Techniques in Machine Learning after Large Language Model Integration: A Comprehensive Survey of Emerging Paradigms, Methods, and Open Challenges

Shrikant Bhagat[1], Prathmesh Kolte[2]
[1,2]*Department of Information Technology, Mauli College of Engineering and Technology, Shegaon, Maharashtra, India*

***Abstract*—The emergence of Large Language Models (LLMs) has fundamentally reorganized the theory and practice of machine learning (ML), shifting the field from narrow, task-specific model training toward general-purpose foundation models that are adapted, augmented, and orchestrated rather than built from scratch. This survey provides a comprehensive and structured account of the techniques that have emerged, matured, or been substantially redefined as a direct consequence of LLM integration into ML pipelines. We organize the surveyed techniques into five functional categories: (i) adaptation techniques that specialize a frozen or lightly modified LLM for a task, including prompt engineering, in-context learning, and parameter-efficient fine-tuning methods such as LoRA, QLoRA, and adapters; (ii) knowledge-augmentation techniques, principally retrieval-augmented generation (RAG) and its graph- and hybrid-search variants; (iii) alignment techniques, including supervised fine-tuning, reinforcement learning from human feedback (RLHF), direct preference optimization (DPO), and constitutional AI approaches; (iv) reasoning and orchestration techniques, including chain-of-thought prompting, self-consistency, tree- and graph-of-thought search, and agentic tool-use frameworks built on planning-execution-observation loops; and (v) efficiency techniques, including knowledge distillation, quantization, pruning, and model merging, which make LLM-derived capability deployable under real-world latency, memory, and cost constraints. For each technique we describe its underlying mechanism, mathematical or algorithmic intuition, representative use cases, and trade-offs relative to classical ML approaches. We further present a multi-dimensional comparative analysis across computational cost, data requirements, controllability, and interpretability; a discussion of evaluation methodologies specific to generative and agentic systems; a survey of applied case studies drawn from enterprise, educational, and public-service domains; and a detailed treatment of open challenges including hallucination, evaluation validity, security vulnerabilities such as prompt injection, bias propagation, and the environmental cost of large-scale training and inference. The paper concludes with a research agenda covering hybrid symbolic-neural reasoning, continual and incremental post-training, standardized agentic benchmarks, and sustainable small-model specialization. This survey is intended as a rigorous reference for students, researchers, and practitioners transitioning from classical ML system design to LLM-integrated architectures.**

***Index Terms*—Large Language Models, Retrieval-Augmented Generation, Parameter-Efficient Fine-Tuning, LoRA, QLoRA, Reinforcement Learning from Human Feedback, Direct Preference Optimization, Prompt Engineering, Chain-of-Thought Reasoning, Agentic AI, Multimodal Learning, Knowledge Distillation, Model Quantization, Model Merging.**

I. INTRODUCTION

Machine learning has, since its formalization as an engineering discipline, followed a broadly consistent paradigm: a model architecture is chosen or designed for a narrow task, the model is trained from randomly initialized weights or fine-tuned from a smaller pretrained checkpoint on labeled data specific to the task, and performance is evaluated on a held-out test partition using task-specific metrics such as accuracy, F1-score, or mean squared error. This paradigm underlies decades of progress in computer vision, speech recognition, tabular prediction, and natural language processing, and it remains dominant in many production settings today.

The emergence of transformer-based Large Language Models (LLMs), trained on internet-scale text corpora

with hundreds of billions to trillions of tokens and containing tens of billions to over a trillion parameters, has introduced a qualitatively different mode of building ML systems. Once pretrained, a single LLM checkpoint exhibits broad, cross-task competence and can be redirected toward new tasks through natural-language instructions, a small number of demonstration examples, or lightweight parameter updates, without the need to design or train a new model per task. This property, often described as emergent few-shot and zero-shot generalization, has decoupled the acquisition of general capability from the process of task adaptation, and it is the underlying cause of nearly every technique surveyed in this paper.

A second and equally consequential shift is architectural: LLMs are increasingly deployed not as isolated predictors but as components embedded inside larger software systems that combine retrieval over external knowledge bases, invocation of external tools and APIs, multi-step planning, and orchestration across multiple specialized model instances. This has given rise to an entire sub-discipline of "LLM systems engineering" that sits between classical ML and traditional software engineering, encompassing retrieval architecture design, prompt and context management, agent orchestration graphs, and evaluation of open-ended, non-deterministic outputs.

