tuning updates every parameter of the pre-trained model by gradient descent on a downstream dataset. It typically achieves the highest task performance but requires storing gradients and optimiser states for the entire parameter set, making it costly for models beyond a few billion parameters.

B. Parameter-Eflcient Fine-Tuning (PEFT)

PEFT methods freeze most of the base model and train only a small number of additional or selected parameters:

- **LoRA:** injects trainable low-rank matrices □ and □ into weight matrices (typically the attention projections). The frozen weight $□_0$ is augmented as $□ = □_0 + Δ□$ with $Δ□ = □□$, where the

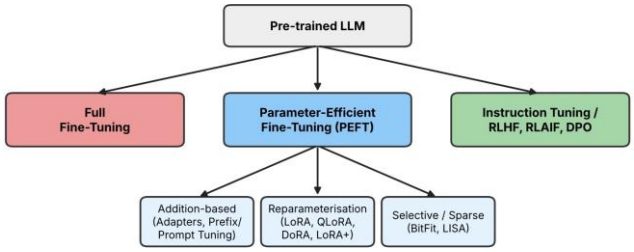


Fig. 2. Taxonomy of LLM fine-tuning approaches.

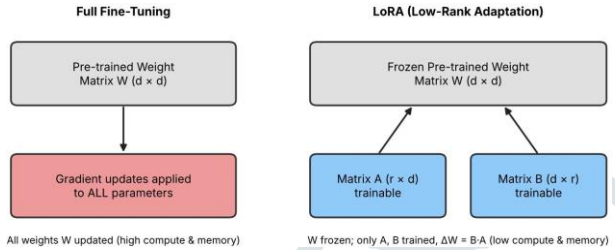


Fig. 3. Full fine-tuning versus LoRA-based fine-tuning.

rank □ is much smaller than the original dimension [7]. Only □ and □ are trained.

- **QLoRA:** quantises the frozen base model to 4 bits (NF4) and trains LoRA adapters in higher precision, enabling fine-tuning of very large models on a single GPU [8].
- **Adapters:** small bottleneck feed-forward modules inserted between Transformer layers; only these modules are trained [5].
- **Prompt/Prefix tuning:** learns a small set of continuous embeddings prepended to the input or to each layer's keys and values while the base model stays frozen [6].

For LoRA, the adapted forward pass for input □ is:

$$h = \square_0\square + \frac{\square}{\square}\square\square\square, \quad (4)$$

where □ is a scaling constant. Fig. 3 contrasts this with full fine-tuning.

C. Instruction Tuning and Preference-Based Alignment

Instruction tuning fine-tunes a model on curated instruction–response pairs. It is often followed by reinforcement learning against a reward model trained on human (RLHF) or AI (RLAIF) preference data to align outputs with desired behaviour [18]. DPO [19] optimises directly on preference pairs without training a separate reward model, reducing pipeline cost and complexity.

V. Comparative Analysis of Fine-Tuning Methods

Table II compares the principal techniques by trainable-parameter proportion, memory requirement and typical use case, based on the reviewed literature [7], [8], [10]. The figures are approximate and vary with model, rank and configuration.

VI. Illustrative Case: Domain Adaptation via LoRA

Consider adapting a general-purpose open-weight LLM (e.g., LLaMA) for medical text summarisation. Rather than full fine-tuning—which updates billions of parameters and needs substantial GPU memory—a LoRA adapter with rank □ = 8–16 can be trained on a domain corpus of clinical notes and reference summaries.

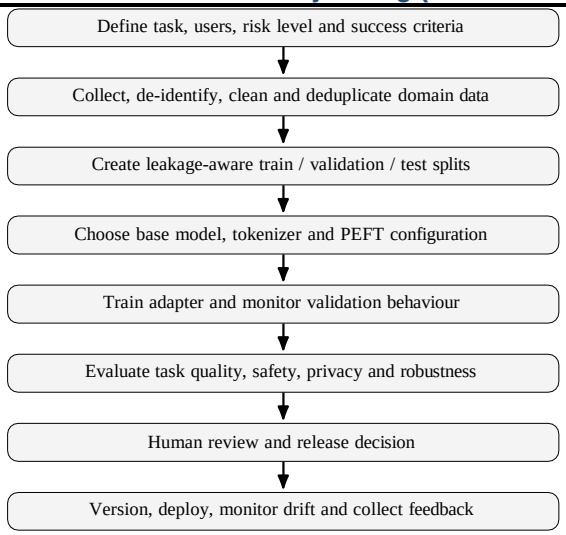


Fig. 4. End-to-end workflow for a controlled PEFT adaptation project.

