International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 6 Issue 3 PP 09-15, 2025
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V6I3P102

A Comparative Research Study

Adapting Large Language Models: A Comparative Study of Prompt Engineering and Fine-Tuning

Srinivasa Kalyan Vangibhurathachhi
Solution Architect, Texas, USA.

Received On: 15/05/2025 Revised On: 05/06/2025 Accepted On: 17/06/2025 Published On: 05/07/2025

Abstract *- Large Language Models (LLMs) have become foundational tools in natural language processing but adapting them effectively remains a technical and practical challenge. This paper presents a comparative analysis of prompt engineering and fine-tuning as key strategies for customizing LLM behavior. Drawing on secondary data from studies and technical documents, the study evaluates each method in terms of performance, cost-efficiency, robustness, and use-case suitability. The analysis concludes that such fine-tuning results in a higher task-specific accuracy and stability, especially in domain-intensive uses, but that prompt engineering allows a more flexible over-all task as well as having a reduced resource requirement. LoRA and prefix tuning are parameters-efficient methods that are found as potential trade-offs. It has also been noted in the study that there is a variation in the interpretability and risk exposure between the approaches. The findings serve the purpose of assisting developers and researchers in the recommendation of proper adaptation strategies in accordance with the technical constraints and task requirements.*

Keywords *- Large Language Models, Prompt Engineering, Fine-Tuning, Transfer Learning, Parameter-Efficient Tuning.*

1. Introduction

Large Language Models (LLMs) have emerged as foundational components in modern artificial intelligence as they transform the way machines understand and generate human language (Kumar, 2024). Built on transformer-based architectures, these models deploy billions or trillions of tokens and parameters which enables them to accomplish tasks that were unattainable previously (Zhang et al., 2024). Raiaan et al. (2024) assert that models such as GPT-4, PaLM 2, and LLaMA 2 have been trained on vast corpora of texts and thus have demonstrated state-of-the-art performance across a wide range of tasks, from question answering and translation to summarization and coding assistance. Their remarkable generalization capabilities, especially in zero-shot and few-shot settings marks a significant shift away from narrowly trained models that once dominated natural language processing (NLP).

However, ethical issues arise as a question of practical issues due to the scale and generalizability of LLMs. Bharathi Mohan et al. (2024) argues that such models are trained on wide internet-scale data which makes them highly competent at generic applications but are usually insufficiently capable to be used in domain-specific applications without tuning. As an example, linguistic and factual behavior may need more customization compared to what can be reliably generated by pre-trained model in legal, medical or technical use cases (Nazi & Peng, 2024). Furthermore, training/fine-tuning these large architectures is computationally expensive, meaning that most institutions or developers would not train or fine-tune models via conventional methods (Menghani, 2023). The issue is not only in technical practicability but also in preserving the performance stability, minimizing biases, and ensuring that the model fits the specific task expectations.

To address these gaps, two major strategies have become central to the adaptation of LLMs: prompt engineering and fine-tuning. Prompt engineering involves designing input prompts in a way that elicits desired outputs from a model without altering its parameters (Velásquez-Henao et al., 2023). It ranges from simple task framing to advanced methods such as few-shot prompting, chain-of-thought prompting (Sahoo et al., 2024), and retrieval-augmented generation. The appeal of prompt engineering lies in its flexibility and low computational cost, particularly valuable when rapid prototyping or deployment is needed (Jiang et al., 2022). Yet, it often requires significant manual effort, and prompt performance can vary unpredictably across tasks.

To fill these gaps, prompt engineering and fine-tuning have taken center stage in the adaptation of LLMs. Prompt engineering entails designing the prompts such that the prompting is in a manner that induces the wanted response out of a model without a change to the parameters of the model (Velasquez-Henao et al., 2023). It extends from basic task framing and more advanced versions (few-shot prompting), chain-of-thought prompting (Sahoo et al., 2024), to retrieval-augmented generation. The flexibility and computational cost make the appeal of prompt engineering especially today, as an early prototype or deployment may be necessary (Jiang et al., 2022). However, it demonstrates a tendency to consume substantial manual resources and could be inconsistently performed promptly on different tasks.

