

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Fine-tuning Open-Source Large Language Models for Cybersecurity and IT Support: A Comparative Study and Evaluation of Fine-Tuning Techniques

CHAITHANYA VAMSHI SAI¹, IAIN RICE¹, HAITHAM MAHMOUD¹, IAN VICKERS², XAVIER SCHMOOR², NOUH SABRI ELMITWALLY^{1,3}

¹Faculty of Computing, Engineering and The Built Environment, Birmingham City University, Birmingham, B5 5JU, United Kingdom.

²Managed Enterprise Technologies Limited (METCLOUD), Birmingham, B16 8PE, United Kingdom.

³Faculty of Computers and Artificial Intelligence, Cairo University, Giza, 12613, Egypt.

Corresponding author: Nough Sabri Elmitwally (e-mail: nouh.elmitwally@bcu.ac.uk)

This research case study is supported by Birmingham City University and Managed Enterprise Technologies Limited (METCLOUD), as part of the Knowledge Transfer Partnership (KTP) project funded by Innovate UK Research and Innovation (UKRI), United Kingdom.

ABSTRACT The focus of Artificial Intelligence (AI) has shifted from questioning whether Large Language Models (LLMs) can revolutionize domain-specific tasks to understanding how they can be optimized for maximum impact. The recent rise in Natural Language Processing (NLP) has led to the development of LLMs capable of addressing complex domain-specific challenges, such as question-answering, text classification, and text translation with remarkable accuracy and efficiency. This study investigates the effectiveness of fine-tuning these models for cybersecurity and IT support tasks by comparing three advanced fine-tuning techniques: Supervised Fine-Tuning (SFT), Low-Rank Adaptation (LoRA), and Quantized Low-Rank Adaptation (QLoRA) on open-source LLMs (2B–9B parameters). Our experiments demonstrate that SFT achieves up to 90.96% accuracy in cybersecurity and IT support tasks, while LoRA and QLoRA reduce GPU memory usage and computational costs by 99% and 60%, respectively, without compromising performance. Notably, models such as Llama, Mistral, and Phi demonstrated superior adaptability and generalization across all fine-tuning methods, making them well-suited for production environments, especially those with limited computational resources. The study emphasizes the importance of selecting the right fine-tuning techniques and model architectures for domain-specific tasks in cybersecurity and IT support, particularly in question-answering systems like customer support chatbots.

INDEX TERMS

Large language models (LLMs), Generative AI (GenAI), Natural language processing (NLP), Supervised Fine-Tuning (SFT), Parameter-Efficient Fine-Tuning (PEFT), Low-Rank Adaptation (LoRA), Quantized Low-Rank Adaptation (QLoRA), Cybersecurity, IT Support.

I. INTRODUCTION

LLMs have significantly transformed the field of Natural Language Processing (NLP), offering remarkable advances in the comprehension, generation, and reasoning of human language. These models, characterized by their vast architectures composed of billions of parameters, have been shown across various applications. Notable areas of impact include creative writing, language translation, task-oriented dialogue systems and question-answering systems which leverage the model's deep understanding of language to provide contextually relevant and accurate responses [1].

As illustrated in the Figure 1, the evolution of search interest for the term "Generative AI" on a global scale, is

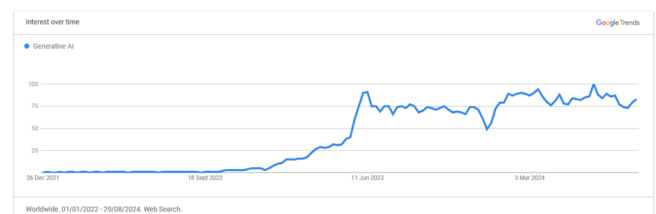


FIGURE 1. Google Trends Interest in "Generative AI" (Jan 2022 – Aug 2024). The chart depicts a notable surge in search interest for the term "Generative AI" beginning in late 2022 and peaking in early 2023. The high level of interest persisted throughout 2023 and 2024.

based on Google Trends data from January 2022 to August 2024. Initially, between January and late 2022, the interest in "Generative AI" remains relatively low and stable, reflecting limited public attention. However, starting in late 2022, there is a pronounced increase in search activity, marking the beginning of a significant surge in interest. This upward trend accelerates sharply, culminating in a peak around mid-2023, where search interest reaches its highest recorded value (index of 100), signifying the term's peak popularity. Following this peak, the interest level stabilizes but remains consistently elevated throughout the remainder of 2023 and into early 2024, with minor fluctuations. The sustained high interest suggests that "Generative AI" continued to capture widespread attention during this period, likely driven by advancements in technology, increased media coverage, and its expanding application across various industries.

One of the most promising and impactful applications of fine-tuned LLMs is in the fields of cybersecurity and IT support. In these domains, the adoption of LLMs can significantly enhance the quality and efficiency of customer support services. Recent studies have shown that LLM-powered chatbots are capable of resolving up to 40% more support requests accurately compared to traditional rule-based systems [2]. The increasing complexity of IT environments, coupled with the growing sophistication of cyber threats, has led to a surge in demand for efficient and effective support services. A recent survey revealed that over 70% of IT professionals have experienced an increase in technical queries and support requests within the past year, underscoring the need for intelligent, responsive solutions that can adapt to the evolving landscape of cybersecurity and IT support [3].

The rapid advancements in NLP have led to the development of LLMs that can perform a wide range of tasks with remarkable accuracy and contextual understanding [4]. However, fine-tuning these models for specific tasks remains a significant challenge due to the vast computational resources required and the complexity of the fine-tuning process itself [5]. To address these challenges, researchers have proposed various fine-tuning techniques, including Supervised Fine-Tuning (SFT) and Parameter-Efficient Fine-Tuning (PEFT) methods such as Low-Rank Adaptation (LoRA) [6] and Quantized Low-Rank Adaptation (QLoRA) [7]. These techniques have shown promise in improving the scalability and task-specific performance of LLMs, making them more practical for real-world applications. In the context of cybersecurity and IT support, the need for robust and adaptable LLMs is particularly urgent. As the digital landscape becomes increasingly complex, the demand for intelligent systems capable of handling technical queries, incident response, and customer IT support has grown exponentially. Traditional methods, such as rule-based or keyword-matching systems, often fail to capture the nuanced and context-dependent nature of queries in these fields. Fine-tuned LLMs, in contrast, offer the potential to deliver more accurate, contextually aware, and natural language-based solutions. This not only enhances the user experience but also significantly improves the efficiency of

cybersecurity operations and IT support services.

Despite the clear advantages of using LLMs in these domains, the process of fine-tuning these models presents several challenges. Each fine-tuning technique, including SFT, LoRA, and QLoRA, has distinct strengths and weaknesses. The effectiveness of these methods can vary depending on factors such as the size of the dataset, the complexity of the task, and the computational resources available. Understanding the trade-offs and determining the most suitable fine-tuning approach is crucial for maximizing the potential of LLMs in cybersecurity and IT support. This research paper aims to provide a comprehensive comparative analysis of fine-tuning techniques and evaluate their impact on the performance of open-source LLMs in domain-specific applications related to cybersecurity, cloud computing, and IT support. The study will systematically benchmark these techniques to identify the most effective methods for adapting LLMs to the unique challenges and requirements of these critical fields. Additionally, the research will seek to identify the most suitable open-source LLM model that can be fine-tuned to develop advanced customer support chatbots and IT service desk applications. By doing so, this study aims to offer valuable insights and practical recommendations for leveraging LLMs to drive innovation, enhance customer experience and improve operational efficiency in cybersecurity and IT support.

The structure of this paper is as follows: Section 2 provides a comprehensive background and discusses related work. Section 3 introduces and details the proposed methodology. Section 4 presents the experimental setup, followed by Section 5, which discusses the results and their implications. Section 6 provides further discussion and analysis. Finally, Section 7 summarizes the conclusions and outlines potential directions for future research.

A. BACKGROUND

The Figure 2 visually represents the hierarchical structure of Artificial Intelligence (AI) and its key sub-fields, showing the relationship between AI, Machine Learning (ML), Deep Learning (DL), Generative AI (Gen AI), and Large Language Models (LLMs). Each layer in the diagram progressively narrows in focus:

- **Artificial Intelligence (AI):** The outermost layer, AI encompasses the broad field of creating machines with human-like intelligence.
- **Machine Learning (ML):** Within AI, ML involves algorithms that learn from historical data to identify patterns and predict future outcomes.
- **Deep Learning (DL):** A subset of ML, DL utilizes deep neural networks with many layers to process complex data and generate predictive or generative outcomes.
- **Generative AI (GenAI):** A further specialization within DL, GenAI focuses on models trained to create new data such as text, images, or audio, rather than just making predictions.

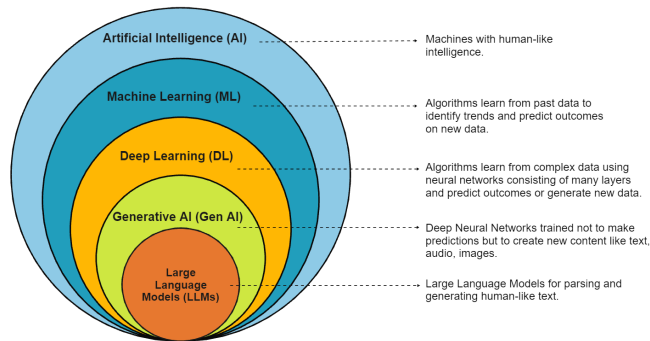


FIGURE 2. Hierarchy of Artificial Intelligence (AI) shows the nested structure of AI, such as Machine Learning (ML), Deep Learning (DL), and Generative AI (Gen AI). At the core, LLMs specialize in generating human-like text.

- **Large Language Models (LLMs):** At the core, LLMs are advanced neural networks specifically designed for understanding and generating human-like text, a key component of Generative AI.

LLMs such as GPT-3 [8], PaLM [9], and LLaMA [10] have demonstrated remarkable advancements in recent years, showcasing exceptional capabilities across a wide range of NLP tasks. These powerful models have the potential to revolutionize various domains, including cybersecurity and IT support, where accurate, context-aware, and responsive systems are crucial. However, a significant challenge in deploying LLMs lies in the inherent mismatch between their training objectives and the specific goals of end-users.

LLMs are typically trained to minimize the contextual word prediction error on vast, general-purpose corpora. While this approach enables the models to develop a broad understanding of language, it does not necessarily align with the nuanced requirements of real-world applications, where users expect the models to follow instructions accurately and deliver safe, task-specific outputs. To bridge this gap, researchers have proposed various fine-tuning techniques that modify LLMs to better align with user objectives, thus enhancing their practical utility.

Supervised fine tuning (SFT) is one such technique that directly addresses the mismatch between the training objectives of LLMs and the needs of users. SFT involves further training the LLM on pairs of instructions and desired outputs, thereby aligning the model's behaviour more closely with the intended use cases. This technique has been shown to significantly improve the model's ability to follow instructions and generate outputs that are both relevant and accurate. SFT as a method to inject new, out-of-domain knowledge into LLMs, focusing on recent sporting events. The study compares two dataset generation strategies, token-based and fact-based scaling, for training models to learn new information effectively [11]. Results indicate that while token-based scaling can improve question-answering accuracy, fact-based scaling provides more consistent coverage of new knowledge. The authors introduce a novel dataset generation process that enhances the factuality of LLM responses in specific

domains. Their experiments with GPT-4 show significant improvements in handling out-of-domain knowledge, demonstrating the potential of SFT in improving LLM performance in domain-specific tasks [11]. For instance, in the context of IT support, an SFT-fine-tuned model could be trained on a dataset of customer queries and corresponding resolution steps, enabling it to provide more precise and contextually appropriate responses.

In addition to SFT, PEFT techniques have been developed to optimize the performance of LLMs while reducing the computational and memory overhead typically associated with full fine-tuning. PEFT methods, such as Low-Rank Adaptation (LoRA) [6] and QLoRA [7], focus on updating only a small subset of the model's parameters, making them more efficient and scalable. These techniques allow LLMs to be adapted to specific tasks with minimal resource consumption, which is particularly important for models with billions or trillions of parameters.

LoRA, for example, modifies specific layers within the Transformer architecture by adding low-rank matrices, effectively updating only a small portion of the model's parameters. This approach has been demonstrated to achieve performance levels comparable to full fine-tuning while reducing memory usage by up to 99% [12]. Similarly, QLoRA extends this efficiency by quantizing the low-rank matrices, further reducing the memory footprint without significantly impacting the model's performance [13].

These fine-tuning techniques have been applied to various domains, including task completion, question answering, and code generation, with promising results. In cybersecurity, for instance, SFT-fine-tuned models can be trained on datasets of incident response protocols, enabling them to provide more accurate and contextually relevant guidance during security events. Meanwhile, in IT support, PEFT methods allow for the deployment of powerful LLMs on resource-constrained devices, such as mobile platforms, where full fine-tuning would be impractical.

While these techniques have shown considerable promise, they are not without challenges. SFT, for instance, may be prone to dataset bias, where the fine-tuning data does not fully capture the diversity of possible instructions, potentially limiting the model's effectiveness in less common scenarios [5]. Moreover, the choice of fine-tuning method must be carefully considered based on factors such as the size of the target dataset, the complexity of the task, and the computational resources available.

B. SIGNIFICANCE OF THE CASE STUDY

The significance of this case study lies in its potential to address critical challenges faced by the cybersecurity and IT support sectors through the application of advanced fine-tuning techniques to LLMs. As the digital landscape becomes increasingly complex, with the rise of sophisticated cyber threats and the growing demand for efficient IT support services, the need for intelligent, adaptable, and scalable solutions has never been more pressing. This study seeks

to explore the effectiveness of different fine-tuning methodologies, such as Supervised Instruction Fine-Tuning (SFT), Low-Rank Adaptation (LoRA), and QLoRA, in enhancing the performance of LLMs for these critical domains.