Comparative studies of PEFT for medical summarisation report that LoRA and prompt tuning can approach full fine-tuning performance while training only a small fraction of parameters, with the low-rank constraint acting as a regulariser that reduces overfitting on limited domain data.

The saving can be quantified for a single square projection matrix of dimension □ = 4096. Full fine-tuning updates $\square^2$ = 16,777,216 parameters, whereas a rank-□ LoRA update trains only 2□□ parameters (Table III).

This exemplifies how PEFT makes domain-specific adaptation feasible even for organisations with limited computational resources.

A. A Practical Adaptation Workflow

A real deployment requires more than selecting a rank and starting an optimiser. The team first defines the intended task and the boundary of acceptable model behaviour. For clinical summarisation, for example, the target is to compress a note while retaining clinically meaningful facts, temporal relationships, uncertainty and negation. The dataset is then curated into input–target pairs, de-identified where necessary, checked for duplicates and partitioned by patient or source rather than by randomly splitting near-duplicate passages. This reduces the risk that nearly identical records appear in both training and evaluation partitions.

The base model and tokenizer are selected against context-length, licence, language and hardware constraints. A LoRA configuration specifies the modules to adapt, rank □, scaling factor □, dropout and optimiser settings. Training is monitored using held-out task loss and task-specific quality checks. The best checkpoint is not automatically the final deployment candidate: it must pass factuality, privacy, robustness and human-review gates. Finally, the adapter is versioned with its base-model identifier, dataset provenance, configuration and evaluation report. This workflow is a recommended synthesis of the literature, not a report of an experiment performed by the authors.

B. LoRA Configuration and Design Choices

The rank controls the dimensionality of the trainable update. A small rank reduces adapter size and may act as a useful constraint when the target dataset is limited; a larger rank increases representational capacity but also increases trainable parameters and may overfit. The scaling factor controls the magnitude of the low-rank

TABLE II
COMPARATIVE OVERVIEW of LLM FINE-TUNING METHODS

Method	Trainable Params	Memory Use	Base Weights	Typical Use Case
Full Fine-Tuning	~100%	Very High	Updated	Maximum performance, ample compute
Adapters	< 5%	Low–Moderate	Frozen	Multi-task deployment
Prompt/Prefix Tuning	< 1%	Low	Frozen	Lightweight task switching
LoRA	~0.1–1%	Low	Frozen	General-purpose PEFT
QLoRA	~0.1–1%	Very Low	Frozen + Quantised	Fine-tuning on limited hardware
Instruction Tuning + RLHF	Varies	High	Updated/Adapted	Alignment and safety

TABLE III
TRAINABLE PARAMETERS PER $d_{\text{in}} \times d_{\text{out}}$ WEIGHT MATRIX ($d_{\text{in}} = 4096$), COMPUTED ANALYTICALLY

LoRA rank r	Trainable params ($2rd_{\text{in}}$)	Share of d_{in}^2
4	32,768	0.195%
8	65,536	0.391%
16	131,072	0.781%

TABLE IV
ILLUSTRATIVE LoRA CONFIGURATION DECISIONS

Decision	Lower-cost choice	Trade-off to examine
Rank r	Smaller rank	May constrain adaptation capacity
Target modules	Attention projections only	May miss useful changes in feed-forward layers
Adapter dropout	Non-zero regularisation	Can reduce overfitting; excessive dropout may underfit
Quantisation	4-bit frozen base	Lower memory; numerical behaviour requires validation
Sequence length	Task-relevant context	Truncation may remove critical evidence
Learning rate	Conservative adapter rate	Too low underfits; too high destabilises updates

update relative to the frozen projection. Dropout can regularise the adapter path, while the choice of target modules determines which transformations can change. Applying LoRA to query and value projections is a common configuration, but adapting additional attention or feed-forward projections changes the parameter budget and should be justified by validation results.