On the other hand, fine-tuning refers to the process of updating internal parameters of a model, with more annotated information, whether fully re-training the model or parameter-efficient strategies like adapters or Low-Rank Adaptation (LoRA) (Wang et al., 2024). Such a strategy provides more control and better results on task-specific metrics at the expense of training overhead and deployment complexity (Han et al., 2024). There are also a couple of downsides that come with fine-tuning, including a risk of model overfitting and catastrophic forgetting, as well as the possible deterioration of general language abilities (Luo et al., 2023). To this end, this paper comparatively analyses prompt engineering and fine-tuning as the most effective approaches to making LLMs adaptive. This aim is supported by the following key objectives;

- To explore theoretical foundations and practical techniques behind prompt engineering and fine-tuning of LLMs;
- To examine reported performance and trade-offs across common benchmarks;
- To identify strengths, limitations and appropriate contexts for each approach and thereby provide guidance on how to choose between the two strategies

2. Research methods

This research deploys secondary research methodology which reanalyzes, reviews and interprets existing data (Hair et al., 2019) on prompt engineering and fine-tuning of LLMs. The study uses secondary data sources like peer-reviewed publications, open-access benchmark reports, and technical documentation provided by model developers. Core sources include empirical evaluations of large language models on tasks from the MMLU, Super GLUE and HELM benchmark suites, as well as implementation details from repositories such as GitHub and arXiv. This paper uses a comparative synthesis approach to analyze how prompt engineering and fine-tuning techniques are applied in practice. The review focuses on models widely referenced in current researchGPT-4 (Liu et al, 2023a), PaLM (Chowdhery et al., 2023) and LLaMA (Naveed et al., 2023)to ensure relevance and consistency. Special attention is given to parameter-efficient fine-tuning methods such as LoRA (Han et al., 2024), prefix tuning, and adapters, which offer practical alternatives to full model updates. By triangulating performance data, cost metrics, and implementation patterns, this paper identifies key trade-offs that shape adaptation strategies.

3. Results and Discussion

3.1. Prompt Engineering

Velasquez-Henao et al. (2023) considers prompt engineering as the process of designing inputs that can influence large language model (LLM) to generate the desired outputs by avoiding the modification of model weights. Its accessibility and limited computational requirements have made this approach one of the most widely known, mainly when working with proprietary paths such as GPT-3.5 or GPT-4 through API. Prompt engineering can take the form of manual prompting, few-shot prompting, soft prompting and retrieval augmented prompting. Manual prompting allows users to type task-specific words or phrases like a prompt, i.e., translate this sentence into French (Schulhoff et al., 2024). Manual prompts are easy to write but they may be brittle since minor alterations to their form can lead to quite different results (Kulkarni & Tupsakhare, 2024). Research has demonstrated high sensitivity with regard to wording, order and format of inputs in this approach (Webson & Pavlick, 2022). Few-shot prompting approaches improve by including a small number of example input-output tuples in the prompt (Chang et al., 2024). The method which is promoted in the initial GPT-3 paper, allows the model to transfer to new tasks without gradient optimization (Ma et al., 2023). The context window provided by the model restricts the number of examples and efficacy hinges on quality and choice of examples.

A prompt tuning approach also known as soft prompting substitutes textual prompts with set of trainable continuous embeddings (Liu et al., 2023b). The goal is to optimize these embedding via gradient descent, leaving the rest of the model fixed. Lester et al. (2021) showed that soft prompts can match or even outperform traditional fine-tuning tasks in certain scenarios, but with much fewer parameters (Wang et al., 2022). However, the approach continues to be dependent on labeled data and infrastructure in terms of tuning. Retrieval-augmented prompting embeds external knowledge into the prompt at inference time. Such systems as RETRO (Gao et al., 2023) and RAG (Zhao et al., 2024) retrieve documents or passages using a retrieval component, as a part of which they are passed with the query. This plan enhances factuality and minimizes hallucinations, and is particularly successful in dynamic knowledge systems. A comparison of the various forms of prompt engineering is depicted in Table 1 below.

Table 1: Prompt engineering methods comparison

Type	Description	Strengths	Weaknesses
Manual Prompting	Hand-crafted instructions written in plain language to guide model behavior.	Simple to implement; fast deployment; human-readable and editable.	Highly sensitive to phrasing; brittle; requires manual iteration.
Few-Shot Prompting	Includes a few labeled examples within the prompt to demonstrate task structure.	Improves generalization with minimal data; no parameter updates needed.	Context length limitations; results depend heavily on example quality.
Soft Prompting	Uses learned continuous embeddings as prompts, optimized during training.	Parameter-efficient; reusable across tasks; better performance consistency.	Requires labeled data and infrastructure; embeddings not human-readable.
Retrieval-Augmented Prompting	Appends retrieved external documents or passages to provide factual grounding.	Enhances factual accuracy; reduces hallucination; good for knowledge-rich tasks.	Retrieval may introduce noise; dependent on external knowledge base quality.