In the context of cybersecurity, the ability to quickly and accurately respond to incidents is paramount. Traditional methods, which often rely on rule-based systems or manual interventions, are increasingly insufficient in the face of modern threats that require rapid, context-aware decision-making. LLMs, when fine-tuned appropriately, offer a powerful alternative by enabling real-time, intelligent responses to complex security incidents. For instance, an LLM fine-tuned with SFT on datasets of incident response protocols could provide more accurate and contextually relevant guidance during security breaches, thereby improving the efficiency and effectiveness of cybersecurity operations [4].

Similarly, in the domain of IT support, the demand for solutions that can handle a diverse range of technical queries is growing. As organizations adopt more complex IT infrastructures, the volume and complexity of support requests have increased, placing additional strain on IT departments. LLMs have the potential to alleviate this burden by automating responses to common queries and assisting IT support engineers with more complex issues. By fine-tuning these models using techniques like LoRA and QLoRA, it is possible to develop models that are not only more memory-efficient but also capable of providing high-quality, domain-specific responses [6].

The importance of this case study is further underscored by the broader implications of deploying fine-tuned LLMs in these domains. Beyond the immediate benefits of improved efficiency and responsiveness, the successful application of these models could pave the way for more widespread adoption of AI-driven solutions in other critical sectors. The insights gained from this research could inform the development of best practices for fine-tuning LLMs across a range of applications, contributing to the advancement of AI technologies and their integration into essential services [1]. Moreover, the study's comparative approach to evaluating different fine-tuning techniques provides valuable insights into the trade-offs between performance, resource efficiency, and model scalability.

As LLMs continue to grow in size and complexity, understanding these trade-offs is crucial for optimizing their deployment in real-world scenarios. This research not only identifies the most effective fine-tuning methods for specific tasks but also highlights the potential limitations and challenges associated with each approach, offering a comprehensive guide for practitioners and researchers alike [14] [15].

Overall, the significance of this case study extends beyond the immediate context of cybersecurity and IT support. By systematically exploring the application of fine-tuning techniques to LLMs, this research contributes to the broader field of AI and machine learning, offering practical solutions to real-world challenges while advancing our understanding of how these powerful LLM models can be effectively tailored

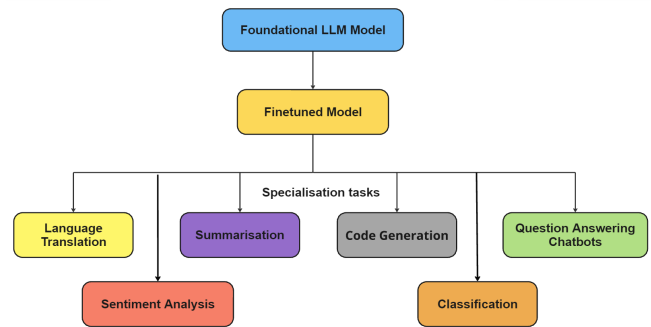


FIGURE 3. Specialization of Fine-tuned LLMs illustrating the progression from a foundational LLM model to a fine-tuned model, which is then specialized for various tasks such as Sentiment Analysis, Language Translation, Summarization, Code Generation, Question Answering Chatbots, and Classification.

to meet specific needs. The findings of this study have the potential to drive innovation, enhance operational efficiency, and improve the overall quality of customer service in critical domains, making it a valuable contribution to the ongoing development of AI technologies.

C. MOTIVATION AND CONTRIBUTIONS

The accelerating development of LLMs has created new possibilities for their use in specialized fields, such as cybersecurity and IT support. Despite these advances, a key challenge remains by adapting LLMs to meet the specific and complex needs of these fields. This article seeks to address this gap by conducting a comparative analysis of fine-tuning techniques, focusing on their influence on the performance and adaptability of open-source LLMs in cybersecurity and IT support applications.

To fully leverage LLMs in these areas, it is crucial to evaluate the effectiveness of various fine-tuning methodologies. Each technique presents distinct advantages and limitations, and understanding these is key to optimizing model performance. This research responds to the increasing need for intelligent systems capable of providing precise, context-aware, and reliable support in increasingly complex digital environments. For instance, cybersecurity should have the ability to deliver rapid and accurate responses to emerging threats is paramount, while IT support requires models that can handle diverse and technical queries with efficiency.

Traditional LLMs that are trained on general-purpose data, often lack the domain-specific insights necessary for such specialized tasks. Fine-tuning these models is not just an enhancement, it is a necessity for aligning their functionality with the unique requirements of cybersecurity and IT support. By addressing these gaps, the study aims to make LLMs more relevant and effective in real-world applications, ultimately improving their practical utility across these critical sectors. The key contributions of this article are as follows:

- 1) **Benchmarking Optimal Fine-Tuning Techniques for LLMs in Cybersecurity and IT Support:** The primary contribution of this research is to benchmark

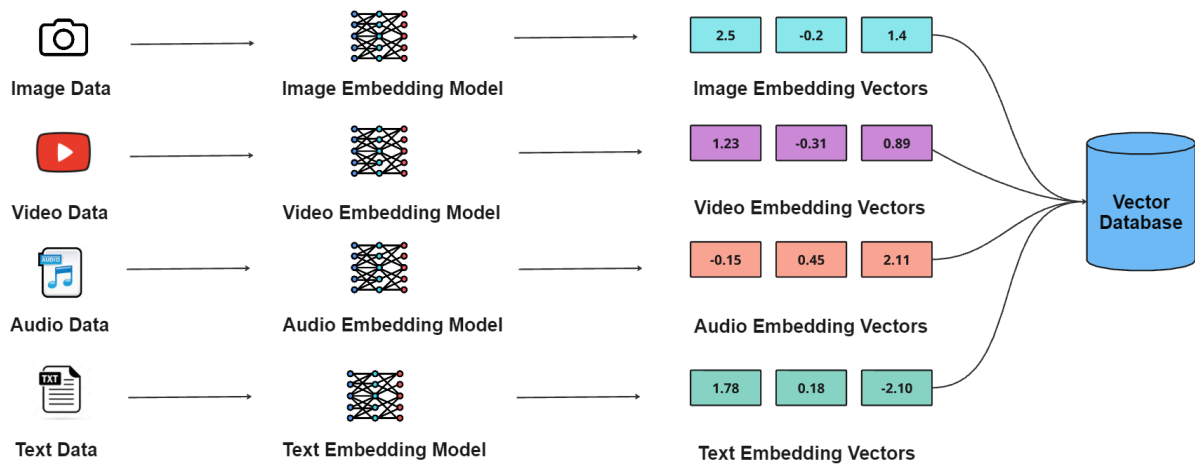


FIGURE 4. Multi-Modal Vector Embedding Representation shows how different types of data—image, video, audio, and text—are processed through their respective embedding models to generate embedding vectors. These vectors are then stored in a vector database for efficient retrieval and similarity search across modalities.

fine-tuning techniques that offer an optimal balance between efficiency and effectiveness in domain-specific tasks such as cybersecurity, cloud computing, and IT support. This study will compare methods such as Supervised Fine-Tuning (SFT), Low-Rank Adaptation (LoRA), and Quantized LoRA (QLoRA) to identify the technique that best addresses the critical requirements of these fields, including accuracy, computational resource usage, response speed, and contextual understanding.

2) Identifying the Most Suitable Open-Source LLM Architectures for Cybersecurity and IT Support:

The second contribution is to determine the most adaptable open-source LLM architectures for fine-tuning in cybersecurity and IT support. With various architectures available, differing in parameters and pre-training datasets, this study will explore which models are most effectively fine-tuned using the aforementioned techniques. The goal is to pinpoint models that are best suited for specialized tasks in these domains, ensuring high adaptability to the evolving needs of cybersecurity and cloud computing environments.

II. LITERATURE REVIEW

A. INTRODUCTION TO LARGE LANGUAGE MODELS (LLMs)

LLMs represent a significant advancement in NLP, leveraging the power of deep learning to understand, generate, and respond to human-like text. These models are based on neural networks, which are computational frameworks inspired by the structure and function of the human brain and are designed to recognise patterns within large datasets.

LLMs are specifically trained on massive amounts of text data, encompassing diverse linguistic inputs from sources like books, websites, articles, and social media platforms. The term "large" in LLMs refers to the model's extensive size in terms of the number of parameters and the vast datasets used for training. Parameters in a neural network are the

components that the model learns during training; these include weights and biases that help the model make predictions based on input data. In the context of LLMs, these parameters often number in the billions, with models such as OpenAI's GPT-3 containing 175 billion parameters [1].

As illustrated in Figure 3, LLMs have emerged as powerful tools for natural language processing tasks, exhibiting remarkable capabilities in text generation, summarisation, classification, translation, question answering, and language understanding [16]. These models are trained on vast amounts of textual data, allowing them to capture complex linguistic patterns and relationships. As a result, they have demonstrated exceptional performance across a wide range of applications, from creative writing to task completion [17].

1) Word Embeddings

Deep neural networks, including LLMs, cannot process raw text directly because text data is categorical and incompatible with the mathematical operations required for neural network training. To address this, words are represented as continuous-valued vectors through a process known as word embedding [18]. This transformation is crucial for enabling deep learning models to handle various data types, such as images, text, audio, and video, by converting them into dense vector representations [19] as illustrated in Figure 4.

Embeddings map discrete objects like words or documents into continuous vector spaces, allowing neural networks to process non-numeric data effectively [20]. While word embeddings are the most common, sentence and paragraph embeddings are also used, particularly in retrieval-augmented generation techniques [21].

2) Language Modeling

Language modelling is a fundamental task in NLP, where the goal is to predict the probability distribution of a sequence of words in a sentence. Early approaches to language modelling, such as N-grams, relied on fixed context windows and were

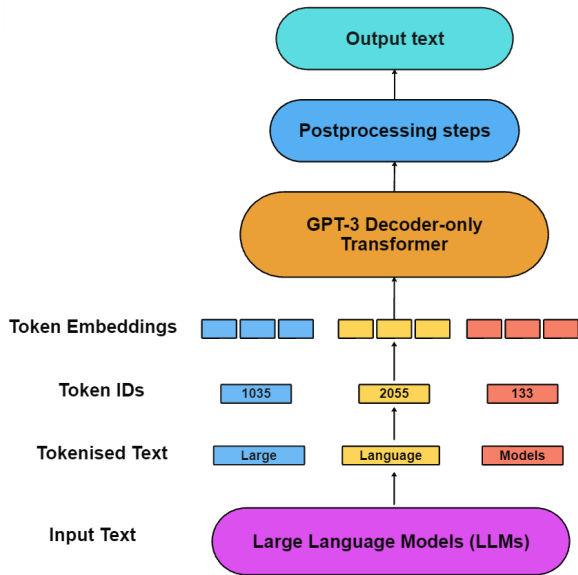


FIGURE 5. Text Tokenization Process: Input text is transformed through tokenization into token IDs, embedded into vector representations, processed by a powerful decoder-only transformer, and refined through post-processing to generate coherent output.

limited by their inability to capture long-range dependencies [22]. The advent of neural networks, particularly Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, marked a significant shift, allowing models to handle sequential data and better capture the contextual flow of language [23]. However, it was the introduction of the Transformer model by [24] that truly revolutionized language modelling. Unlike RNNs, Transformers utilize self-attention mechanisms, enabling the model to consider all words in a sentence simultaneously, which enhances its ability to understand complex linguistic patterns. This architectural innovation laid the groundwork for the development of more advanced LLMs, such as BERT [21] and GPT [14].

3) Text Tokenization

Tokenization is a critical preprocessing step in the development of LLMs. It involves breaking down text into smaller units, such as words, sub-words, or characters, which the model can then process as illustrated in the Figure 5. Traditional tokenization methods, like word-level tokenization, faced challenges with handling out-of-vocabulary words and morphological variations [25].

Sub-word tokenization methods, such as Byte Pair Encoding (BPE) and Word-Piece, were introduced to address these challenges [26]. These methods allow the model to represent rare words as compositions of sub-words, improving the model's ability to handle a diverse vocabulary. Recent models like GPT-3 utilize sophisticated tokenization techniques to manage vast corpora of text efficiently, ensuring the model can generalize across different languages and domains [1].

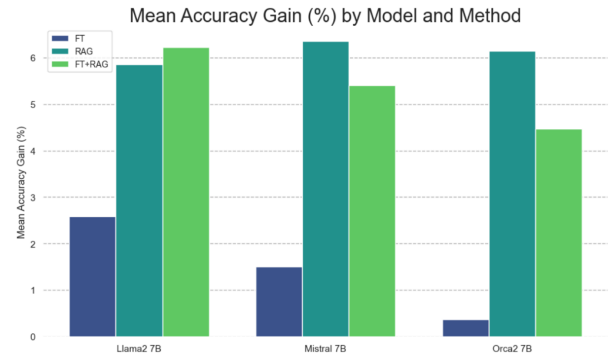


FIGURE 6. Comparative analysis of mean accuracy gains (%) across three LLMs (Llama2 7B, Mistral 7B, and Orca2 7B) using Fine-Tuning (FT), RAG, and their combination (FT+RAG). The graph illustrates the impact of these methodologies on model performance, with RAG consistently yielding superior accuracy improvements across all model architectures. [29]

B. RAG VS FINE-TUNING

Retrieval Augmented Generation (RAG) and fine-tuning represent two distinct approaches for enhancing the capabilities of LLMs. RAG combines the power of retrieval systems with generative models by first retrieving relevant documents from a large corpus and then generating answers based on these documents. This approach is particularly effective for tasks requiring factual accuracy, as it allows the model to access up-to-date information beyond its training data [27]. Fine-tuning, on the other hand, adapts the model itself to a specific task by further training on a related dataset. Unlike RAG, which relies on external information sources, fine-tuning directly adjusts the model's internal parameters to improve performance on the desired task. While both methods have their merits, RAG is often preferred when the task involves queries requiring access to current or extensive external knowledge, such as answering fact-based questions or generating content with references. Fine-tuning is more suitable for tasks where the model's ability to generate text in a specific style or perform a particular function is paramount [28].