This paper aims to provide a rigorous and pedagogically useful survey of the techniques that constitute this new landscape. Our contributions are threefold. First, we propose a five-category taxonomy adaptation, knowledge augmentation, alignment, reasoning and orchestration, and efficiency that organizes an otherwise fragmented literature into a coherent structure. Second, for each technique within this taxonomy we provide a mechanistic description, intuition, representative applications, and an honest account of limitations, rather than treating each technique in isolation. Third, we synthesize a comparative analysis and an open-problems discussion that is directly actionable for practitioners deciding which combination of techniques to adopt for a given system. The remainder of the paper is organized as follows: Section II reviews background and the evolution from classical ML to LLM-integrated systems; Section III presents the core survey of techniques; Section IV offers a comparative analysis; Section V discusses evaluation methodology; Section VI surveys applications; Section VII discusses open challenges; Section VIII outlines future research directions; and Section IX concludes.

II. BACKGROUND AND EVOLUTION

A. Classical Machine Learning Pipelines

Classical ML pipelines are typically composed of a data collection and labeling stage, a feature engineering or representation-learning stage, a model selection stage among algorithm families such as linear models, decision trees and ensembles, support vector machines, or convolutional and recurrent neural networks, and a supervised training stage that minimizes a task-specific loss function via gradient-based optimization. Deep learning reduced the burden of manual feature engineering by learning hierarchical representations directly from raw data, but it did not remove the requirement for task-specific architectures, substantial labeled datasets, and dedicated training runs for each new task or domain.

B. The Transformer and the Rise of Foundation Models

The transformer architecture replaced recurrence with self-attention, a mechanism that computes pairwise interactions between all positions in an input sequence and thereby models long-range dependencies more effectively and more parallelizably than recurrent networks. Scaling transformer decoders to increasingly large parameter counts and training them with a simple next-token-prediction objective over broad, diverse text corpora produced models with capabilities not explicitly present in the training objective, including multi-step arithmetic, translation, summarization, and code generation, purely as a byproduct of scale. Models exhibiting this property are commonly referred to as foundation models, reflecting their role as a reusable base upon which many downstream capabilities are built rather than as single-purpose predictors.

C. Why Integration Changes ML Practice

The practical consequence of foundation-model-style pretraining is that task adaptation can occur along a spectrum of intervention that ranges from purely input-side manipulation, such as prompt engineering, through lightweight parameter updates, such as low-rank adaptation, to full retraining, which is now reserved for only the most resource-rich organizations

building base models themselves. In parallel, because a single LLM can reason over arbitrary natural-language context, it has become practical to insert retrieved external documents, tool outputs, or the outputs of other models directly into its input, effectively allowing the model's apparent knowledge and capability to be extended at inference time without any weight modification at all. These two properties, cheap adaptation and inference-time extensibility, together explain the proliferation of the techniques surveyed in Section III.

III. CORE TECHNIQUES AFTER LLM INTEGRATION

We organize the techniques surveyed in this section into five functional categories, summarized in Fig. 1 (described textually here as figure rendering is outside the scope of this manuscript): adaptation, knowledge augmentation, alignment, reasoning and orchestration, and efficiency. Each subsection below describes one technique's mechanism, intuition, use cases, and limitations.

A. Prompt Engineering and In-Context Learning

Prompt engineering is the systematic practice of designing the natural-language input to an LLM so that it produces a desired output without any update to model parameters. In-context learning (ICL) is the closely related phenomenon whereby a model infers the intended task from a small number of input-output demonstration pairs placed directly within the prompt, effectively performing task adaptation through pattern completion rather than gradient descent. Common variants include zero-shot prompting, in which only an instruction is provided; few-shot prompting, in which several demonstrations precede the query; role- or persona-based system prompting, which sets a behavioral frame for the entire interaction; and structured-output prompting, which constrains the model to emit a fixed schema such as JSON so that its output can be consumed programmatically by downstream code.

Because prompt engineering requires no training infrastructure, it is the fastest and cheapest adaptation technique available and is typically the first technique attempted when building a new LLM-integrated feature. However, model behavior under prompting is known to be sensitive to surface-level factors such as the ordering of demonstrations, the exact phrasing of instructions, and the presence of irrelevant context, which has motivated automated prompt-optimization methods that treat the prompt itself as a search space to be explored, for example through iterative rewriting guided by validation-set performance, rather than relying purely on manual trial and error.