Prompt engineering, notwithstanding its strengths, has shortcomings. It can have mixed performance per task and model (Zhou et al., 2022). Prompts are not necessarily generalizable, and frequently efficacy depends on ad hoc experimentation (Strobelt et al., 2022). This is compounded by poor interpretations and reproducibility, especially where manual input has been exerted. Despite this, prompt engineering can serve as a reasonably mindful and adaptable alternative in times of rapid deployment, low resource availability, and API-driven usage, especially in cases where fine-tuning cannot be applied comprehensively (Raiaan et al., 2024).

3.2. Fine-Tuning

Fine-tuning is a process of adapting an off-the-shelf language model to a particular downstream task via re-training of some or all of its parameters, on a labelled training set (Ding et al., 2023). Panigrahi et al. (2023) explain that this would enable the model to internalize task-specific patterns, providing good performance and stability, in specialized areas, in particular. Fine-tuning can be broadly classified into two types: full fine-tuning and parameter-efficient fine-tuning. Full fine-tuning encompasses the transformation of the weights of the whole model with the aid of supervised learning. Such a strategy can result in the best task-specific accuracy because the model is optimized in the new objective (Zheng et al., 2025). Nonetheless, it has significant computation costs and storage requirements. Even on a moderate dataset, fine-tuning a language model such as GPT-3 or PaLM demands exposure to GPUs or TPUs, hyperparameter optimization with mode selection, with the risk of over fitting or catastrophic forgetting (Han et al., 2024). In addition, it builds a distinct version of the model, thereby enhancing the complexity of deployment and maintenance. To counter such drawbacks, parameter-efficient tuning schemes are devised.

Low-Rank Adaptation (LoRA), adapter layers, and prefix tuning are methods that alter only a tiny fraction of the model parameters or add other simple modules (Wang et al., 2024; Han et al., 2024). Xu et al. (2023) showed that one can update fewer than 1 percent of parameters in LoRA but match their performance to full fine-tuning. In parallel, prefix tuning adds learned embeddings to the beginning of the model input layers and directs the behavior without touching the fundamental model weights (Huber et al., 2025). These solutions significantly minimize computation requirements and ease storage and deployment. The main advantage of fine-tuning or partial versus full fine-tuning is that it can produce reliable, high-quality performance in particular tasks. Models adapted manifest more consistent behavior, conformity with domain requirements, and fewer random outputs as compared to prompt-based paradigms (Bai et al., 2024).

Also, tunable models can be optimized for latency, token and/or content filtering. Nevertheless, fine-tuning has noticeable drawbacks. Ding et al. (2023) further argue that fine-tuning commonly necessitates access to the model architecture and training infrastructure, whereas it may not be possible with proprietary models. There is also the danger of overfitting when learning on small datasets, and less flexibility in processing more than one task without training independent models (Zheng et al., 2025; Han et al., 2024). Moreover, parameter updating can lead to a dispersion of earlier acquired general abilities, which is referred to as catastrophic forgetting. Table 2 below compares full fine-tuning vs. parameter-efficient tuning.

Table 2: Full Fine-Tuning vs. Parameter-Efficient Tuning

Tuning Method	Parameter Updates	Compute Requirements	Flexibility	Use Case
Full Fine-Tuning	All model parameters	High – requires full model retraining	Low – retraining needed for each task	High-stakes, domain-specific tasks
LoRA (Low-Rank Adaptation)	Small trainable matrices injected into attention layers	Moderate – updates <1% of parameters	Medium – reusable base model	Task-specific fine-tuning with limited compute
Adapters	Small bottleneck layers added between existing layers	Moderate – minor added layers	High – adapters can be swapped easily	Multi-task models, modular deployment
Prefix Tuning	Trainable tokens prepended to inputs	Low to moderate	High – no core model changes	Quick adaptation, especially for generative tasks

3.3. Comparative Analysis

Prompt engineering and fine-tuning are two different avenues to adapt large language models (LLMs) to downstream tasks, and their relative merits rely substantially on context. In this section, their comparative strengths and trade-offs are discussed with regard to accuracy, computational expense, and practical applicability to various use cases.