Rank should therefore be treated as a tunable design variable rather than a universal constant. A sound comparison holds the base model, data split, training budget and evaluation protocol fixed while varying one configuration dimension at a time. Reporting only the best score without the rank, target modules, precision, sequence length and training budget makes results difficult to reproduce.

C. Data Preparation and Leakage Control

Domain adaptation quality is bounded by the quality and representativeness of the examples. A dataset should define the input unit, expected output format, inclusion criteria and exclusion criteria before annotation begins. For summarisation, examples should preserve the distinction between observed facts, reported history, uncertain hypotheses and absent information. Inconsistent annotation

TABLE V
DATASET QUALITY CHECKS BEFORE FINE-TUNING

Check	Purpose and practical evidence
Provenance	Record source, permission, collection date and permitted use
Privacy	Remove or transform identifiers; restrict access to raw records
Deduplication	Detect exact and near-duplicate inputs across partitions
Coverage	Inspect domain, language, subgroup and task coverage
Annotation consistency	Use a written rubric and adjudicate disagreements
Split integrity	Keep related entities in one partition where appropriate
Output validity	Check required fields, formatting and missing-value conventions

conventions can teach the model contradictory output styles even when the underlying facts are correct.

Partitioning is especially important for records that share a patient, customer, case, document template or source. A random row-level split can place highly similar examples in both training and test sets, inflating apparent generalisation. Grouped splits reduce this leakage. The test set should remain untouched during configuration selection; otherwise repeated inspection turns it into an implicit validation set. Dataset documentation should record provenance, licensing, sensitive fields, filtering rules, known skews and intended use.

D. Training and Release Decision Flow

A practical release process separates model optimisation from deployment approval. A decreasing training loss is evidence that the model is fitting the training objective; it is not proof that summaries are faithful, fair or safe. The validation set supports model selection, while the held-out test set estimates performance after choices are fixed. High-risk use cases additionally require expert review and a fallback path when the model is uncertain or the input falls outside the supported scope.

VII. EVALUATION FRAMEWORK FOR FINE-TUNED LLMS

Evaluation should reflect the intended task rather than relying on one aggregate score. For summarisation, lexical overlap metrics can indicate similarity to a reference but may reward fluent paraphrases that omit a critical fact or introduce an unsupported statement. Human assessment and targeted error analysis are therefore important complements. For instruction-following systems, evaluation should include whether the response follows constraints, refuses disallowed requests appropriately and communicates uncertainty.

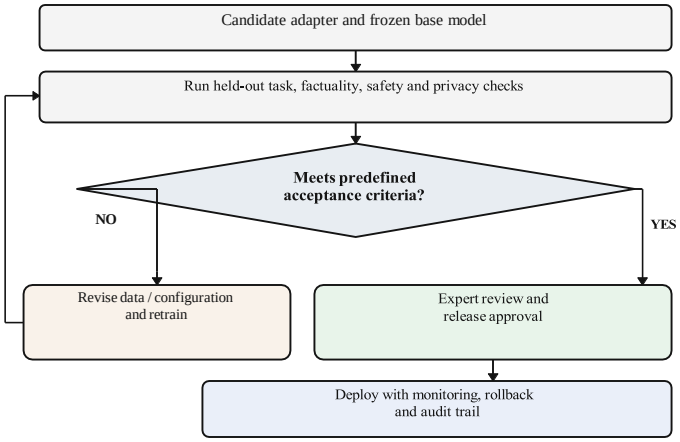


Fig. 5. Quality-gated release flow. Passing automated metrics alone does not replace expert review in high-impact settings.