B. Parameter-Efficient Fine-Tuning (PEFT)

Full fine-tuning, in which every parameter of an LLM is updated via backpropagation on a task-specific dataset, is computationally expensive for models with tens of billions of parameters and carries a risk of catastrophic forgetting, whereby the model's general pretrained capabilities degrade as it over-specializes to the fine-tuning distribution. Parameter-efficient fine-tuning methods address this by freezing the base model and training only a small number of additional parameters. Low-Rank Adaptation (LoRA) inserts trainable low-rank matrices alongside the frozen weight matrices of attention and feed-forward layers, exploiting the empirical observation that the weight updates required for task adaptation tend to have low intrinsic rank; this reduces the number of trainable parameters by several orders of magnitude relative to full fine-tuning while achieving comparable downstream performance on many tasks.

QLoRA extends this idea by additionally quantizing the frozen base model's weights to 4-bit precision using a specialized data type designed to preserve the statistical properties of normally distributed weights, combined with double quantization of quantization constants and paged optimizers to manage GPU memory spikes. This combination makes it feasible to fine-tune models with tens of billions of parameters on a single consumer-grade GPU, which has substantially broadened access to LLM customization beyond organizations with large compute clusters. Related techniques include adapter modules, which insert small trainable feed-forward blocks between transformer layers, and prefix- or prompt-tuning, which prepend a sequence of trainable continuous vectors to the input rather than modifying the model's internal weights at all.

C. Retrieval-Augmented Generation (RAG)

A fundamental limitation of any LLM is that its parametric knowledge is fixed at the end of pretraining; it cannot natively answer questions about information that postdates its training data or that is

private to an organization. Retrieval-Augmented Generation addresses this by decoupling knowledge storage from knowledge use. A RAG pipeline first encodes a corpus of documents into dense vector embeddings using a separate embedding model, and indexes these embeddings in a vector database or approximate nearest-neighbor search structure. At query time, the user's query is embedded with the same model, the most semantically similar document chunks are retrieved, and these chunks are inserted into the LLM's context window alongside the query, so that the model's generation is conditioned on retrieved evidence rather than solely on its parametric memory.

Variants of RAG include hybrid retrieval, which combines dense embedding similarity with sparse keyword-based retrieval such as BM25 to improve recall on rare terms and proper nouns; re-ranking, in which an initial broad retrieval pass is followed by a more expensive but more accurate cross-encoder model that re-orders candidates before they are passed to the LLM; and graph-based RAG, in which a knowledge graph of entities and relations is constructed from the corpus and traversed to assemble multi-hop evidence chains for questions that cannot be answered from a single document chunk. RAG substantially reduces hallucination relative to purely parametric generation and provides traceability, since retrieved passages can be surfaced to the user as citations, which is particularly valuable in regulated domains such as finance, healthcare, and law.

D. Alignment: Supervised Fine-Tuning, RLHF, and DPO

A model pretrained purely to predict the next token in internet text is not necessarily helpful, harmless, or aligned with the intentions of an instruction-giver; it merely continues plausible text. Alignment techniques adjust this base behavior. Supervised fine-tuning (SFT) trains the pretrained model on curated instruction-response pairs written or vetted by human annotators, teaching the model the general format of following an instruction and producing a helpful response.

Reinforcement Learning from Human Feedback (RLHF) goes further by first training a separate reward model on human-provided pairwise preference comparisons between candidate model outputs, and then using this reward model as a proxy objective within a reinforcement learning algorithm, typically Proximal Policy Optimization (PPO), to further fine-tune the policy model so that it produces outputs the reward model scores highly, subject to a Kullback-Leibler divergence penalty that discourages the policy from drifting too far from its supervised-fine-tuned starting point. Direct Preference Optimization (DPO) simplifies this pipeline considerably by showing that the RLHF objective can be reformulated as a simple classification loss computed directly on the policy model using the preference pairs, eliminating the need for a separate reward model, an online sampling loop, and the associated instability of reinforcement learning training. Constitutional AI approaches extend alignment further by having a model critique and revise its own outputs against a written set of principles, reducing dependence on large volumes of human-labeled preference data. Together these techniques are responsible for transforming a raw pretrained base model into the instruction-following, conversational assistant models in common production use.