3.3.1. Accuracy and benchmark performance

Fine-tuned models tend to be more accurate and reliable on task-specific benchmarks. For instance, on the Super GLUE benchmark, fine-tuned models like T5 and PaLM consistently outperform their prompted counterparts, particularly in structured tasks like textual entailment and coreference resolution (Naveed et al., 2023). More specific, prompt engineering like the few-shot variant allows competitive performance in open-ended tasks. However, it performs worse than conventional programs in highly constrained environments or on tasks demanding sophisticated thinking (Velasquez-Henao et al., 2023). Further, soft prompting techniques and retrieval augmented prompting have reduced the gap to some extent, but they remain behind supervised fine-tuning across a variety of high-stakes applications.

3.3.2. Cost-efficiency and infrastructure needs

Early-stage development and experimentation are more cost-effective in prompt engineering. It involves no retraining, minimal infrastructure, and is specifically more suitable for API-based models where users are not allowed access to model internals (Golani, 2025). On the contrary, complete fine-tuning requires parameter access and substantial GPU resources, particularly with TB-size models, such as GPT-3 or LLaMA-65B (Rostam et al., 2024). Even parameter-efficient fine-tuning (LoRA, for example) leads to lower costs but nevertheless necessitates training pipelines and labelled data (Xu et al., 2023). Budget engineering, therefore, usually succeeds on the criterion of speed and resource consumption, especially where computations are limited or where the environment is not academically oriented. Table 3 below illustrates the comparison between cost vs. control trade-offs across adaptation strategies.

Table 3: Cost vs Control Trade-offs

Strategy	Accuracy	Flexibility	Infrastructure	Cost
Prompt Engineering	Moderate	High	Low	Low
Full Fine-Tuning	High	Low	High	High
LoRA	High	Medium	Moderate	Moderate
Prefix Tuning	Moderate–High	High	Low	Low

The selection process between the two mostly depends on the usage. Prompt engineering works best in a general-purpose application and where the author needs fast iteration or is facing many lightweight applications, such as chat interfaces, trivial Q&A, or content summarizing (Hadi et al., 2023). It also provides enhanced update speed and experiments without training the model. Optimization, in its turn, is a better choice when domain-specific applications are in consideration, such as legal document classification, clinical text processing, or customer service chatbots that are trained on in-house datasets (Zheng et al., 2024). It provides superior long-term stability and performance predictability when performance testing is critical or where the demands of the user require a high accuracy of behavior and dependability. These methods may supplement one another even in a hybrid workflow. As an example, a fine-tuned base model can be refined further with prompt design to achieve greater control over tasks (Pornprasit & Tantithamthavorn, 2024). With the evolution of the ecosystem in question, it becomes even more important to know when it is better to prefer one or the other among them, or use them together, in order to deploy the models both efficiently and effectively.

3.4. Interpretability, robustness, and risk

The issues of interpretability and robustness of language models, as well as their risk are of importance in real-life implementation. Both prompt engineering and fine-tuning have their own challenges and benefits towards these aspects. Barman et al. (2024) argues that prompt engineering has greater explainability to end-users in terms of transparency. Due to the fixed model, and the input being the only variable, users can trace more easily which prompts result in which outputs. Nevertheless, this interpretability is, to some degree, superficial; the internal rationale of the model is not visible, only the prompt itself. Further, small variations in the phrasing of prompts may cause extreme changes in behavior, making it hard to form predictable mappings of input and output (Barman et al., 2024).

On its part, fine-tuning yields a model with more consistent behavior across inputs, but a lesser ability to explain why given inputs are yielding given outputs. The adjustments are incorporated into multi-millions or even billions of refined weights, and post hoc analysis is complicated in the absence of special tools to interpret results (Ding et al., 2023). Despite this, having the idea to train the model directly on a goal is often more conducive to structured error analysis and per-input-and/or-per-potential-error-type performance evaluation (Zheng et al., 2025). Regarding robustness, the fine-tuned model has a better chance of being stable in its responses, particularly when subjected to adversarial prompts or when the input is ambiguous. Prompt-based approaches are more brittle and tend to break in extremes or when applied outside their specific contexts.