Knowledge injection in LLMs can be achieved via RAG or fine-tuning. Each method has distinct strengths depending on cost, data, and model requirements. Fine-tuning integrates knowledge into the model's parameters, while RAG enriches responses by retrieving external context at inference.

- **Cost and Data Considerations:** Fine-tuning demands significant resources for data preparation, annotation, and retraining, especially with supervised techniques. RAG, on the other hand, offers a cost-effective alternative by integrating external data dynamically, reducing the need for extensive retraining [29].
- **Model Performance:** While fine-tuning can improve performance, it risks catastrophic forgetting, especially in unsupervised cases. RAG mitigates this by preserving base model integrity and enriching responses contextually, often yielding superior results in tasks requiring up-to-date information [28].

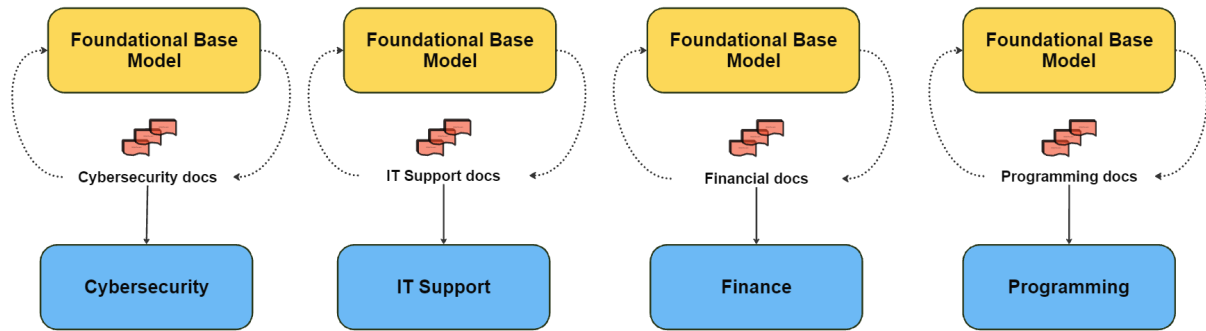


FIGURE 7. Domain-specific LLMs Applications in Cybersecurity, IT support, Finance and Programming.

- **Flexibility and Maintenance:** Fine-tuning ties an organization to a specific model, whereas RAG enables model-agnostic deployments, enhancing flexibility and reducing dependency on proprietary models. RAG also simplifies ongoing updates and maintenance, allowing real-time data integration without retraining.
- **Experimental Results:** In Figure 6, RAG consistently outperforms fine-tuning by dynamically incorporating relevant context. Fine-tuning showed improvement over the base model, but it failed to match RAG's versatility and accuracy [29].

C. DOMAIN-SPECIFIC LLMs

The advent of LLMs has revolutionized a wide array of domains by enabling advanced natural language understanding and generation capabilities. These models, when fine-tuned for specific domains, can offer highly specialized solutions that cater to the unique requirements of various industries. This section explores the applications and use cases of domain-specific LLMs, with a particular focus on cybersecurity and IT support illustrated in Figure 7.

LLMs in Cybersecurity: In the cybersecurity domain, LLMs are increasingly being leveraged to enhance threat detection, automate incident response, and provide real-time analysis of security events. These models can process and analyse large volumes of textual data, such as security logs, network traffic, and threat intelligence reports, to identify patterns indicative of cyber threats. By applying LLMs to cybersecurity, organizations can significantly reduce the time and effort required to detect and mitigate potential threats [30]. One of the key applications of LLMs in cybersecurity is the automation of threat intelligence. LLMs can be trained on vast corpora of cybersecurity literature and threat reports to generate summaries, classify threats, and even predict future attack vectors. For instance, LLMs can parse through cybersecurity advisories and generate actionable insights that security teams can use to prioritize their responses. Moreover, LLMs can assist in phishing detection by analysing email content and identifying suspicious patterns, thereby reducing the risk of social engineering attacks [31]. Despite their potential, the use of LLMs in cybersecurity is not without challenges. One of the primary concerns is the risk of adver-

sarial attacks, where malicious actors can exploit the model's vulnerabilities to evade detection or generate misleading information. Additionally, the integration of LLMs into existing cybersecurity frameworks requires careful consideration of data privacy and compliance with regulatory standards [32].

LLMs in IT Support: In the realm of IT support, LLMs have emerged as valuable tools for automating help-desk operations, providing instant technical support, and improving customer service. By training LLMs on domain-specific datasets, organizations can develop AI-driven support systems capable of understanding and resolving a wide range of IT issues. These systems can handle common queries, troubleshoot technical problems, and even guide users through complex procedures, thereby reducing the workload on human support staff. One prominent application of LLMs in IT support is the development of intelligent chatbots and virtual assistants. These systems can interact with users in natural language, providing solutions to common IT problems such as software installation, network configuration, and account management. For example, LLM-powered chatbots can assist employees with password resets or guide them through the process of setting up a virtual private network (VPN) [33].

The benefits of using LLMs in IT support extend beyond automation. These models can also improve the quality of support by providing consistent, accurate information, and by learning from past interactions to offer personalized assistance. However, the deployment of LLMs in IT support also presents challenges, including the need for continuous model updates to keep pace with evolving technologies and the potential for inaccuracies in complex or uncommon scenarios [34].

LLMs in Healthcare and Medical Research: In the healthcare domain, domain-specific LLMs have shown significant potential in improving patient care, enhancing medical research, and facilitating clinical decision-making. One notable example is Med-PaLM, developed by Google, which is designed to provide accurate and reliable answers to medical questions. Med-PaLM is a multi-modal generative model that can process and integrate various types of biomedical data, including clinical text, medical images, and genomic sequences. This capability allows Med-PaLM to assist healthcare professionals in diagnosing conditions, generating treat-

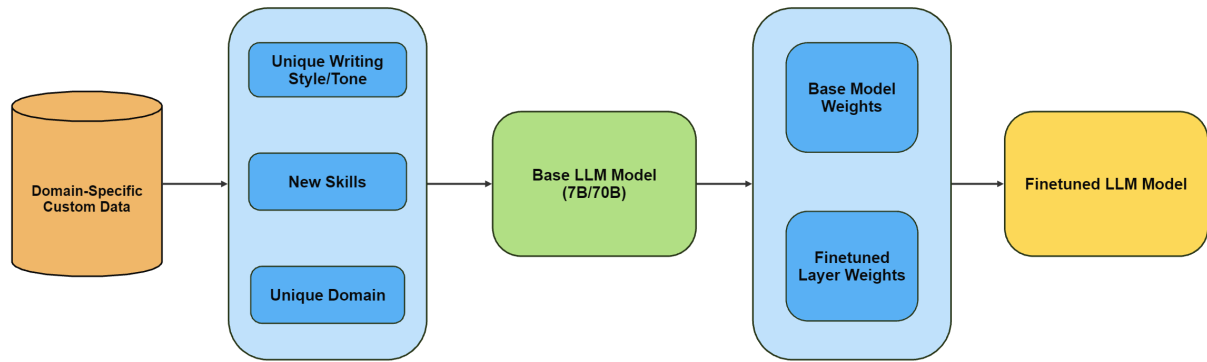


FIGURE 8. Fine-tuning Process for Base LLMs: The diagram illustrates the process of fine-tuning a base LLM model, such as a 7B or 70B parameter model, using domain-specific custom data. This process involves integrating a unique writing style or tone, incorporating new skills, and focusing on specific domain knowledge. The custom data and base model weights are used to adjust and fine-tune the layer weights, resulting in a specialized LLM model.

ment plans, and conducting complex medical research [35].

Another example is BioMedLM, a domain-specific LLM tailored for biomedical text, developed by the Stanford Centre for Research on Foundation Models (CRFM) and MosaicML. BioMedLM is particularly effective in processing and understanding large volumes of biomedical literature, such as PubMed articles. This model aids researchers by summarizing vast amounts of information, identifying key findings, and even generating hypotheses for further investigation [36].

LLMs in Finance: In the finance sector, LLMs are being deployed to enhance financial analysis, automated trading, and improve decision-making processes. One of the leading examples is BloombergGPT, an LLM developed by Bloomberg. Trained on a combination of general-purpose and domain-specific financial documents, BloombergGPT excels in financial natural language processing (NLP) tasks. It outperforms similarly-sized open models on specialized financial tasks, such as sentiment analysis, report generation, and financial forecasting, without compromising on performance in general LLM benchmarks. The application of LLMs in finance is transformative, as they enable the rapid processing and analysis of complex financial data. For example, BloombergGPT can analyse market reports, news articles, and regulatory filings to provide insights that inform trading strategies and investment decisions. Additionally, these models can assist in risk management by identifying potential market risks and generating predictive models that forecast financial trends [37].

LLMs in Programming: In the domain of programming, LLMs enhance developer productivity by providing intelligent code suggestions and automating repetitive coding tasks. GitHub Copilot is an LLM-powered tool trained on billions of lines of code from public repositories. It can transform natural language prompts into coding suggestions across dozens of programming languages, making it easier for developers to write code efficiently. Copilot helps in automating boilerplate code, generating functions based on simple descriptions, and even refactoring existing code to improve its quality [38].

D. LLM FINE-TUNING AND TECHNIQUES

The field of large language models has witnessed remarkable progress in recent years, with the development of models like OpenAI's GPT-4, Anthropic's Claude 3.5 Sonnet, Meta's Llama 3.1, Google Gemini 1.5 Pro and Mistral Large 2. To enhance the capabilities of these LLMs, researchers have explored various fine-tuning techniques, including Supervised Fine-Tuning and Parameter-Efficient Fine-Tuning methods. Fine-tuning is the process of taking a pre-trained large language model (LLM) and adapting it to a specific task or domain by training it further on a smaller, task-specific dataset. This technique is crucial in leveraging the vast knowledge embedded in LLMs, such as GPT-4, while making them effective for specific applications. Fine-tuning typically involves adjusting the model's weights slightly to better align with the particular nuances of the new task or data, rather than starting the training process from scratch [39] as shown in Figure 8.

Fine-tuning aims to adapt the general-purpose capabilities of a pre-trained LLM to perform well on specialized tasks like sentiment analysis, language translation, or customer service automation. This process is highly effective because the model has already learned a wide range of language structures, semantic meanings, and general world knowledge during its pre-training on large-scale corpora [34]. Fine-tuning LLMs is essential for several reasons.

Firstly, it allows models to perform specific tasks with higher accuracy. While a general-purpose LLM can generate coherent text, it may not excel in tasks requiring domain-specific knowledge or linguistic styles. Fine-tuning refines the model's ability to handle these specialized tasks effectively [40]. Secondly, fine-tuning is resource-efficient. Training an LLM from scratch requires enormous computational resources and vast amounts of data. By fine-tuning, we can adapt an already trained model with significantly fewer resources, reducing both time and cost.

Additionally, fine-tuning can mitigate biases present in the pre-trained model by exposing it to more representative and diverse data during the fine-tuning phase [41]. Lastly, fine-tuning supports continuous learning and model updating. As new data becomes available, models can be fine-tuned to

remain current and effective in their predictions and outputs. This is particularly important in dynamic fields such as finance, healthcare, and cybersecurity, where the relevance of data can change rapidly [42].

1) SFT

SFT is a targeted approach for aligning LLMs with specific tasks and domains using explicit instructions and labelled examples. The process begins with the preparation of a specialized dataset that contains pairs of instructions and desired outputs as shown in Figure 9. Each pair typically includes:

- **Instruction:** A clear, task-specific command or directive written in natural language.
- **Input:** The relevant data or context that the model needs to process to perform the task.
- **Output:** The desired response or action that the model should generate based on the instruction and input.

The LLM is then fine-tuned using this dataset, where it learns to map the given instructions and inputs to the corresponding outputs. This enables the model to develop a deeper understanding of the task-specific requirements and to generate accurate responses when faced with similar instructions in real-world applications. SFT not only helps in adapting LLMs to domains but also enhances their ability to handle complex tasks with greater precision. For instance, in a cybersecurity context, an LLM fine-tuned with SFT could be trained to analyse specific types of security threats based on detailed instructions, leading to more effective threat detection and response.

- **Task Specificity:** SFT allows for the precise alignment of LLMs with narrowly defined tasks, improving performance on those tasks even if they were not explicitly covered during the model's initial training.
- **Improved Zero-Shot Performance:** Fine-tuned models often exhibit enhanced zero-shot learning capabilities, meaning they can generalize better to unseen tasks within the same domain.
- **Control and Alignment:** SFT provides greater control over the model's behaviour, ensuring it adheres closely to the specific instructions and desired outcomes.
- **Efficiency:** Compared to training a model from scratch, SFT is significantly more resource-efficient, requiring less data and computational power while still achieving high levels of performance.

2) PEFT Techniques

PEFT techniques are a set of methods designed to fine-tune large language models with minimal computational resources [43]. These techniques focus on reducing the number of parameters that need to be updated during fine-tuning, making the process more efficient and accessible.