E. Chain-of-Thought and Structured Reasoning

Chain-of-thought (CoT) prompting instructs or demonstrates to a model that it should produce intermediate natural-language reasoning steps before committing to a final answer, which has been empirically shown to substantially improve performance on arithmetic, symbolic, and multi-step commonsense reasoning tasks relative to prompting the model to answer directly. The technique is understood to work by allocating additional computation, in the form of generated tokens, to intermediate sub-problems, and by making the model's implicit reasoning process explicit and therefore easier for it to build upon token by token.

Self-consistency extends chain-of-thought by sampling multiple independent reasoning paths for the same question at a non-zero decoding temperature and selecting the final answer that appears most frequently across samples, which reduces the variance introduced by any single reasoning path going astray. Tree-of-thought and graph-of-thought methods generalize this further by explicitly representing partial solutions as nodes in a search structure, allowing the system to evaluate intermediate states, prune unpromising branches, and backtrack, effectively combining classical search algorithms with LLM-generated

candidate moves. These structured reasoning techniques are increasingly used as a lightweight substitute for the specialized symbolic reasoning modules that earlier ML systems required to be hand-engineered for each domain.

F. Agentic AI and Tool-Use Orchestration

An agentic system uses an LLM not purely as a text generator but as a controller embedded in a loop of planning, action, and observation. The ReAct framework formalizes this by interleaving explicit reasoning traces with concrete actions, such as issuing a search query, calling a calculator, executing code, or querying a database, and then incorporating the observed results back into the model's context before deciding on the next action. This loop continues until the model determines that the task goal has been achieved or a stopping condition is met.

Multi-agent frameworks extend single-agent orchestration by decomposing a complex task across several specialized LLM instances that communicate with one another, for example a planning agent that decomposes a goal into subtasks, a retrieval agent responsible for gathering evidence, an execution agent responsible for producing artifacts such as code or documents, and a verification or critic agent responsible for checking the output before it is returned to the user. Graph-based orchestration frameworks make the control flow of such systems explicit as a directed graph of nodes and conditional edges, which improves debuggability and allows deterministic guardrails, such as mandatory human approval before an irreversible action, to be inserted at specific points in an otherwise autonomous workflow. Agentic architectures introduce new failure modes relative to single-pass generation, including compounding errors across steps and the possibility that an agent invokes a tool in an unintended or unsafe manner, which has motivated a growing body of work on agent evaluation and sandboxing.

G. Multimodal Integration

Many post-LLM-integration ML systems combine text with other modalities, most commonly images, audio, and structured tabular or code data, within a single model or a coordinated pipeline. Vision-language models align a visual encoder, typically a vision transformer, with the LLM's token embedding space, often via a lightweight learned projection layer, so that image content can be represented as a sequence of embeddings the language model can attend to alongside text tokens. This enables tasks such as visual question answering, image captioning grounded in a specific instruction, and document understanding over scanned forms, invoices, or handwritten text. Speech-integrated systems similarly combine an automatic speech recognition front-end, an LLM reasoning core, and a text-to-speech back-end to build voice-first agentic assistants, which is of particular relevance for applications targeting users who are more comfortable communicating in a spoken vernacular language than through text-based interfaces.

H. Efficiency: Distillation, Quantization, Pruning, and Merging

Because large LLMs are expensive to serve at scale in terms of latency, memory, and energy, a family of efficiency techniques has developed alongside the capability-oriented techniques described above. Knowledge distillation trains a smaller student model to reproduce the output distribution or the generated text of a larger teacher LLM, often using teacher-generated synthetic instruction-response pairs as training data for the student, which allows much of a large model's behavior to be compressed into a model an order of magnitude smaller. Quantization reduces the numerical precision used to store and compute model weights and activations, for example from 16-bit floating point to 8-bit or 4-bit integer representations, trading a small amount of accuracy for substantial reductions in memory footprint and inference latency. Structured and unstructured pruning remove redundant weights or entire attention heads and feed-forward channels that contribute little to model output, further reducing model size. Model merging, a more recent technique, combines the weights of multiple fine-tuned variants of the same base model through weighted averaging or more sophisticated interpolation in parameter space, allowing multiple specialized capabilities to be consolidated into a single deployed model without additional training.