TABLE VI
EVALUATION DIMENSIONS AND EXAMPLE EVIDENCE

Dimension	Example evidence to report
Task fidelity	Required facts retained; task rubric score
Factuality	Unsupported claims, contradictions and citation correctness
Completeness	Critical omissions and coverage of required fields
Robustness	Performance under paraphrase, noise and long inputs
Fairness	Error distribution across relevant groups or dialects
Safety	Refusal quality, privacy leakage and harmful output rate
Efficiency	Trainable parameters, peak memory, latency and throughput
Human utility	Expert preference, edit distance or acceptance rate

A. Task Quality and Factuality

A task-specific rubric can score coverage of required information, factual consistency, omission severity, readability and format compliance. In medical or legal contexts, an error taxonomy should distinguish a fabricated fact from a missing fact, a changed negation, an incorrect temporal relation and an unsupported inference. These errors have different consequences and should not be collapsed into a single average. Reference-based automatic metrics are useful for screening large test sets, but sampled outputs should be reviewed by qualified evaluators.

B. Robustness, Fairness and Safety

Robustness testing varies spelling, formatting, input length, document structure and irrelevant content while preserving the underlying task. Fairness analysis checks whether performance or error types differ across relevant user groups or language varieties, subject to lawful and ethical data access. Safety evaluation examines harmful instructions, sensitive-data reproduction, overconfident responses and failures to defer to human expertise. A model card or system card should document these limits and the conditions under which the model has not been evaluated.

C. Reproducibility and Reporting

A reproducible report should identify the base model and revision, tokenizer, dataset version, split method, adapter type, target modules, rank, scaling, quantisation format, optimiser, learning rate, batch construction, sequence length, number of updates, random seeds and hardware. It should also state whether hyperparameters were selected using the test set, how missing or malformed outputs were handled, and whether human evaluators were blinded to the model identity. Without these details, differences between methods may reflect inconsistent experimental conditions rather than the adaptation technique itself.

TABLE VII
Operational Checklist for an LLM Adapter Service

Area	Control
Compatibility	Pin base model, tokenizer and adapter versions
Access	Apply authentication, role-based permissions and rate limits
Input handling	Validate size, format and supported task scope
Output handling	Validate schema; flag uncertain or unsafe outputs
Privacy	Minimise logs, encrypt sensitive data and set retention limits
Monitoring	Track drift, latency, failures and reviewed error samples
Recovery	Maintain rollback artefacts and incident procedures

Deploy with monitoring, rollback and audit trail

VIII. Deployment and Operational Considerations

A fine-tuned model is one component in a larger application. A production system typically includes input validation, access control, prompt construction, model inference, output validation, logging and a human escalation path. The adapter should be deployed with the exact base-model version against which it was trained. Updating the base model independently can change tokenisation, representations or output behaviour and should be treated as a new compatibility test.

Latency and throughput depend on more than the number of trainable adapter parameters. Sequence length, batch size, attention implementation, quantisation kernels, hardware memory bandwidth and concurrent requests all affect the serving profile. Merging LoRA weights into the base weights may simplify inference in some deployments, while keeping adapters separate supports switching among domain-specific variants. The appropriate choice depends on whether the application prioritises serving simplicity, adapter interchangeability or runtime flexibility.

Monitoring should look for changes in input distribution, user feedback, refusal patterns, output length, unsupported claims and latency. Logs should be minimised and protected because prompts may contain personal or confidential information. A rollback plan should preserve a known-good model and configuration. These operational safeguards are especially important when the model supports decisions rather than merely drafting text.

IX. Applications

Fine-tuned LLMs are deployed across a wide range of industry and research settings:

- **Healthcare:** clinical note summarisation, medical question answering and literature review; PEFT models are preferred because they train on small, sensitive clinical datasets without the cost or data-exposure risk of full fine-tuning.
- **Legal and compliance:** contract analysis, document summarisation, clause extraction and regulatory checks, where precise terminology and citation of source clauses are essential.
- **Customer support:** domain-aware conversational agents fine-tuned on historical tickets and product documentation, reducing escalation to human agents.