Table 4: Below compares prompt-engineering and fine-tuning and can be used to decide which approach is appropriate for LLMs (Medium, 2024)

Feature	**Prompting**	**Finetuning**
Skill Level Required	Low: Requires a basic understanding of how to construct prompts.	Moderate to High: Requires knowledge of machine learning principles and model architectures.
Pricing and Resources	Low: Uses existing models, minimal computational costs.	High: Significant computational resources needed for training.
Customization	Low: Limited by the model's pre-trained knowledge and user's ability to craft effective prompts.	High: Allows for extensive customization to specific domains or styles.
Data Requirements	None: Utilizes pre-trained models without additional data.	High: Requires a large, relevant dataset for effective finetuning.
Update Frequency	Low: Dependent on retraining of the underlying model.	Variable: Dependent on when the model is retrained with new data.
Quality	Variable: Highly dependent on the skill in crafting prompts.	High: Tailored to specific datasets, leading to more relevant and accurate responses.
Use Cases	General inquiries, broad topics, educational purposes.	Specialized applications, industry-specific needs, customized tasks.
Ease of Implementation	High: Straightforward to implement with existing tools and interfaces.	Low: Requires in-depth setup and training processes.

4. Conclusion

Prompt engineering and fine tuning are the two most dominant approaches to adapting large language models to specific tasks. Both approaches have strengths and weaknesses and one is not uniformly better than the other. Prompt engineering is fast, cheap, and flexible, which makes it highly applicable in cases where model internals are not easily accessible or in cases where fast iteration is desirable. Nevertheless, it is not always stable and generalizable. However, fine-tuning offers better task specific performance and consistency of behavior, particularly in highly specialized or high stakes domains. However, it is associated with increased computational requirements, complexity of infrastructure and dangers like overfitting or loss of generality properties. To developers and researchers, differentiating the two should be affected by practical considerations like complexity of the task, resource availability, and model accessibility. In practice, the combination of both would work best in numerous situations, as it provides the advantages of both foundationally aligned fine-tuning and flexible control through prompts. Further research in hybridization, more automated prompt generation, and enhanced interpretability mechanisms should also be further investigated in order to better allow practitioners harness the unique strengths of each strategy.

5. Conflicts of Interest

I declare that there is no conflict of interest concerning the publishing of this paper.

References

[1] M. A. K. Raiaan, M. S. H. Mukta, K. Fatema, N. M. Fahad, S. Sakib, M. M. J. Mim, et al., "A review on large language models: Architectures, applications, taxonomies, open issues and challenges," IEEE Access, vol. 12, pp. 26839–26874, 2024, doi: 10.1109/ACCESS.2024.3365742.

[2] P. Kumar, "Large language models (LLMs): survey, technical frameworks, and future challenges," Artif. Intell. Rev., vol. 57, no. 10, p. 260, 2024. [Online]. Available: https://doi.org/10.1007/s10462-024-10888-y

[3] G. B. Mohan, R. Prasanna Kumar, P. Vishal Krishh, A. Keerthinathan, G. Lavanya, M. K. U. Meghana, et al., "An analysis of large language models: their impact and potential applications," Knowl. Inf. Syst., vol. 66, no. 9, pp. 5047–5070, 2024. [Online]. Available: https://doi.org/10.1007/s10115-024-02120-8

[4] Z. A. Nazi and W. Peng, "Large language models in healthcare and medical domain: A review," Informatics, vol. 11, no. 3, p. 57, Aug. 2024. [Online]. Available: https://doi.org/10.3390/informatics11030057

[5] G. Menghani, "Efficient deep learning: A survey on making deep learning models smaller, faster, and better," ACM Comput. Surv., vol. 55, no. 12, pp. 1–37, 2023. [Online]. Available: https://doi.org/10.1145/3578938

[6] J. D. Velásquez-Henao, C. J. Franco-Cardona, and L. Cadavid-Higuita, "Prompt engineering: A methodology for optimizing interactions with AI-language models in the field of engineering," Dyna, vol. 90, no. SPE230, pp. 9–17, 2023.