IV. COMPARATIVE ANALYSIS

Table I compares the surveyed techniques along dimensions most relevant to a practitioner selecting among them: whether model parameters are modified,

the volume and type of data required, the typical latency and infrastructure overhead introduced, and the primary benefit obtained. Prompt engineering and in-context learning require no training and impose essentially no additional infrastructure, making them the fastest technique to deploy, but they offer the least fine-grained control over model behavior and are constrained by the size of the model's context window. Parameter-efficient fine-tuning methods occupy an intermediate position: they require a labeled dataset, typically numbering in the hundreds to low thousands of examples for narrow tasks, and modest GPU compute, in exchange for durable, low-latency task specialization that does not depend on lengthy prompts at inference time.

RLHF and DPO require the most upstream infrastructure, principally the collection of human or model-generated preference comparisons, but they operate at the level of general behavior rather than a single task and are therefore typically performed once by a model provider rather than repeatedly by downstream application developers. RAG and agentic orchestration do not modify the underlying model at all; instead, they modify the system surrounding the model, and are consequently the easiest of all techniques to update as underlying information changes, since updating a retrieval index or a tool definition does not require any retraining. Table II further compares these techniques along inference-time cost and typical latency overhead relative to a single direct LLM call, which is often the deciding factor in production system design.

Table I. Comparison of post-LLM-integration ml techniques

Technique	Params Modified	Data Need	Primary Benefit
Prompt Engineering	No	Few examples	Zero-cost, instant adaptation
RAG	No	Document corpus	Up-to-date, cited knowledge
LoRA / QLoRA	Small subset	100s–1000s labeled	Durable low-latency specialization
RLHF / DPO	Full (once)	Preference pairs	General human-aligned behavior
Chain-of-Thought	No	None	Better multi-step reasoning
Agentic / Tool Use	No	Tool/API access	Autonomous multi-step execution
Distillation / Quantization	Yes (student)	Teacher-gen. data	Compact, low-latency deployment
Model Merging	Yes (merge)	None (reuses ckpts)	Multi-capability consolidation

Table II. Inference-time cost and maintenance comparison

Technique	Extra Inference Cost	Latency Overhead	Maintenance Cost
Prompt Engineering	None	None	Low (edit text)
RAG	Embedding + search	Low–Moderate	Low (re-index)
LoRA / QLoRA	Minimal	None	Moderate (retrain adapter)
Chain-of-Thought	More output tokens	Moderate	Low
Agentic / Tool Use	Multiple LLM calls	High	Moderate–High
Distillation	None (post-hoc)	Reduced overall	High (one-time)

V. EVALUATION METHODOLOGY

Evaluating LLM-integrated ML systems requires methodology distinct from classical supervised learning, where a fixed test set and a single scalar metric such as accuracy typically suffice. Generative outputs are open-ended, meaning multiple substantially different responses may be equally correct, and agentic systems produce entire trajectories of actions rather than a single prediction, meaning correctness must sometimes be assessed at the level of the final outcome, the intermediate process, or both.

A. Automatic and Reference-Based Metrics

For tasks with a well-defined reference answer, such as extractive question answering or code generation with unit tests, automatic metrics remain useful: exact match and F1 overlap for short-answer QA, pass rates on executable test suites for code generation, and retrieval metrics such as recall@k and mean reciprocal rank for the retrieval component of a RAG pipeline. These metrics are cheap to compute and reproducible

but do not capture fluency, helpfulness, or safety of open-ended generation.

B. LLM-as-Judge and Human Evaluation

For open-ended generation, a common practice is to use a separate, typically more capable, LLM as an automated judge that scores or ranks candidate responses against a rubric, which scales far better than human annotation but introduces judge-specific biases, including a tendency to favor longer responses or responses stylistically similar to the judge model's own outputs. Human evaluation, through pairwise preference judgments or Likert-scale ratings from trained annotators, remains the most reliable ground truth for alignment-sensitive properties such as helpfulness and harmlessness, but is costly and slow to scale, which is precisely the gap that RLHF reward models and LLM-as-judge methods attempt to fill at a lower marginal cost.

C. Agentic and Trajectory-Level Evaluation

Evaluating agentic systems additionally requires assessing the trajectory of tool calls and intermediate decisions, not merely the final output, since a correct final answer reached through an unsafe or inefficient sequence of actions may still represent a system failure in production. Benchmarks for agentic evaluation therefore increasingly report task success rate under a fixed action budget, the number of steps or tool calls required, and the rate of unrecoverable or unsafe actions, in addition to final-answer correctness.