LLMs are computationally expensive to fine-tune due to their sheer size and the limitations imposed by GPU memory. For instance, a model with 7 billion parameters requires

significant memory to store and update the model weights, represented by a weight matrix W . In the back-propagation process, the model learns a matrix ΔW , which reflects the changes required to optimize the model's parameters. The weight update equation is typically represented as:

$$W_{\text{updated}} = W + \Delta W$$

Low-Rank Adaptation (LoRA): Both W and ΔW have the same dimensions, resulting in a computational burden, especially for LLMs with billions of parameters. The process of calculating and storing ΔW during training can be prohibitive in terms of both compute and memory usage [44].

The Low-Rank Adaptation (LoRA) method, proposed by [6], provides a more efficient approach to fine-tuning LLMs by reducing the computational requirements associated with calculating ΔW . Instead of explicitly computing the full ΔW , LoRA decomposes it into two smaller matrices, A and B as illustrated in the Figure 10, such that:

$$\Delta W = A \times B$$

This decomposition allows the model to bypass the need to compute the full ΔW , thereby significantly reducing memory and computational overhead. Matrices A and B have far fewer parameters, making LoRA highly efficient for fine-tuning large models. Despite this reduction, the model's performance remains competitive, as the decomposed matrices sufficiently capture the necessary weight updates [6].

LoRA offers a significant advancement in the fine-tuning of large language models, addressing the limitations posed by GPU memory constraints. By decomposing the weight update matrix ΔW into smaller matrices A and B , LoRA reduces the computational and memory overhead associated with model training. This method enables fine-tuning of LLMs without compromising performance, making it an essential tool for scalable AI model development.

Quantized Low-Rank Adaptation (QLoRA): QLoRA represents a significant advancement in the fine-tuning of LLMs, combining model quantization with low-rank adaptation techniques to address the challenges posed by memory and computational limitations. LLMs, which often contain billions of parameters, require substantial GPU memory and computational power during fine-tuning, making their deployment in resource-constrained environments challenging. QLoRA mitigates these issues by leveraging both quantization and low-rank decomposition, thereby enabling more efficient fine-tuning while maintaining high performance [7].

Model quantization is a process that reduces the precision of the model's parameters, typically from 32-bit floating-point values to lower-precision formats such as 8-bit or 4-bit integers. Quantization reduces both memory usage and computational load, making it especially useful in hardware-constrained environments. Despite the reduction in precision, advanced quantization techniques aim to preserve the model's performance, ensuring minimal degradation in accuracy [3]. By applying quantization to both the original model weights

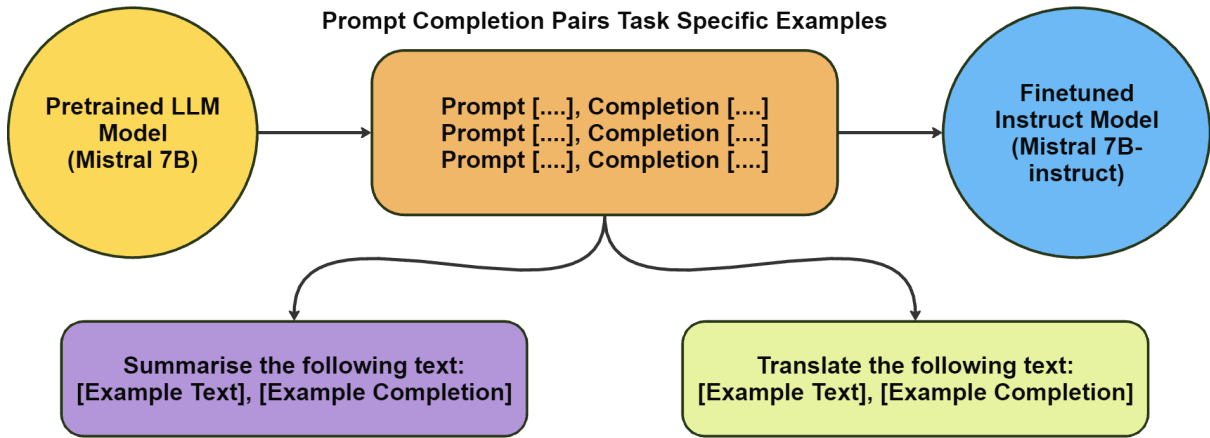


FIGURE 9. SF Workflow: This Figure shows the process of fine-tuning a pretrained LLM model, such as Mistral 7B, using task-specific prompt-completion pairs. The process involves providing examples of prompts and their corresponding completions to teach model-specific tasks like summarization and translation. The fine-tuning process results in a specialized instruction model, Mistral 7B-instruct, that is better adapted to follow specific instructions.

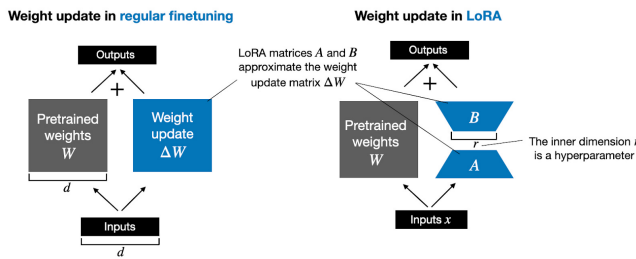


FIGURE 10. LoRA Fine-tuning Methodology [6]

and the decomposed LoRA matrices, QLoRA achieves a dramatic reduction in resource usage as shown in Figure 11.

Figure 11 provides a detailed comparison of three distinct fine-tuning methods for transformer models: Full Fine-tuning, LoRA (Low-Rank Adaptation), and QLoRA (Quantized Low-Rank Adaptation). Full Fine-tuning relies on a single, large 32-bit optimizer state to update all parameters of a 16-bit transformer base model, a process that is computationally demanding and resource-intensive due to the need to update the entire model's parameters.

In contrast, LoRA introduces 16-bit adapter layers, each with its own 32-bit optimizer state, to enable more efficient and targeted updates. This approach significantly reduces the computational burden by focusing updates on the adapter layers rather than the entire model, thereby enhancing efficiency. QLoRA builds upon the LoRA framework by quantizing the base model to 4-bit precision, which dramatically reduces memory requirements.

Additionally, QLoRA employs paged optimizers to manage memory spikes, allowing for smoother and more effective handling of computational resources. The diagram effectively illustrates the flow of gradients and parameter updates, with arrows indicating the direction of these processes, and highlights QLoRA's advanced memory management capabilities through CPU paging. This comparison underscores

the evolution from resource-intensive full fine-tuning to more memory-efficient techniques like QLoRA.

QLoRA integrates the strengths of LoRA and quantization, enabling efficient fine-tuning of LLMs with significantly reduced memory and computational requirements. The method applies quantization to both the pre-trained model and the matrices A and B used in the low-rank decomposition. This approach allows QLoRA to leverage the advantages of LoRA's parameter efficiency and the memory savings provided by quantization [6].

- **Reduced Memory Footprint:** By quantizing the original model and the LoRA matrices, QLoRA significantly reduces the memory needed for fine-tuning, making it feasible to fine-tune LLMs on smaller hardware, such as consumer-grade GPUs.
- **Faster Fine-Tuning:** The reduction in the number of trainable parameters and lower precision computations result in faster training times, allowing for quicker model adaptation.
- **High Performance:** Despite the reduction in memory and computational requirements, QLoRA retains a high level of model accuracy and performance, making it suitable for deployment in real-world applications where efficiency and scalability are essential [45].

III. METHODOLOGY

A. DATA COLLECTION AND PREPROCESSING

The data collection and preprocessing phase is a critical component of any research involving the fine-tuning of LLMs. This section details the systematic approach employed in gathering and preparing the dataset illustrated in Figure 12, which was foundational to the subsequent stages of model training and evaluation.

1) Data collection from real-world CRM knowledge base

The dataset was collected from a real-world knowledge base (METCLOUD company). This data focuses on real-life cus-

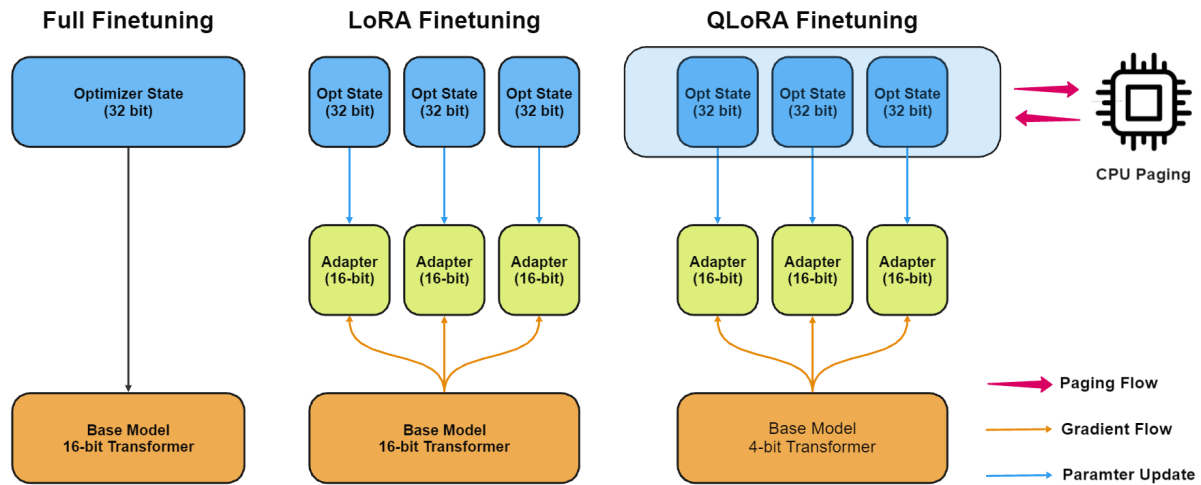


FIGURE 11. Full Fine-tuning, LoRA and QLoRA Fine-tuning Comparison: Full Fine-tuning uses a single large optimizer state to update all model parameters, while LoRA introduces adapter layers for more efficient updates, and QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimisers to handle memory spikes. Adapted from [7]

tomter support tickets from a cybersecurity and cloud computing IT services company based in the United Kingdom. METCLOUD operates within the high-demand sectors of cybersecurity and cloud computing, where customer interactions often involve complex technical issues and require precise, knowledgeable responses.

The primary data collection involved extracting customer service desk tickets recorded through both phone and email channels. These tickets were logged within the company's Customer Relationship Management (CRM) system. The dataset comprised approximately 7,000 tickets, offering a substantial corpus for analysis. The focus was on identifying and extracting frequently asked questions (FAQs) that were recurring themes in customer queries as shown in figure 13. The nature of these tickets reflected a wide array of issues commonly faced by customers of IT services, particularly in areas requiring specialized knowledge, such as IT support, cybersecurity and cloud computing.

Data Preprocessing: Following data collection, the CRM tickets underwent a rigorous preprocessing phase to ensure they were suitable for fine-tuning LLMs. Preprocessing is a crucial step in transforming raw data into a structured format that can be effectively utilized by Large Language models. The primary objective was to convert the unstructured customer support tickets into a structured question-and-answer format. This format is essential for effectively fine-tuning LLMs, enabling them to learn from real-world examples and generalize to similar queries. The preprocessing began with the extraction of relevant text from the tickets, which involved filtering out irrelevant information, such as greetings, user information, signatures, and non-informative text. Next, the text was categorized based on the nature of the query. The categories identified included common issues such as password resets, VPN connectivity problems, account lockouts, password expirations, Microsoft Outlook issues, antivirus checks, network security concerns, cloud security inquiries, and hard-

ware and software. Each category represented a specific type of frequently asked question that users commonly posed.

To ensure consistency and accuracy, each ticket was manually reviewed by a team of cybersecurity experts and IT support specialists. This manual intervention was necessary to maintain the integrity of the dataset, given the technical nature of the queries and the potential for variability in how different issues were described by customers. Once categorized, the text was further processed to create standardized question-and-answer pairs. For instance, a ticket originally describing an issue with "VPN connectivity" was restructured into a direct question such as "How can I resolve VPN connectivity issues?" followed by a corresponding answer based on the support resolution provided.

The preprocessing also involved normalization tasks, including the removal of duplicate entries, correction of typographical errors, and ensuring that technical terms were consistently represented across the dataset. These steps were essential to minimize noise and variability in the input data, which can negatively impact the performance of LLMs if left unaddressed [34]. Overall, the preprocessing phase transformed the real-world CRM data into a high-quality, structured dataset that was well-suited for fine-tuning LLMs. The careful handling of data ensured that the models trained on this dataset would be better equipped to handle the specific challenges of cybersecurity and cloud computing support, ultimately leading to more accurate and reliable Generative AI-driven customer service chat solutions.

2) Data Collection from Websites and Articles

In addition to the data obtained from customer support tickets, domain-specific knowledge was sourced from various authoritative websites and articles. The primary sources included the official documentation from the METCLOUD website, Microsoft, Hewlett Packard Enterprise (HPE), MITRE ATT&CK and Cisco etc. These sources were selected due

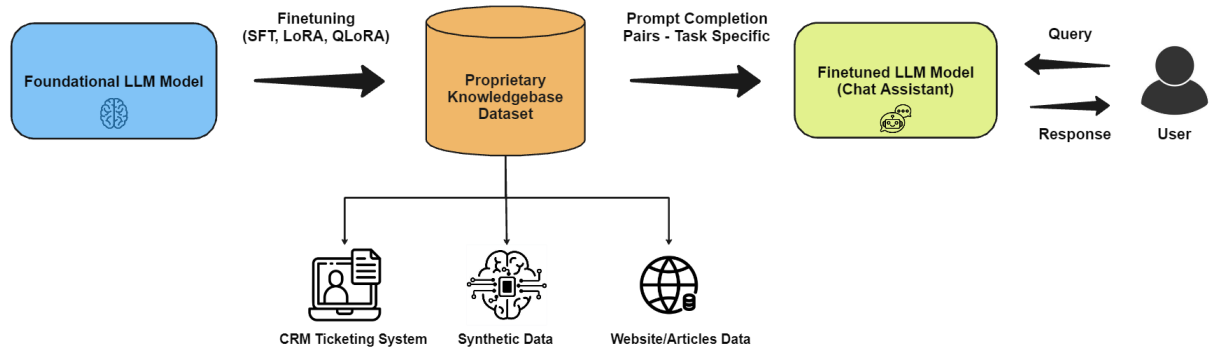


FIGURE 12. Data Collection Pipeline for LLM Fine-tuning: The diagram depicts the data collection pipeline for fine-tuning LLMs integrating diverse data sources like CRM ticketing systems, synthetic data generation, and website/article data. These sources are combined into a proprietary knowledgebase dataset, which serves as the foundation for generating task-specific prompt-completion pairs. The collected dataset supports fine-tuning of LLMs, ensuring the model is enriched with domain-relevant information to enhance its performance for specific user queries.