[7] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," arXiv preprint, arXiv:2402.07927, 2024. [Online]. Available: https://arxiv.org/abs/2402.07927

[8] E. Jiang, K. Olson, E. Toh, A. Molina, A. Donsbach, M. Terry, and C. J. Cai, "Promptmaker: Prompt-based prototyping with large language models," in Proc. CHI Conf. Human Factors Comput. Syst. Extended Abstracts, Apr. 2022, pp. 1–8. [Online]. Available: https://doi.org/10.1145/3491101.3503564

[9] L. Wang, S. Chen, L. Jiang, S. Pan, R. Cai, S. Yang, and F. Yang, "Parameter-efficient fine-tuning in large models: A survey of methodologies," arXiv preprint, arXiv:2410.19878, 2024. [Online]. Available: https://arxiv.org/abs/2410.19878

[10] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-efficient fine-tuning for large models: A comprehensive survey," arXiv preprint, arXiv:2403.14608, 2024. [Online]. Available: https://arxiv.org/abs/2403.14608

[11] Y. Luo, Z. Yang, F. Meng, Y. Li, J. Zhou, and Y. Zhang, "An empirical study of catastrophic forgetting in large language models during continual fine-tuning," arXiv preprint, arXiv:2308.08747, 2023. [Online]. Available: https://arxiv.org/abs/2308.08747

[12] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, et al., "Summary of ChatGPT-related research and perspective towards the future of large language models," Meta-radiology, vol. 1, no. 2, p. 100017, 2023. [Online]. Available: https://doi.org/10.1016/j.metrad.2023.100017

[13] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, et al., "PaLM: Scaling language modeling with pathways," J. Mach. Learn. Res., vol. 24, no. 240, pp. 1–113, 2023.

[14] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, et al., "A comprehensive overview of large language models," ACM Trans. Intell. Syst. Technol., pp. 1–69, 2023. [Online]. Available: https://doi.org/10.1145/3744746

[15] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, et al., "The prompt report: A systematic survey of prompt engineering techniques," arXiv preprint, arXiv:2406.06608, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2406.06608

[16] N. D. Kulkarni and P. Tupsakhare, "Crafting effective prompts: Enhancing AI performance through structured input design," J. Recent Trends Comput. Sci. Eng. (JRTCSE), vol. 12, no. 5, pp. 1–10, 2024.

[17] A. Webson and E. Pavlick, "Do prompt-based models really understand the meaning of their prompts?," in Proc. 2022 Conf. North Am. Chapter Assoc. Comput. Linguistics: Human Lang. Technol., pp. 2300–2344, July 2022.

[18] K. Chang, S. Xu, C. Wang, Y. Luo, X. Liu, T. Xiao, and J. Zhu, "Efficient prompting methods for large language models: A survey," arXiv preprint, arXiv:2404.01077, 2024. [Online]. Available: https://arxiv.org/abs/2404.01077

[19] H. Ma, C. Zhang, Y. Bian, L. Liu, Z. Zhang, P. Zhao, et al., "Fairness-guided few-shot prompting for large language models," in Advances Neural Inf. Process. Syst., vol. 36, pp. 43136–43155, 2023.

[20] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," ACM Comput. Surv., vol. 55, no. 9, pp. 1–35, 2023. [Online]. Available: https://doi.org/10.1145/3560815

[21] C. Wang, Y. Yang, C. Gao, Y. Peng, H. Zhang, and M. R. Lyu, "No more fine-tuning? An experimental evaluation of prompt tuning in code intelligence," in Proc. 30th ACM Joint Eur. Software Eng. Conf. Symp. Found. Software Eng., pp. 382–394, Nov. 2022.

[22] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, et al., "Retrieval-augmented generation for large language models: A survey," arXiv preprint, arXiv:2312.10997, vol.

2, no. 1, 2023. [Online]. Available: https://arxiv.org/abs/2312.10997

[23] S. Zhao, Y. Yang, Z. Wang, Z. He, L. K. Qiu, and L. Qiu, "Retrieval augmented generation (RAG) and beyond: A comprehensive survey on how to make your LLMs use external data more wisely," arXiv preprint, arXiv:2409.14924, 2024. [Online]. Available: https://arxiv.org/abs/2409.14924

[24] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," in Proc. 11th Int. Conf. Learn. Representations (ICLR), Nov. 2022.