TABLE VIII
DOMAIN-SPECIFIC ADAPTATION PRIORITIES

Domain	Typical task	Critical validation focus
Healthcare	Clinical note summary	Negation, chronology, omissions, expert review
Legal	Contract analysis	Clause traceability, exact terminology, jurisdiction limits
Support	Ticket response	Policy consistency, escalation and tone
Software	Code explanation	Executability, tests, security and dependency awareness
Education	Guided explanation	Correctness, learner level and feedback quality
Finance	Report synthesis	Numeric fidelity, date alignment and source provenance
Research	Literature synthesis	Citation integrity, uncertainty and coverage

- **Software engineering:** code completion, bug detection, automated documentation and code review, often via instruction tuning on code–comment pairs.
- **Education:** personalised explanations, adaptive feedback and grading support adapted to learner profiles.
- **Finance:** report summarisation, sentiment analysis of market news and fraud-pattern detection in transaction narratives.
- **Scientific research:** literature synthesis, hypothesis generation and structured data extraction from papers.

A. Application Requirements by Domain

Although the same underlying model family may be adapted to several domains, the acceptable error profile differs substantially. A customer-support assistant may prioritise consistency with product documentation and correct escalation. A legal document assistant must preserve clause wording, identify the source passage and avoid presenting a generated interpretation as legal advice. In scientific research, traceability to the cited paper and accurate representation of uncertainty are central. Healthcare summarisation requires particular attention to negation, dosage, timing and the distinction between a confirmed diagnosis and a possibility mentioned in the record.

The domain comparison illustrates why a single generic benchmark is insufficient. Fine-tuning can improve terminology and response structure, but it does not automatically provide access to current facts or guarantee that every generated statement is grounded. Systems that need current or source-specific evidence may combine fine-tuning with retrieval, deterministic validation and explicit citations. In such architectures, fine-tuning teaches the model how to perform the task, while retrieval supplies the evidence for a particular query.

X. CHALLENGES AND LIMITATIONS

- **Computational and memory cost:** even PEFT retains training memory overhead from gradient computation and backpropagation [10].
- **Catastrophic forgetting:** fine-tuning on narrow domain data can degrade general capabilities acquired in pre-training.
- **Hallucination and reliability:** fine-tuned models can still produce plausible but incorrect output, which is risky in healthcare and law.

TABLE IX
METHOD SELECTION UNDER COMMON PROJECT CONSTRAINTS

Project constraint	Candidate approach and qualification
Large compute budget; broad model change	Full fine-tuning; validate whether the additional flexibility is needed
Small trainable budget; modular variants	LoRA or adapters; tune rank or bottleneck size
Base-weight memory is limiting	QLoRA; verify quantisation compatibility and output quality
Very small task-specific parameter budget	Prompt/prefix tuning; assess task controllability
Response preference alignment	Instruction tuning plus RLHF, RLAIF or DPO as appropriate
Sensitive distributed data	PEFT with federated or privacy-preserving workflow; audit leakage risk

- **Bias and fairness:** fine-tuning data can introduce or amplify social, cultural or demographic biases.
- **Data scarcity and quality:** high-quality labelled data is scarce in specialised domains.
- **Evaluation difficulty:** standardised benchmarks for alignment and safety of fine-tuned models remain an open problem.
- **Rank and capacity bottlenecks:** the low-rank constraint of LoRA limits capacity, motivating variants such as DoRA [11] and AuroRA.
- **Security and privacy:** fine-tuning on proprietary or personal data raises risks of data leakage, membership inference and training-data extraction.
- **Governance and reproducibility:** the proliferation of variants complicates standardised evaluation, checkpoint provenance and reproducibility.

A limitation of the present study is that it is literature-based: no new fine-tuning experiments were run, and comparative figures in Tables II and III are drawn from published sources or computed analytically.

A. Selecting an Adaptation Strategy

A practical selection begins with constraints rather than a preference for a particular method. If the organisation has sufficient accelerator memory and needs broad changes across the model, full fine-tuning remains a possible option. If the base model must remain unchanged and multiple task-specific variants are required, adapters or LoRA offer modularity. When the frozen model weights dominate the memory budget, QLoRA can reduce their storage footprint through low-bit quantisation. Prompt tuning is attractive when minimal parameter updates and lightweight task switching are priorities. Alignment methods are selected when the objective is to change response preferences or instruction-following behaviour, and they may be combined with parameter-efficient updates.

The decision must also account for data volume and risk. Small datasets favour conservative adaptation and stronger regularisation, but they do not guarantee that PEFT will outperform other methods. High-impact tasks require a higher evaluation and review burden regardless of the number of trainable parameters. Licensing, deployment hardware, inference latency, adapter portability and the availability of domain experts should be documented as explicit selection criteria.