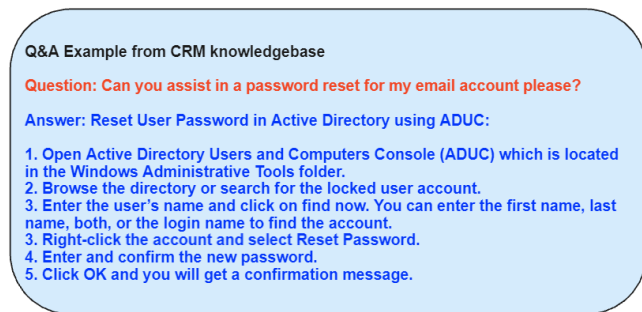


FIGURE 13. Q&A Example from CRM knowledgebase

to their extensive documentation and expertise in cybersecurity and IT support, ensuring data relevancy and accuracy. The data collected from these sources consisted of domain knowledge, glossaries, terminologies, and frequently asked questions (FAQs). This information was critical for creating a comprehensive dataset that covered the breadth of issues that users typically encounter in cybersecurity and IT support. The data was manually curated to identify key terms, definitions, and standard practices, which were then organized into a structured format suitable for model training.

Step 1. Generation of Initial Sample Inputs: The next phase involved manually generating a set of sample inputs to serve as a foundation for expanding the dataset. A total of 150 input examples were created, each consisting of two columns:

- **Question:** This column contained questions typically asked by users seeking support. These questions were designed to reflect common issues in cybersecurity and IT support, such as troubleshooting VPN connectivity issues, resetting passwords, and managing account lock-outs.
- **Context:** The context column provided relevant information that the model should consider when generating an answer. This context was derived from the knowledge collected from the mentioned websites and articles. The inclusion of a detailed context ensured that the LLM

could generate responses that were not only accurate but also contextually appropriate and aligned with best practices in the industry [1].

Step 2. Dataset Expansion Using a State-of-the-Art LLM: To expand the dataset beyond the manually generated examples, a state-of-the-art LLM was employed. In this case, OpenAI's GPT-4 Turbo (GPT-4-1106-preview) was used, known for its enhanced instruction-following capabilities and improved contextual understanding [46]. The model leveraged the initial 120 examples to generate a broader set of inputs, mimicking the format and style of the manually created samples. This automated expansion process involved prompting the LLM to generate new question-context pairs based on the existing examples. The use of an advanced LLM allowed for the creation of a large, diverse dataset, essential for training models that need to generalize across a wide range of scenarios in cybersecurity and IT support.

Step 3. Generating Outputs Using the LLM: Once the expanded dataset was created, the next step was to use the LLM to generate answers for each (question, context) pair. This step was crucial for validating the dataset and ensuring that the model could provide accurate and reliable responses. The following prompt illustrated in Figure 14 was used to guide the LLM in generating answers. This prompt was carefully crafted to ensure that the generated responses were aligned with the specific needs of the domain. By instructing the LLM to rely on the provided context, the responses remained consistent with the domain knowledge and best practices identified during data collection.

Step 4. Validation and Quality Assurance: After generating the outputs, a thorough validation process was undertaken to assess the quality and accuracy of the responses. This involved cross-referencing the generated answers with the sources and verifying their relevance and correctness. Any inconsistencies or errors were addressed through manual review and adjustments, ensuring that the final dataset was robust and reliable for fine-tuning purposes [47]. This meticulous approach to data collection, preprocessing, and expansion not only enhanced the quality of the dataset but also ensured

Q&A Example from Websites and Articles

Prompt Instruction: You are a safe, helpful, knowledgeable AI assistant and customer support expert specializing in cybersecurity and IT Support. Your primary role is to deliver precise, comprehensive, and user-friendly responses to customer queries related to the field of cybersecurity and IT Support. Leveraging your deep knowledge in cybersecurity and expertise in customer service, please generate relevant answers based on the question and the context to guarantee customer satisfaction. Do not create a new answer, but rather use the existing ones.

Question: What is two-factor authentication, and why is it important?

Context: Two-factor authentication (2FA) is a security process in which users provide two different authentication factors to verify themselves. This process is done to better protect both the user's credentials and the resources the user can access. 2FA adds an additional layer of security by making it harder for attackers to gain access to a person's devices or online accounts because knowing the victim's password alone is not enough to pass the authentication check.

Answer: Two-factor authentication (2FA) is a security process that requires users to provide two different authentication factors to verify themselves. This adds an extra layer of security to protect both the user's credentials and the resources they can access. 2FA is crucial because it makes it significantly harder for attackers to gain access to a person's devices or online accounts, even if they know the victim's password. Here's why:

1. Additional layer of security: 2FA requires a second factor, such as a phone, fingerprint, or security token, which only the user possesses.
2. Reduced risk of security breaches: Even if a hacker obtains the user's password, they still need the second factor to access the account.
3. Protection of sensitive data: 2FA provides a higher level of security for sensitive data, making it more challenging for unauthorized individuals to access it.

FIGURE 14. Q&A Example from Website and Articles Data

that the LLM could be fine-tuned effectively for real-world applications in cybersecurity and IT support.

3) Domain-specific Synthetic Data Generation using LLMs

The generation of synthetic data using LLMs is an innovative approach to addressing the challenges faced by developers and researchers who often lack sufficient datasets for fine-tuning LLMs. This section delves into the concept of synthetic data generation, the necessity for domain-specific datasets, the methodologies employed, and the detailed process involved in generating synthetic data for domains such as cybersecurity, cloud computing, and IT support.

Synthetic data generation using LLMs involves creating artificial datasets that mimic real-world data by leveraging the generative capabilities of advanced language models. These models, such as OpenAI's GPT-4o, are trained on vast corpora of text and can generate coherent and contextually relevant text based on prompts provided by the user. The synthetic data produced is often used for training, validating, and fine-tuning LLMs in various applications, particularly in scenarios where real-world data is scarce or sensitive [32]. The need for domain-specific synthetic data arises from the unique challenges posed by different fields. In highly specialized areas like cybersecurity and cloud computing, generic datasets may not capture the nuances and specific requirements of the domain. Domain-specific synthetic data ensures that the LLMs are fine-tuned with relevant, context-aware information that mirrors real-world scenarios within that field. This leads to more accurate, reliable, and contextually appropriate responses from the models, enhancing their applicability in real-world tasks [48]. Domain-specific synthetic data also addresses issues of data privacy and confidentiality, especially in sensitive fields where real data may contain personal or proprietary information. By generating synthetic data, researchers can avoid the ethical and legal implications associated with using real-world data, while still benefiting from a robust and relevant dataset for model training [49].

Synthetic Data Generation Process: The process of synthetic data generation in this study was implemented using a custom library named Pluto [50] designed to facilitate the

creation of domain-specific synthetic datasets using state-of-the-art LLMs. The process is broken down into several key steps as shown in Figure 15.

Step 1. Generate Domain Topic Tree: The first step involves generating a "topic tree" using OpenAI's GPT-4 Turbo and GPT-4o. A topic tree is a hierarchical structure where each node represents a topic or subtopic, starting from broader categories and branching into more specific areas. For this study, 104 topic trees were generated across domains such as cybersecurity frameworks, cyber law and regulations, security standards, and password management etc. This hierarchical structure was crucial for ensuring that the synthetic data was diverse and covered a wide range of subtopics without redundancy. The topic tree methodology allows for a systematic exploration of each domain, ensuring that all relevant aspects are included in the training dataset. The paths from root to leaves within the topic tree represent specific subtopics, guiding the LLM in generating diverse and relevant training samples.

Step 2. Generate Data from Topic Tree: Once the topic trees were generated, they were fed into the `create_data` function of the DataEngine, a tool within the Pluto library. This function iterates over each root-to-leaf path in the topic tree, instructing GPT-4 Turbo and GPT-4o to generate training data specific to the corresponding subtopics. This approach ensures that the generated dataset is comprehensive and covers all identified aspects of the domain, providing a rich source of training data for fine-tuning LLMs.

Step 3. Save and Format the Generated Datasets: The synthetic datasets generated through this process were saved in both Jsonl and CSV formats, facilitating easy integration into various training pipelines. Each dataset consisted of three columns:

- **System Content (Prompt):** The prompt provided to the LLM, typically includes instructions or initial context.
- **User Content (Question):** The specific question asked by the user, is derived from the topic tree.
- **Assistant Content (Answer):** The answer generated by the LLM is based on the question and context.

This structured format allowed for consistent and efficient use of the data in the fine-tuning process. The entire process was repeated across the 104 topic trees, resulting in the generation of over 10,000 domain-specific question-and-answer pairs. These pairs were then used to fine-tune the LLMs, enhancing their performance in cybersecurity, cloud computing, and IT support applications.

B. PROMPT ENGINEERING

Prompt engineering is a critical aspect of utilizing LLMs effectively. It refers to the process of designing and refining input prompts to guide the model's output in the desired direction. This technique leverages the model's extensive pre-trained knowledge to generate responses that are accurate, contextually relevant, and aligned with the task at hand. Prompt engineering serves as a bridge between raw model

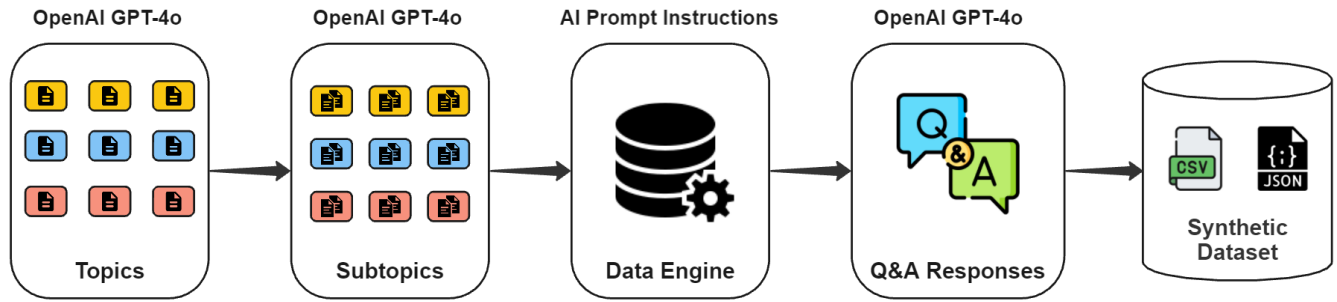


FIGURE 15. Synthetic Data Generation Pipeline: This Figure illustrates the process of generating synthetic datasets using OpenAI GPT-4o. The pipeline begins with the generation of topics and subtopics trees by GPT-4o, which are then fed into a data engine. AI prompt instructions guide the generation of Q&A responses by GPT-4o, based on these predefined topics and subtopics. The generated Q&A responses are compiled into a synthetic dataset, which can be exported in formats such as CSV or JSONL for fine-tuning LLMs.

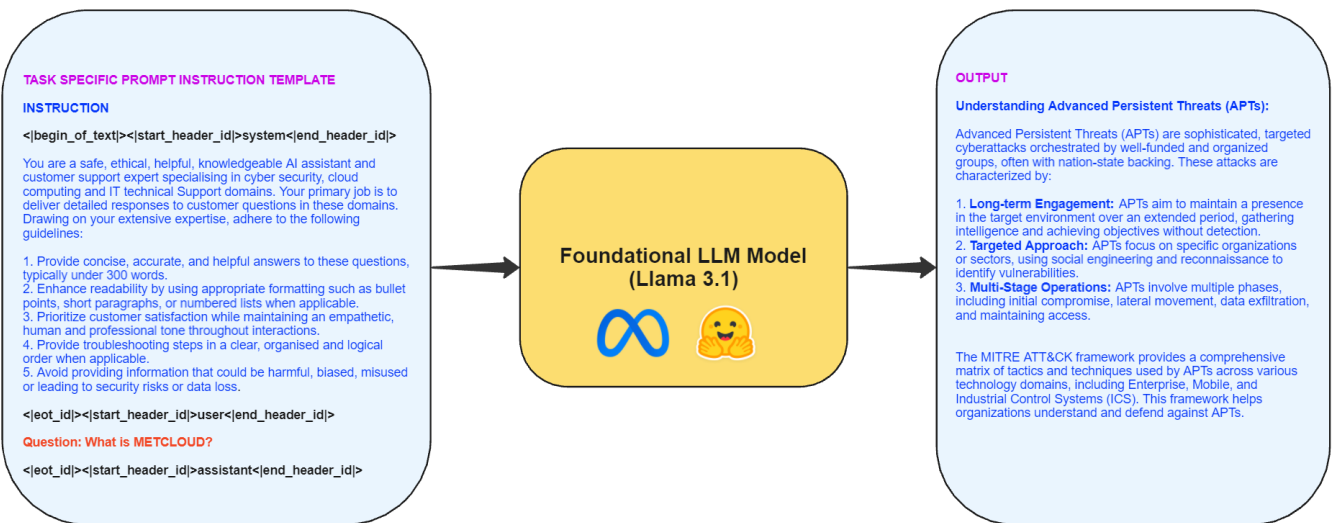


FIGURE 16. Task-Specific Instruction Prompt Template for Q&A Task with Llama 3.1: The process begins with the input prompt template, which includes detailed task-specific instructions aimed at guiding the AI assistant in cybersecurity, cloud computing, and IT support. The foundational model processes this input and generates detailed and structured responses, such as providing explanations on advanced cybersecurity threats like Advanced Persistent Threats (APTs) and MITRE Attack framework.

capabilities and practical applications, ensuring that LLMs are not only powerful but also tailored to specific needs. The necessity of prompt engineering arises from the inherent versatility of LLMs. While these models are trained on vast amounts of data and can perform a wide range of tasks, their performance is heavily dependent on the prompts they receive. Poorly designed prompts can lead to sub-optimal or irrelevant outputs, undermining the potential of the model [1]. Therefore, prompt engineering is essential to maximize the utility of LLMs, particularly in domain-specific applications where precision and relevance are paramount.