[25] H. Strobelt, A. Webson, V. Sanh, B. Hoover, J. Beyer, H. Pfister, and A. M. Rush, "Interactive and visual prompt engineering for ad-hoc task adaptation with large language models," IEEE Trans. Vis. Comput. Graph., vol. 29, no. 1, pp. 1146–1156, 2022. doi: 10.1109/TVCG.2022.3209479

[26] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, et al., "Parameter-efficient fine-tuning of large-scale pre-trained language models," Nat. Mach. Intell., vol. 5, no. 3, pp. 220–235, 2023. [Online]. Available: https://doi.org/10.1038/s42256-023-00626-4

[27] A. Panigrahi, N. Saunshi, H. Zhao, and S. Arora, "Task-specific skill localization in fine-tuned language models," in Proc. Int. Conf. Mach. Learn. (ICML), Jul. 2023, pp. 27011–27033. PMLR.

[28] H. Zheng, L. Shen, A. Tang, Y. Luo, H. Hu, B. Du, et al., "Learning from models beyond fine-tuning," Nat. Mach. Intell., vol. 7, no. 1, pp. 6–17, 2025. [Online]. Available: https://doi.org/10.1038/s42256-024-00961-0

[29] L. Xu, H. Xie, S. Z. J. Qin, X. Tao, and F. L. Wang, "Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment," arXiv preprint, arXiv:2312.12148, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2312.12148

[30] B. Huber, G. Fazelnia, A. Damianou, S. Peleato, M. Lefarov, P. Ravichandran, et al., "Embedding-to-Prefix: Parameter-efficient personalization for pre-trained large language models," arXiv preprint, arXiv:2505.17051, 2025. [Online]. Available: https://doi.org/10.1609/aaai.v38i2.27830

[31] S. Bai, M. Zhang, W. Zhou, S. Huang, Z. Luan, D. Wang, and B. Chen, "Prompt-based distribution alignment for unsupervised domain adaptation," in Proc. AAAI Conf. Artif. Intell., vol. 38, no. 2, pp. 729–737, Mar. 2024.

[32] R. R. Golani, "LLM fine-tuning vs prompt engineering for consumer products," Int. J. Sci. Technol. (IJSAT), vol. 16, no. 2, 2025.

[33] Z. R. K. Rostam, S. Szénási, and G. Kertész, "Achieving peak performance for large language models: A systematic review," IEEE Access, 2024.

[34] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, et al., "Large language models: A comprehensive survey of its applications, challenges, limitations, and future prospects," Authorea Preprints, vol. 1, no. 3, pp. 1–26, 2023.

[35] J. Zheng, H. Hong, F. Liu, X. Wang, J. Su, Y. Liang, and S. Wu, "Fine-tuning large language models for domain-specific machine translation," arXiv preprint, arXiv:2402.15061, 2024. [Online]. Available: https://arxiv.org/abs/2402.15061

[36] C. Pornprasit and C. Tantithamthavorn, "Fine-tuning and prompt engineering for large language models-based code review automation," Inf. Softw. Technol., vol. 175, p. 107523, 2024.

[37] K. G. Barman, N. Wood, and P. Pawlowski, "Beyond transparency and explainability: On the need for adequate and contextualized user guidelines for LLM use," Ethics Inf. Technol., vol. 26, no. 3, p. 47, 2024.

[38] T. T. Kim, M. Makutonin, R. Sirous, and R. Javan, "Optimizing large language models in radiology and mitigating pitfalls: Prompt engineering and fine-tuning," RadioGraphics, vol. 45, no. 4, p. e240073, 2025.

[39] Medium, "Prompt Engineering vs Fine-tuning vs RAG," Available at: https://medium.com/@myscale/prompt-engineering-vs-finetuning-vs-rag-cfae761c6d06

[40] Zhang, N. Talukdar, S. Vemulapalli, S. Ahn & J. Wang, "Comparison of Prompt Engineering and Fine-Tuning Strategies in Large Language Models in the Classification of Clinical Notes," 2024, preprint. Doi: https://doi.org/10.1101/2024.02.07.24302444;

[41] Khan, S., Noor, S., Awan, H.H. *et al.* Deep-ProBind: binding protein prediction with transformer-based deep learning model. *BMC Bioinformatics* 26, 88 (2025). https://doi.org/10.1186/s12859-025-06101-8

[42] Govindarajan Lakshmikanthan, Sreejith Sreekandan Nair (2022). Securing the Distributed Workforce: A Framework for Enterprise Cybersecurity in the Post-COVID Era. International Journal of Advanced Research in Education and Technology 9 (2):594-602.