VI. APPLICATIONS

A. Enterprise Knowledge and Document Systems

Enterprise document assistants combine RAG with prompt engineering to answer natural-language questions over internal knowledge bases, policy documents, and technical manuals, returning citations back to source passages so that answers remain auditable. Resume and candidate-screening copilots apply the same retrieval pattern over resume corpora, combined with structured-output prompting, to extract, summarize, and rank candidates against a job description in a format consumable by an applicant-tracking system.

B. Software Engineering Assistants

Coding assistants combine agentic tool use, chain-of-thought reasoning, and access to a code execution sandbox to plan a change, write code, run tests, interpret failures, and iterate, closely mirroring the workflow of a human developer rather than performing single-pass code completion. This class of system illustrates the compounding of nearly every technique surveyed in this paper within a single application: retrieval over a codebase, chain-of-thought planning, tool-based execution, and often a distilled or quantized model for latency-sensitive inline suggestions alongside a larger model for complex multi-file changes.

C. Conversational and Customer-Facing Systems

Customer-facing chatbots typically rely on an RLHF- or DPO-aligned base model that has been further specialized with LoRA on domain-specific conversation transcripts, combined with RAG over a product knowledge base to ground responses in accurate, current information and reduce hallucinated claims about products, pricing, or policies.

D. Vernacular and Voice-First Public-Service Applications

Voice-first, vernacular-language assistants aimed at underserved or unbanked populations combine automatic speech recognition, an LLM reasoning core, and RAG over domain-specific knowledge such as financial products or government schemes, often layered with a lightweight behavioral scoring model that substitutes for traditional credit history in populations that lack formal credit records. Such systems illustrate multimodal integration and knowledge augmentation operating together to extend digital services to populations historically excluded from them.

E. Education and Academic Support Systems

Tutoring and academic-preparation systems apply chain-of-thought and Socratic-style prompting, in which the model is instructed to guide a learner toward a solution through incremental questioning rather than directly revealing an answer, combined with RAG over a fixed curriculum or textbook to keep explanations grounded in a specific syllabus rather than general internet knowledge.

VII. CHALLENGES AND OPEN PROBLEMS

A. Hallucination and Factual Reliability

LLMs can produce fluent, confident, and internally consistent statements that are nonetheless factually incorrect, a phenomenon commonly termed hallucination. Retrieval augmentation reduces but does not eliminate this problem, since a model may still ignore, misread, or overgeneralize beyond retrieved evidence, particularly when retrieved passages are incomplete, outdated, or mutually contradictory.

B. Evaluation Validity

As discussed in Section V, no single metric adequately captures the quality of open-ended or agentic system output, and both automatic and LLM-as-judge metrics can be gamed or biased in ways that are not immediately apparent, making genuine progress difficult to distinguish from metric overfitting.

C. Security: Prompt Injection and Tool Exploitation

Agentic, tool-augmented systems introduce a new attack surface in which adversarial content embedded in retrieved documents, web pages, or tool outputs can attempt to override the system's original instructions, a class of attack known as prompt injection. Because agentic systems can take real-world actions through tool calls, a successful injection can have consequences beyond incorrect text generation, including unauthorized data exfiltration or unintended transactions, which motivates strict permissioning, sandboxing, and human-in-the-loop approval for high-consequence actions.

D. Bias and Preference Propagation

RLHF and DPO align model behavior to the preferences encoded in human or model-generated comparison data; any systematic bias present in that data, whether demographic, cultural, or stylistic, is liable to propagate into the aligned model's behavior at scale, and detecting such propagation is complicated by the difficulty of establishing a single agreed-upon ground truth for many subjective judgments.

E. Cost, Latency, and Sustainability

Multi-step agentic pipelines multiply the number of LLM calls required per user request relative to a single direct generation, increasing both latency and inference cost, which must be weighed against the accuracy gains such pipelines provide. At a broader scale, the energy and water consumption associated with training frontier-scale models and serving them to large user populations remains a significant and only partially quantified environmental cost, one that efficiency techniques such as distillation and quantization mitigate at the level of a single deployed model but do not eliminate at the level of the overall training ecosystem.