1) Task-Specific Prompts

Task-specific prompts are custom-designed prompts that are tailored to the requirements of a given task. These prompts are essential in guiding the LLM to focus on relevant aspects of the task, thereby improving the quality and accuracy of the generated responses. In the Figure 16, a question-answering task within the cybersecurity domain, prompts will be designed to prioritize factual correctness and technical

detail, ensuring the model provides responses that are both accurate and informative [10]. Task-specific prompts can vary significantly depending on the nature of the task, and their design often involves iterative testing and refinement to achieve optimal results.

2) Prompt Instruction Formats

Instruction formats are predefined templates that standardize the way instructions are presented to the LLM. These formats are particularly useful in ensuring consistency and clarity across different tasks. Open-source models typically come in two versions: a pre-trained base model (e.g., Llama-3-8B) and an instruct version (e.g., Llama-3.1-8B-Instruct). The instruct version undergoes further training using specific instruction templates that enhance the model's ability to follow detailed instructions accurately. Research has shown that using model-specific instruction templates significantly boosts performance, particularly for instruction-tuned models as shown in Figure 17. While custom templates can be applied to base models, model-specific instruction formatting usually

PROMPT INSTRUCTION TEMPLATE - LLama 3 and LLama 3.1**INSTRUCTION**

```

<|begin_of_text|><|start_header_id|>system<|end_header_id|>
{system_message}
<|eot_id|><|start_header_id|>user<|end_header_id|>
{prompt}
<|eot_id|><|start_header_id|>assistant<|end_header_id|>

```

FIGURE 17. Prompt Instruction Template - LLama 3 and LLama 3.1: This Figure defines how the prompt is divided into sections for system instructions, user inputs, and assistant responses, enabling structured guidance and response generation for the model.

results in superior performance, especially in tasks that require detailed and precise outputs [14]. The use of standardized instruction formats also facilitates the reproducibility of experiments and the comparability of results across different studies.

C. SELECTING THE FOUNDATIONAL LLM MODEL

The selection of an appropriate foundational Large Language Model (LLM) is a pivotal aspect of AI system development, necessitating a structured and meticulous approach. This section delineates a comprehensive framework designed to assist researchers and practitioners in selecting LLMs, drawing from both contemporary research and industry practices. The framework integrates various considerations such as model architecture, specialization, accessibility, licensing, evaluation criteria, and performance trade-offs, thereby offering a holistic guide for informed decision-making as illustrated in the Figure 18.

1) Foundation Model Architecture

The foundation model architecture is central to the selection process and can be classified into three primary modalities: Vision, Language, and Multi-modal as illustrated in the Figure 18. [51] suggest that this tripartite classification mirrors the dynamic evolution in AI, where models are increasingly capable of integrating multiple input and output modalities. The distinction between these modalities is crucial, as it reflects the underlying capabilities and intended applications of the models. For instance, while language models like GPT are designed primarily for text generation and understanding, multi-modal models are equipped to handle diverse inputs such as images and text, thus enabling more complex interactions [52].

2) Model Specialization

A critical decision in the LLM selection process pertains to whether the model should be task-specific, instruction-tuned, or maintain a generalist orientation. This decision has

profound implications for the model's applicability across different scenarios. Task-specific models are optimized for narrow domains, offering high performance within those domains but often at the expense of generalisability [53]. Conversely, instruction-tuned models strike a balance between specialization and flexibility, allowing them to perform well across a broader range of tasks [54]. The choice between these options should be informed by the specific requirements of the intended application, with a focus on balancing precision and versatility.

3) Model Accessibility and Licensing

The accessibility and licensing of LLMs are also paramount considerations in the selection framework. LLMs can be broadly categorized into open-source and proprietary models, each presenting distinct advantages and limitations. Open-source models, for example, offer greater transparency and adaptability, as they often provide access to both the source code and the trained weights [55]. This transparency is particularly beneficial for research purposes, where customization and reproducibility are critical. However, proprietary models may offer more robust support and streamlined deployment, albeit at the cost of higher licensing fees and potential restrictions on usage [56]. Licensing terms vary widely, ranging from research-only licenses to commercial-use licenses, which determine the extent to which a model can be adapted and deployed in real-world applications.

4) Model Evaluation Criteria

- **Latency:** For real-time applications, latency is a critical factor. Smaller models typically offer faster response times, which are essential for tasks requiring immediate feedback [33]. However, this speed may come at the cost of reduced complexity in model outputs.
- **Parameters:** The number of parameters in a model directly impacts its capabilities and resource requirements. While larger models generally exhibit superior performance, they also demand more computational power and memory, which can be prohibitive in resource-constrained environments [1].
- **Context Window:** The context window defines the model's ability to process and retain information across long sequences of text. A larger context window is advantageous for tasks that require extensive contextual understanding, such as document summarization or dialogue systems [57].
- **Cost:** The economic considerations of model deployment are particularly significant for large-scale implementations. Models with higher operational costs may offer better performance, but their long-term sustainability should be carefully evaluated [58].
- **Data Trained:** The scope and diversity of the data used to train the model are crucial in determining its knowledge base and potential biases. Models trained on large, diverse datasets are generally more robust but may also inherit biases present in the data [59].

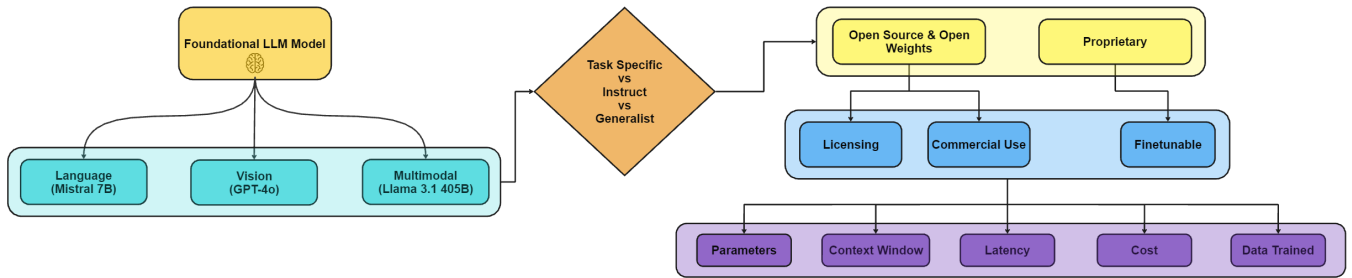


FIGURE 18. Selecting the Foundational LLM Model: The diagram outlines the selection process of a foundational LLM, categorizing models by their specialization in the language (e.g., Mistral 7B), vision (e.g., GPT-4o), or multimodal capabilities (e.g., Llama 3.1 405B). It emphasizes the distinction between task-specific, instructive, and generalist models, as well as open-source versus proprietary licensing options. Key decision factors include parameters, context window size, latency, cost, and the data on which the models were trained.

IV. EXPERIMENTAL SETUP

The experimental setup used in this article attempts to perform SFT on open-source LLMs using HPE Machine Learning Development Environment Software (HPE MLDES). HPE MLDES is a cloud-based platform providing scalable infrastructure for AI model training, supporting high computational demands for large-scale AI projects. Our experiments were conducted on HPE's AI cluster, which consisted of 8 NVIDIA A100 40GB GPUs, providing the required processing power with automatic scalability to accommodate increasing workloads.

The specific environment also included access to GenAI Studio, a GUI-driven tool integrated into MLDES, which offers Python SDK support. GenAI Studio facilitates prompt engineering and fine-tuning, allowing users to customize LLMs for specific tasks while ensuring traceability and explainability, which are critical for sensitive or proprietary applications [60]. An example of SF configuration code in HPE MLDES Python SDK is shown in Figure 19.

For low-rank adaptation (LoRA) fine-tuning, we utilized the Predibase platform [61], which is built on top of open-source technologies like Ludwig and LoRAX. Ludwig is a declarative AI framework designed to simplify the process of developing, fine-tuning, and deploying complex deep learning models, including LLMs [62]. The advantage of Ludwig is that it allows developers to engage in AI development without requiring intricate low-level coding skills. Moreover, LoRAX (LoRA exchange) enables users to deploy thousands of fine-tuned models on a single GPU, optimizing cost-efficiency while maintaining high throughput and low latency [63]. We performed LoRA fine-tuning on a system equipped with NVIDIA A100 80GB PCIe GPU, which was optimal for handling the computational complexity of fine-tuning large models. Fine-tuning recipes for each model are provided as Ludwig [62] configurations for each of the fine-tuned LLMs and can be found at <https://huggingface.co/predibase>. Figure 22 shows an example of a LoRA Fine-tuning configuration based on Ludwig.

In addition to LoRA, we also explored Quantized LoRA (QLoRA) fine-tuning, which was executed using Google Colab Pro with NVIDIA A100 GPU. This setup incorporated the use of Huggingface libraries such as Transformers, PEFT and

```
name: Llama2-7B-Chat-MET-HPE-SFT-v0.1
labels:
  - train
searcher:
  name: single
  metric: eval_loss
  max_length:
    epochs: 5
  resources:
    slots_per_trial: 6
    resource_pool: a100
  experiment_settings:
    model:
      hf_model_name: meta-llama/Llama-2-7b-chat-hf
      model_architecture: Llama-2-7b
      source: HUGGING_FACE
      model_type: STANDARD_LLM
    name: Llama2-7B-Chat-MET-HPE-SFT-v0.1
  train_config:
    data_args:
      dataset_data:
        name: MET-Data-0724-5k - generated on 1722961992838
        source: LOCAL
        data_type:
          - TEXT
        size: 32298279
        row_count: 5170
        split_row_counts: '{"test": 517, "train": 4136, "validation": 517}'
        description: "Dataset contains 5170 samples of Q&A for SFT"
        block_size: 4096
      slots_per_trial: 6
      max_steps: -1
      num_train_epochs: 5
      save_total_limit: 1
      learning_rate: 0.00005
      per_device_train_batch_size: 4
      per_device_eval_batch_size: 2
      do_train: true
      do_eval: true
      fp16: true
      bf16: false
      gradient_checkpointing: true
      deepspeed: true
      resource_pool: a100
      seed: 1337
      eval_steps: 1
      logging_steps: 1
      save_steps: 1
      logging_strategy: epoch
      evaluation_strategy: epoch
      save_strategy: epoch
      save_on_each_node: true
    eval_config: null
    model_load_config:
      torch_dtype: float16
      low_cpu_mem_usage: false
      vllm_config: null
      peft_config: null
      resource_pool: a100
```

FIGURE 19. Example of Supervised Fine-tuning configuration

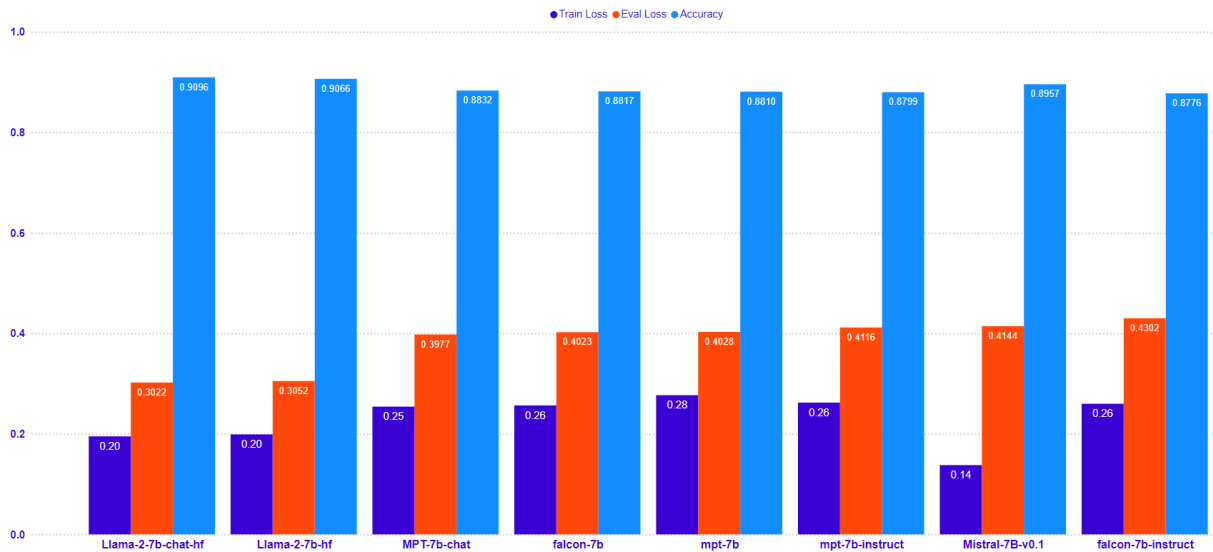


FIGURE 20. SF Performance Evaluation for LLM Models: The bar chart compares training loss, evaluation loss, and accuracy across various models such as Llama-2-7b-chat-hf, MPT-7b-chat, Falcon-7b, Mistral-7B-v0.1, and others. Accuracy (blue) is notably high for all models, while the training loss (purple) and evaluation loss (orange) exhibit variations, indicating differences in model generalisation capabilities during fine-tuning. Models like Llama-2-7b-chat-hf and Mistral-7B-v0.1 show lower evaluation loss, suggesting better performance in retaining learned information during supervised tasks.