B. Interpreting the Parameter-Eflciency Calculation

For a square weight matrix $\square_0 \in R^{\square\times\square}$, a full update contains $\square^2$ trainable values. LoRA represents the update as the product of two

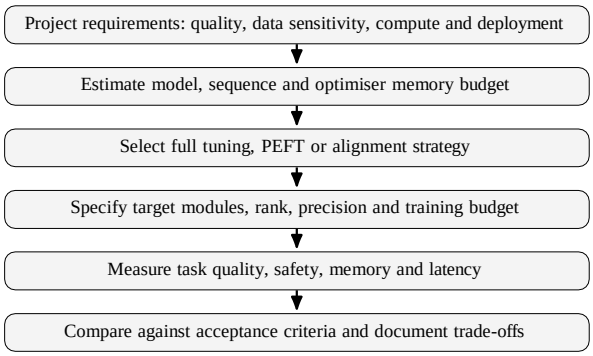


Fig. 6. Constraint-driven selection and evaluation loop for fine-tuning methods.

matrices, $\square \in R^{\square \times \square}$ and $\square \in R^{\square \times \square}$, so the number of trainable values is $\square\square + \square\square = 2\square\square$. The ratio of LoRA parameters to the full matrix is therefore

$$\square = \frac{2\square\square}{\square^2} = \frac{2\square}{\square}. \quad (5)$$

For $\square = 4096$ and $\square = 8$, this ratio is $16/4096 = 0.00390625$, or approximately 0.391%. This calculation applies to one square projection matrix. A Transformer contains many projections across multiple layers, and the total adapter count depends on which modules are targeted. The ratio is consequently a transparent local comparison, not a universal statement that every LoRA model trains exactly 0.391% of its parameters.

The same distinction applies to memory. Trainable parameter count is only one component of training memory. Activations, gradients, optimiser states, temporary workspaces and the precision used for each tensor also contribute. Quantising the frozen base weights primarily reduces their storage requirement; gradient and optimiser memory for the trainable adapter still exists. This is why reported memory savings should always specify the model size, sequence length, batch size, optimiser and hardware configuration.

XI. Discussion

The reviewed methods address different bottlenecks and should not be treated as interchangeable. Full fine-tuning offers broad parameter flexibility but has the largest training-state footprint. Prompt tuning minimises changes to the model but may have limited control over internal representations, depending on model scale and task. Adapters add modular trainable components, while LoRA expresses the update as a low-rank factorisation that can often be merged for inference. QLoRA primarily addresses the memory footprint of the frozen base model; it does not remove the need to select data carefully or evaluate the resulting adapter.

A further distinction is between capability adaptation and behavioural alignment. Domain fine-tuning can improve terminology, format and task conventions. Instruction tuning can improve the mapping from natural-language requests to useful responses. Preference optimisation attempts to shape which of several plausible responses is preferred. None of these steps alone guarantees factual correctness, privacy protection or fairness. These properties require dedicated data controls, evaluation and system-level safeguards.

The illustrative medical summarisation case makes this distinction concrete. A compact LoRA adapter may learn to produce a concise clinical style, but a safe deployment must still detect missing source information, preserve uncertainty and route ambiguous cases to a qualified professional. The analytical parameter calculation in Table III demonstrates parameter reduction for one projection matrix; it should not be interpreted as a measured end-to-end memory or latency result. Actual savings depend on the complete model, optimiser, precision, batch and sequence length.

The central implication is that efficient adaptation is an engineering process rather than a single algorithm choice. The strongest design aligns method selection, data governance, evaluation criteria and deployment controls with the consequences of model error. This is particularly important because PEFT reduces the number of updated parameters, not the responsibility associated with the model's outputs.