VIII. FUTURE RESEARCH DIRECTIONS

- Hybrid symbolic-neural reasoning that couples LLMs with formal theorem provers, constraint solvers, or knowledge graphs to obtain verifiable, checkable reasoning chains rather than purely fluent but unverified text.
- Continual and incremental post-training methods that update a deployed model's knowledge or behavior without the cost and risk of full retraining, extending the benefits of parameter-efficient fine-tuning to an ongoing rather than one-time process.
- Standardized, trajectory-aware benchmarks for agentic reliability that assess safety and efficiency of the full action sequence, not merely correctness of the final answer.
- Smaller, domain-specialized foundation models produced through distillation and merging, reducing dependence on frontier-scale models for narrow enterprise applications and lowering the associated cost and environmental footprint.
- Retrieval architectures that jointly reason over structured data, such as relational tables and knowledge graphs, and unstructured text within a single coherent retrieval and reasoning framework.
- Lightweight, verifiable guardrail and monitoring models capable of supervising agentic systems in real time, detecting prompt injection attempts and unsafe tool invocations before they are executed.

IX. CONCLUSION

The integration of Large Language Models into machine learning practice has produced a distinct and rapidly maturing toolkit of techniques organized around five functional goals: adapting a general-purpose model to a specific task without full

retraining, augmenting that model with external, verifiable knowledge, aligning its behavior with human intent, extending its reasoning through structured and multi-step orchestration, and compressing its capability into deployable, efficient form factors. These techniques do not replace classical machine learning so much as they extend and recontextualize it, and modern ML practitioners increasingly require fluency across all five categories, alongside traditional model training skills, to design reliable production systems. The open problems surveyed in this paper, spanning factual reliability, evaluation validity, security, bias, and sustainability, indicate that this remains an active and unsettled area of research rather than a solved engineering discipline, and continued progress will depend on rigorous evaluation methodology as much as on new modeling techniques themselves.

REFERENCES

[1] A. Vaswani et al., "Attention Is All You Need," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2017.

[2] T. Brown et al., "Language Models Are Few-Shot Learners," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2020.

[3] J. Wei et al., "Emergent Abilities of Large Language Models," Trans. Mach. Learn. Res. (TMLR), 2022.

[4] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2020.

[5] G. Izacard and E. Grave, "Leveraging Passage Retrieval with Generative Models for Open-Domain Question Answering," in Proc. Conf. European Chapter of the Association for Computational Linguistics (EACL), 2021.

[6] E. J. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models," in Proc. Int. Conf. Learning Representations (ICLR), 2022.

[7] T. Dettmers et al., "QLoRA: Efficient Finetuning of Quantized LLMs," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2023.

[8] N. Houlsby et al., "Parameter-Efficient Transfer Learning for NLP," in Proc. Int. Conf. Machine Learning (ICML), 2019.

[9] X. L. Li and P. Liang, "Prefix-Tuning: Optimizing Continuous Prompts for Generation," in Proc. Annu. Meeting of the Association for Computational Linguistics (ACL), 2021.

[10] L. Ouyang et al., "Training Language Models to Follow Instructions with Human Feedback," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2022.

[11] R. Rafailov et al., "Direct Preference Optimization: Your Language Model Is Secretly a Reward Model," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2023.

[12] Y. Bai et al., "Constitutional AI: Harmlessness from AI Feedback," arXiv preprint arXiv:2212.08073, 2022.

[13] J. Schulman et al., "Proximal Policy Optimization Algorithms," arXiv preprint arXiv:1707.06347, 2017.

[14] J. Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2022.

[15] X. Wang et al., "Self-Consistency Improves Chain-of-Thought Reasoning in Language Models," in Proc. Int. Conf. Learning Representations (ICLR), 2023.

[16] S. Yao et al., "Tree of Thoughts: Deliberate Problem Solving with Large Language Models," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2023.

[17] S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," in Proc. Int. Conf. Learning Representations (ICLR), 2023.

[18] A. Radford et al., "Learning Transferable Visual Models from Natural Language Supervision," in Proc. Int. Conf. Machine Learning (ICML), 2021.

[19] H. Liu et al., "Visual Instruction Tuning," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2023.

[20] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," arXiv preprint arXiv:1503.02531, 2015.

[21] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," in Proc. Int. Conf. Learning Representations (ICLR), 2016.
[22] G. Ilharco et al., "Editing Models with Task Arithmetic," in Proc. Int. Conf. Learning Representations (ICLR), 2023.
[23] J. Achiam et al., "GPT-4 Technical Report," arXiv preprint arXiv:2303.08774, 2023.
[24] P. Liang et al., "Holistic Evaluation of Language Models," Trans. Mach. Learn. Res. (TMLR), 2023.
[25] L. Zheng et al., "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," in Proc. Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track, 2023.