Transformer Reinforcement Learning (TRL). These libraries are specialized for training large-scale models efficiently, focusing on resource optimization while retaining model performance [64]. The PEFT library, for example, allows for efficient fine-tuning with fewer trainable parameters, drastically reducing the computational burden while maintaining model performance [43]. Figure 23 shows an example of a QLoRA Fine-tuning configuration based on Huggingface Transformers, PEFT and TRL libraries. To ensure robust tracking and monitoring of our fine-tuning experiments, we utilized the Weights and Biases (W&B) platform. W&B allowed us to monitor essential metrics such as model accuracy, loss, GPU memory usage, and overall run-time efficiency [65].

V. RESULTS

A. SFT RESULTS

The following section presents the results of the SF conducted on various LLM base models, with a specific focus on performance metrics such as train loss, evaluation loss, and accuracy. The models analyzed include Llama2-7b, Llama2-7b-chat, Falcon-7b, Falcon-7b-instruct, MPT-7b, MPT-7b-chat, MPT-7b-instruct, and Mistral-7B-v0.1. The results are summarised in table 1 and visualized in Figure 20.

The evaluation metrics used in the SFT process provide insights into how well each model generalizes after fine-tuning. The training loss indicates the model's error on the training set, while the evaluation loss reflects the error on a separate validation set. Accuracy, as the proportion of correct predictions, serves as a key measure of model performance.

1) Comparative Analysis of Model Performance

Among the models tested, Llama2-7b-chat-hf demonstrated the highest accuracy of 0.9096, coupled with a low train loss

of 0.1951 and evaluation loss of 0.3022. Similarly, Llama2-7b-hf achieved comparable performance with an accuracy of 0.9066, train loss of 0.1989, and evaluation loss of 0.3052. These results suggest that the Llama2-7b variants are more robust in terms of both training and generalization capabilities when compared to other models, affirming their effectiveness in handling diverse language modeling tasks. This finding is consistent with prior research emphasizing the versatility of Llama-2 models in NLP tasks [10].

In contrast, the Falcon-7b-instruct model exhibited the lowest accuracy at 0.8776, paired with a relatively high train loss of 0.2599 and evaluation loss of 0.4302. This indicates that while the Falcon-7b models are effective, they may not generalize as well as the Llama2 variants. The MPT-7b models, such as MPT-7b-chat and MPT-7b-instruct, reported moderate accuracy scores of 0.8832 and 0.8799, respectively. These model's evaluation losses were also moderate, ranging from 0.3977 to 0.4116, suggesting balanced but not superior performance. The Mistral-7B-v0.1 model offers a unique performance that achieved a high accuracy of 0.8957 but showed a relatively high evaluation loss of 0.4144 despite having the lowest train loss of 0.1378. This could indicate over-fitting, where the model performs well on the training data but less on unseen validation data, aligning with findings from prior studies on the trade-offs in training deep neural networks [66].

2) Model Training and Generalisation

Overall, the models demonstrate a variety of trade-offs between train loss, evaluation loss, and accuracy. The Llama2 series, particularly the Llama2-7b-chat-hf model, shows superior generalization capabilities, as evidenced by their lower evaluation loss and higher accuracy. In contrast, the Falcon-7b series, despite achieving reasonable train losses, tends

TABLE 1. SF Results

S.No	Model Name	Learning Rate	Epochs	Duration	Train Loss	Eval Loss	Accuracy
1	Mistral-7B-v0.1	0.00005	5	28m 00s	0.1378	0.4144	0.8957
2	Llama2-7b-hf	0.00005	5	27m 49s	0.1989	0.3052	0.9066
3	Llama2-7b-chat-hf	0.00005	5	29m 54s	0.1951	0.3022	0.9096
4	Falcon-7b	0.00005	5	28m 42s	0.2566	0.4023	0.8817
5	Falcon-7b-instruct	0.00005	5	32m 10s	0.2599	0.4302	0.8776
6	MPT-7b	0.00005	5	30m 27s	0.2770	0.4028	0.8810
7	MPT-7b-chat	0.00005	5	28m 48s	0.2542	0.3977	0.8832
8	MPT-7b-instruct	0.00005	5	28m 52s	0.2622	0.4116	0.8799

to underperform in terms of evaluation loss and accuracy, indicating a possible limitation in generalization or overfitting issues. Meanwhile, the MPT-7b and Mistral-7B models occupy a middle ground, presenting balanced performances without distinctly excelling in any specific metric. For future work will consider exploring large datasets and state-of-the-art architectures to fine-tune these model's generalisation abilities further. Additionally, performing in-depth hyperparameter tuning, including adjusting learning rates, epoch sizes, batch size, and model architectures, could potentially yield further performance improvements [1].

B. LOW-RANK ADAPTATION (LORA) FINE-TUNING RESULTS

LoRA has emerged as a prominent technique for fine-tuning LLMs by reducing the number of trainable parameters while maintaining model performance. LoRA accomplishes this by adding low-rank matrices that adapt the weights of the model during fine-tuning. This approach not only reduces the computational cost but also allows for rapid training iterations, making it suitable for tasks that require frequent updates or domain-specific adaptation. This section presents a detailed comparative analysis of various models fine-tuned using LoRA, highlighting key insights into their training performance and generalization capabilities.

1) Comparative Analysis of Model Performance

The comparative analysis of model performance using LoRA is shown in Table 2 and illustrated in Figure 21. Thirteen different models were fine-tuned, with parameters such as learning rate, epochs, duration, training loss, and evaluation loss being consistently controlled across experiments. The models evaluated include variants from the Mistral, Llama, Phi, Gemma, and Zephyr families.

- 1) **Mistral Models:** The Mistral family models (Mistral-7b, Mistral-7b-instruct, Mistral-7b-instruct-v0.2, and Mistral-7b-instruct-v0.3) demonstrated varied performance in terms of both training loss and evaluation loss. The Mistral-7b-instruct-v0.2 model achieved a notable balance between train loss (0.4413) and evaluation loss (0.929), suggesting better generalization

capabilities compared to the original Mistral-7b model, which had a higher evaluation loss of 0.9187. However, the overall best performance within the Mistral series was achieved by the Mistral-7b-instruct-v0.3 variant, which registered a lower evaluation loss (0.9114) while maintaining a competitive train loss (0.4461). This indicates that additional instruct-tuning steps (v0-3) could lead to more refined model adjustments that enhance performance.

- 2) **Llama Models:** Llama models, specifically Llama3.1-8b and its instruct variant, Llama3.1-8b-instruct, exhibited slightly higher train and evaluation losses compared to the Mistral models. The instruct variant of Llama3.1-8b reported a train loss of 0.5432 and an evaluation loss of 0.94, indicating marginal improvements over the base model's evaluation loss of 0.9307. Despite these minor differences, both models demonstrated robust training stability over the five epochs, as reflected by the small difference between their train and evaluation losses. This consistency could be attributed to Llama's architecture which balances complexity and scalability, enhancing its adaptability to various fine-tuning regimes
- 3) **Phi and Gemma Models:** The Phi models (Phi-3.5-mini-4k-instruct and Phi-3-mini-4k-instruct) achieved the best performance in terms of both train loss and evaluation loss, with the Phi-3-mini-4k-instruct model having the lowest train loss (0.3274) and evaluation loss (0.6484) among all models tested. The significant reduction in losses observed in the Phi models indicates the effectiveness of their fine-tuning strategy, which could involve a better parameter initialization or a more effective use of the low-rank adaptation approach. On the other hand, Gemma models, especially Gemma2-9b and its instruct variant, also showed commendable performance, with evaluation losses as low as 0.8827. The instruct variant, Gemma2-9b-instruct, reported the lowest evaluation loss (0.8826) within the Gemma family, indicating that instruct-based fine-tuning leads to more generalized models.

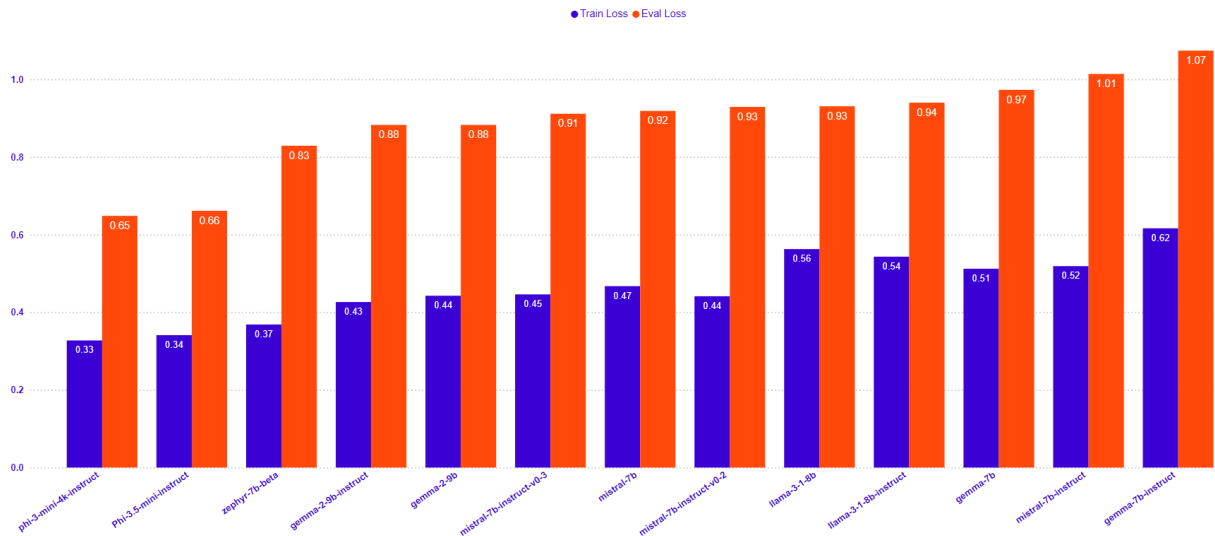


FIGURE 21. Low-Rank Adaptation (LoRA) Fine-tuning Results: The bar chart comparing the Train Loss and Eval Loss performance for LLMs after LoRA fine-tuning. The x-axis lists different models such as phi-3-mini-4k-instruct, zephyr-7b-beta, and various versions of Mistral, LLaMA, and Gemma models. The y-axis represents the loss values, with blue bars indicating Train Loss and orange bars showing Eval Loss. The chart demonstrates a trade-off between training performance and generalization across the different models

TABLE 2. Low-Rank Adaptation (LoRA) Fine-tuning Results.

S.No	Model Name	Learning Rate	Epochs	Duration	Train Loss	Eval Loss
1	Mistral-7b	0.0002	5	24m 36s	0.4673	0.9187
2	Mistral-7b-instruct	0.0002	5	26m 51s	0.5189	1.014
3	Mistral-7b-instruct-v0-2	0.0002	5	26m 14s	0.4413	0.929
4	Mistral-7b-instruct-v0-3	0.0002	5	25m 15s	0.4461	0.9114
5	Llama3.1-8b	0.0002	5	40m 12s	0.5628	0.9307
6	Llama3.1-8b-instruct	0.0002	5	39m 23s	0.5432	0.94
7	Phi-3.5-mini-4k-instruct	0.0002	5	28m 13s	0.3411	0.6614
8	Phi-3-mini-4k-instruct	0.0002	5	28m 32s	0.3274	0.6484
9	Gemma-7b	0.0002	5	46m 56s	0.5122	0.973
10	Gemma-7b-instruct	0.0002	5	48m 58s	0.6163	1.074
11	Gemma2-9b	0.0002	5	58m 53s	0.4426	0.8827
12	Gemma2-9b-instruct	0.0002	5	58m 37s	0.4264	0.8826
13	Zephyr-7b-beta	0.0002	5	47m 18s	0.3685	0.829

4) **Zephyr Models:** The Zephyr-7b-beta model displayed a competitive performance with a train loss of 0.3685 and an evaluation loss of 0.829. This model outperformed several other models, especially those with higher parameter counts, such as the Gemma2-9b, suggesting that the Zephyr model's architecture could be inherently more suited for low-rank adaptation-based fine-tuning.

2) Model Training and Generalization

In assessing model training and generalization, it is crucial to understand the trade-offs between train loss and evaluation loss. Models that exhibit significantly lower train losses

but higher evaluation losses, such as Gemma-7b-instruct, which reported a train loss of 0.6163 and an evaluation loss of 1.074, could be over-fitting to the training data. Conversely, models like Phi-3-mini-4k-instruct, with closely aligned train (0.3274) and evaluation losses (0.6484), demonstrate a stronger generalization capability, suggesting that their training is more robust to unseen data.