XII. Future Directions

Building on these challenges, several directions are likely to shape the next generation of fine-tuning methods. Further memory reduction through zeroth-order and memory-efficient optimisers such as MeZO [16] and GaLore [17] aims to bring fine-tuning within reach of consumer hardware. Higher-capacity low-rank methods will continue to address the rank bottleneck, with adaptive-rank and mixture-of-adapter approaches likely. Federated and privacy-preserving fine-tuning will enable collaboration across organisations without sharing raw data, particularly in healthcare and finance. Improved alignment techniques such as DPO [19] and RLAIF should mature further, while automated, adaptive pipelines may select the method, rank and learning rate from task characteristics and compute budget. Robust evaluation frameworks are needed to jointly assess performance, factual reliability, bias and safety. Finally, multimodal fine-tuning will extend PEFT to vision–language and speech–language foundation models.

XIII. Conclusion

This paper examined the architecture of modern LLMs and surveyed the principal techniques used to fine-tune them. The decoder-based Transformer provides the structural foundation for LLM capability, while fine-tuning—ranging from full fine-tuning to PEFT methods such as LoRA and QLoRA and alignment approaches such as instruction tuning and RLHF—determines how effectively that capability can be specialised, controlled and made safe. PEFT methods have substantially lowered the barrier to adapting large models across healthcare, legal, customer support, software engineering and education, without the prohibitive cost of full fine-tuning. Significant challenges remain, including catastrophic forgetting, hallucination, bias and the continued computational burden of training even reduced parameter sets. Future work should further reduce the resource footprint of fine-tuning, improve evaluation of alignment and safety, and develop more robust, generalisable adaptation techniques.

Acknowledgment

The authors thank Mr. Ritesh Kumar for invaluable guidance and support throughout this work, and the Department of Artificial Intelligence & Data Science for providing research resources and infrastructure.

References

[1] A. Vaswani et al., "Attention is all you need," in *Advances in NeurIPS*, vol. 30, 2017, pp. 5998–6008.
[2] T. B. Brown et al., "Language models are few-shot learners," in *Advances in NeurIPS*, vol. 33, 2020, pp. 1877–1901.
[3] H. Touvron et al., "LLaMA: Open and efficient foundation language models," arXiv:2302.13971, 2023.
[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, Minneapolis, MN, USA, 2019, pp. 4171–4186.
[5] N. Houlsby et al., "Parameter-efficient transfer learning for NLP," in *Proc. 36th Int. Conf. Machine Learning (ICML)*, Long Beach, CA, USA, 2019, pp. 2790–2799.

[6] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proc. EMNLP*, 2021, pp. 3045–3059.
[7] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," in *Proc. ICLR*, 2022.
[8] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," arXiv:2305.14314, 2023.
[9] Z. Hu et al., "LLM-Adapters: An adapter family for parameter-efficient fine-tuning of large language models," in *Proc. EMNLP*, 2023, pp. 5254–5276.
[10] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-efficient fine-tuning for large models: A comprehensive survey," arXiv:2403.14608, 2024.
[11] S. -Y. Liu et al., "DoRA: Weight-decomposed low-rank adaptation," in *Proc. ICML*, 2024 (arXiv:2402.09353).
[12] Y. Mao, K. Huang, C. Guan, G. Bao, F. Mo, and J. Xu, "DoRA: Enhancing parameter-efficient fine-tuning with dynamic rank distribution," in *Proc. ACL*, 2024.
[13] S. Hayou, N. Ghosh, and B. Yu, "LoRA+: Efficient low rank adaptation of large models," arXiv:2402.12354, 2024.
[14] Y. Li et al., "LoftQ: LoRA-fine-tuning-aware quantization for large language models," in *Proc. ICLR*, 2024.
[15] Y. Xu et al., "QA-LoRA: Quantization-aware low-rank adaptation of large language models," in *Proc. ICLR*, 2024.
[16] S. Malladi et al., "Fine-tuning language models with just forward passes," in *Advances in NeurIPS*, vol. 36, 2023.
[17] J. Zhao et al., "GaLore: Memory-efficient LLM training by gradient low-rank projection," in *Proc. ICML*, 2024.
[18] L. Ouyang et al., "Training language models to follow instructions with human feedback," in *Advances in NeurIPS*, vol. 35, 2022.
[19] R. Rafailov et al., "Direct preference optimization: Your language model is secretly a reward model," in *Advances in NeurIPS*, vol. 36, 2023.