3) Training Duration and Efficiency:

The training duration is another critical factor affecting the practical utility of fine-tuning models. Models such as Mistral-7b, which completed training in 24 minutes and 36 seconds, contrast sharply with larger models like Gemma2-

```
# LoRA Finetuning - Ludwig Config

model_type: llm
base_model: meta-llama/Llama-2-7b-hf

adapter:
  type: lora

prompt:
  template: |
    ### Instruction:
    {instruction}

    ### Input:
    {input}

    ### Response:

input_features:
  - name: instruction
    type: text

output_features:
  - name: output
    type: text

trainer:
  type: finetune
  learning_rate: 0.0002
  batch_size: 1
  gradient_accumulation_steps: 8
  epochs: 3

backend:
  type: local
```

FIGURE 22. Example of LoRA Fine-tuning configuration based on Ludwig [62].

```
#QLoRA Finetuning Config

#Import libraries
from transformers import (AutoModelForCausalLM, AutoTokenizer,
BitsAndBytesConfig, HfArgumentParser, TrainingArguments, pipeline, logging)

from peft import (LoraConfig, PeftModel,
prepare_model_for_kbit_training, get_peft_model)

import os, torch, wandb
from datasets import load_dataset
from trl import SFTTrainer, setup_chat_format

# QLoRA config
bnb_config = BitsAndBytesConfig(load_in_4bit=True, bnb_4bit_quant_type="nf4",
bnb_4bit_compute_dtype=torch_dtype, bnb_4bit_use_double_quant=True)

# Load model
model = AutoModelForCausalLM.from_pretrained(base_model,
quantization_config=bnb_config, device_map="auto", attn_implementation="attn_implementation")

# Load tokenizer
tokenizer = AutoTokenizer.from_pretrained(base_model)
model, tokenizer = setup_chat_format(model, tokenizer)

# LoRA config
peft_config = LoraConfig(
    r=16,
    lora_alpha=32,
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM",
    target_modules=['k_proj', 'q_proj', 'v_proj', 'o_proj']
)
model = get_peft_model(model, peft_config)

#Finetuning Hyperparameters
training_arguments = TrainingArguments(
    output_dir=new_model,
    per_device_train_batch_size=1,
    per_device_eval_batch_size=1,
    gradient_accumulation_steps=8,
    optim="paged_adamw_32bit",
    num_train_epochs=2,
    evaluation_strategy="steps",
    eval_steps=100,
    logging_steps=1,
    warmup_steps=5,
    logging_strategy="steps",
    learning_rate=2e-4, #0.0002
    fp16=False,
    bf16=False,
    group_by_length=True,
    report_to="wandb",
    run_name = "Mistral-7B-Instruct-MET-qlora-run1",
    save_strategy="steps",
    save_steps=100,
    load_best_model_at_end=True,
)

trainer=SFTTrainer(model=model, train_dataset=dataset["train"], eval_dataset=dataset["test"],
peft_config=peft_config, max_seq_length=1024, dataset_text_field="text", tokenizer=tokenizer,
args=training_arguments, packing= False)

trainer.train()
```

FIGURE 23. Example of QLoRA Fine-tuning configuration

9b, which required approximately 58 minutes. Notably, despite their shorter training durations, models like Mistral-7b-instruct-v0.3 achieve competitive evaluation loss, indicating an efficient training regimen.

The comparative results of LoRA fine-tuning across multiple models reveal significant insights into the trade-offs between model complexity, training efficiency, and generalization capability. Phi models demonstrated superior performance, particularly Phi-3-mini-4k-instruct, which achieved the lowest evaluation loss, highlighting its effectiveness for domain-specific tasks. The results also underscore the potential benefits of further refining instruct-based fine-tuning techniques, particularly for models like Mistral and Gemma, which show varying improvements with different instruct iterations. Future research should explore optimizing LoRA configurations for other architectures to identify configurations that maximize both training efficiency and model performance.

C. QUANTIZED LOW-RANK ADAPTATION (QLORA) FINE-TUNING RESULTS

QLoRA has emerged as a promising technique for the efficient fine-tuning of large language models by combining quantization and low-rank adaptation to reduce computational overhead without sacrificing model performance. To evaluate the effectiveness of QLoRA, we conducted extensive fine-tuning experiments on several state-of-the-art language models, including Mistral, LLaMA, Phi, and Gemma. All fine-tuning was performed using Google Colab Pro with NVIDIA A10G GPUs equipped with 24GB RAM, leveraging Huggingface Transformers, PEFT [43], and TRL [64] libraries to optimize training processes.

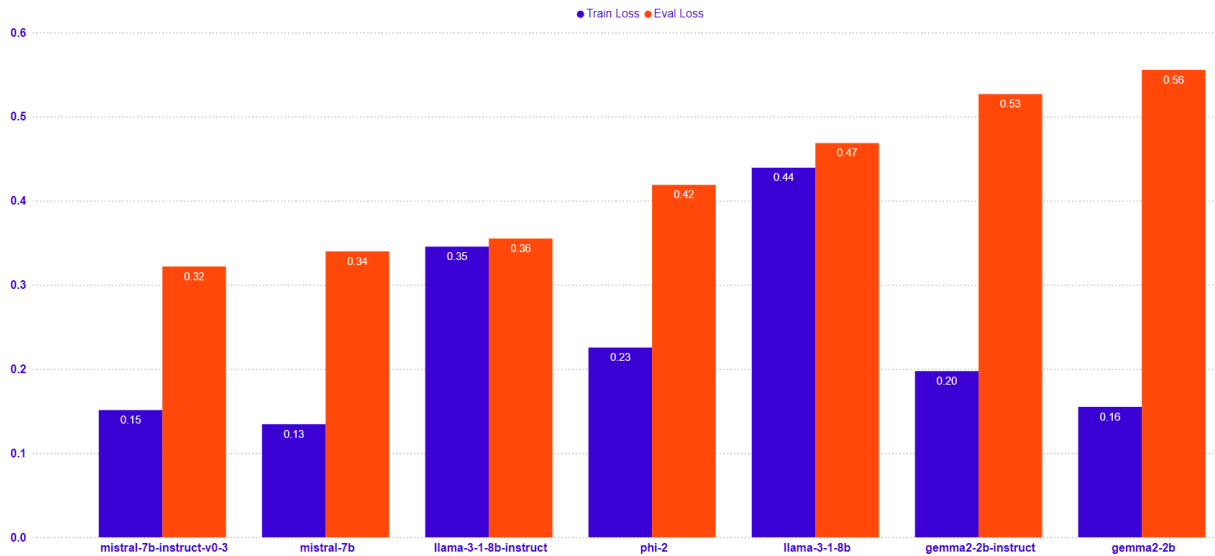


FIGURE 24. QLoRA Fine-tuning Results: The chart compares the Train Loss and Eval Loss performance for several LLMs after QLoRA fine-tuning. The x-axis lists different models, including versions of Mistral, LLaMA, Phi, and Gemma. The y-axis represents the loss values, with blue bars representing Train Loss and orange bars indicating Eval Loss. The chart highlights the variation in training efficiency and evaluation performance across models, showcasing a balance between optimization during training and generalization during evaluation.

TABLE 3. QLoRA Fine-tuning Results.

S.No	Model Name	Learning Rate	Epochs	Duration (min)	Train Loss	Eval Loss
1	Mistral-7b	0.0002	3	184	0.1344	0.3399
2	Mistral-7b-instruct-v0-3	0.0002	3	186	0.1512	0.3219
3	Llama-3-1-8b	0.0002	3	193	0.4393	0.4686
4	Llama-3-1-8b-instruct	0.0002	3	189	0.3455	0.3551
5	Phi-2	0.0002	3	82	0.2255	0.4189
6	Gemma2-2b	0.0002	3	136	0.1550	0.5556
7	Gemma2-2b-instruct	0.0002	3	137	0.1975	0.5268

1) Comparative Analysis of Model Performance

The fine-tuning results for the models, including their respective training times, training loss, and evaluation loss, are presented in Table 3. The training and evaluation loss metrics are crucial indicators of the model's ability to generalize from training data to unseen data. Lower loss values typically signify better model performance.

- **Mistral-7b and Mistral-7b-instruct-v0.3:** The Mistral-7b model demonstrated the lowest training loss (0.1344) among all models. Mistral-7b-instruct-v0.3 slightly increased the training loss (0.1512) but showed the best evaluation loss (0.3219). This indicates that both versions of the Mistral-7b model have strong generalization capabilities, with the instruct version being slightly more balanced in performance.
- **Llama-3.1-8b and Llama-3.1-8b-instruct:** The Llama-3.1-8b model exhibited a considerably higher training loss (0.4393) and evaluation loss (0.4686) compared to the other models, indicating that it may re-

quire more epochs or a lower learning rate for better convergence. However, its instruct version 'llama-3-1-8b-instruct' showed improvements with reduced training loss (0.3455) and evaluation loss (0.3551), highlighting the benefit of fine-tuning with instruction-based datasets.

- **Phi-2 Model:** Phi-2 model had a moderate training loss (0.2255) and evaluation loss (0.4189). Despite its relatively shorter training duration (82 minutes), it demonstrated competitive performance, suggesting it is a viable model for efficient training and deployment in low GPU resource-constrained environments.
- **Gemma2-2b and Gemma2-2b-instruct:** Both versions of the Gemma2-2b and Gemma2-2b-instruct showed higher evaluation losses (0.5556 and 0.5268, respectively) compared to other models, even though the training losses remained relatively low (0.1550 and 0.1975, respectively). This indicates potential over-fitting issues or a need for further hyper-parameter tuning to improve generalization capabilities.

2) Model Training and Generalization

The QLoRA fine-tuning experiments have revealed interesting insights into the generalization capabilities of different models:

- **Training Duration vs Model Performance:** The training duration varied significantly among the models, ranging from 82 minutes for ‘Phi-2’ to 193 minutes for ‘Llama-3.1-8b-instruct’. However, longer training times did not necessarily correlate with better evaluation performance. For instance, the ‘phi-2’ model, despite having a shorter training duration, outperformed ‘Llama-3.1-8b’ in evaluation loss.
- **Impact of Instruction-based Fine-tuning:** Models fine-tuned with instruction-based data (e.g., ‘Mistral-7b-instruct-v0-3’, ‘Llama-3.1-8b-instruct’, and ‘Gemma2-2b-instruct’) generally showed better generalization capabilities compared to their base versions. This aligns with findings from previous research, where instruction-based fine-tuning improves model robustness and adaptability to varied tasks [67].
- **Quantized Fine-tuning Efficiency:** The use of QLoRA in this setup has allowed us to train these large models more efficiently while maintaining comparable performance to their fully-trained counterparts. The benefits of such an approach are particularly evident in models like ‘Mistral-7b’ and ‘Phi-2’, where low training loss and competitive evaluation loss are achieved with shorter training durations and lower computational resources.

The results demonstrate that QLoRA fine-tuning, when implemented using libraries such as Huggingface Transformers, PEFT, and TRL, can lead to substantial savings in computational resources without a significant loss in model performance. The comparative analysis also suggests that the choice of model and fine-tuning strategy (e.g., instruction-based datasets) can significantly influence both training efficiency and generalization capabilities.

VI. DISCUSSION AND ANALYSIS

A. BENCHMARKING THE OPTIMAL LLM FINE-TUNING TECHNIQUE IN CYBERSECURITY AND IT SUPPORT

In the context of optimizing LLMs for specific tasks such as cybersecurity, cloud computing, and IT support, the efficiency and effectiveness of different fine-tuning techniques are of paramount importance. The experimental setup leveraging HPE Machine Learning Development Environment Software (HPE MLDES), Predibase platform with Ludwig and LoRAX, and Google Colab Pro with advanced libraries such as Huggingface Transformers and PEFT, provides a robust foundation for evaluating fine-tuning methods like SF, LoRA, and QLoRA. This section critically examines these techniques by comparing their computational efficiency, accuracy, response speed, and contextual understanding in specialized domains.

1) Supervised Fine-Tuning (SFT):

SF involves fine-tuning pre-trained LLMs on a labelled dataset to align them with domain-specific requirements. Using HPE’s AI cluster with 8 NVIDIA A100 40GB GPUs and GenAI Studio integrated into HPE MLDES, the SFT experiments demonstrated that models such as Llama2-7b and its variants achieved high accuracy and low train and evaluation losses. For example, the Llama2-7b-chat-hf model reported the highest accuracy of 0.9096, coupled with a low train loss of 0.1951 and evaluation loss of 0.3022, showcasing its strong generalization capabilities for language modelling tasks.

The scalability of the HPE MLDES environment, which supports prompt engineering and fine-tuning with Python SDK, ensures that SFT can be conducted efficiently for large-scale projects. This is critical for cybersecurity and IT support contexts where models must adapt to continuously evolving datasets. However, the relatively high computational demands of SFT, evident from training durations of up to 30 minutes per model, suggest that while SFT provides a robust performance baseline, it may not always be the most computationally efficient method, especially in environments with low-constrained resources.

2) Low-Rank Adaptation (LoRA):

LoRA provides a more computationally efficient alternative to SFT by reducing the number of trainable parameters while preserving model performance. Utilizing the Predibase platform, which leverages an open-source framework like Ludwig, LoRA fine-tuning was executed on systems equipped with NVIDIA A100 80GB PCIe GPUs, optimizing cost-efficiency and maintaining high throughput. The results demonstrated that models like Phi-3-mini-4k-instruct achieved the lowest train loss (0.3274) and evaluation loss (0.6484) among all tested models, outperforming larger models such as Llama3.1-8b in terms of both training efficiency and generalization capability.

The Ludwig framework simplifies the fine-tuning process by allowing developers to engage without deep coding expertise, making LoRA highly suitable for rapid deployment and adaptation in domains like cybersecurity, where new threats require frequent model updates. Additionally, the integration of LoRAX allows thousands of fine-tuned models to be deployed on a single GPU, further enhancing LoRA’s applicability in resource-constrained settings. However, models like Gemma-7b-instruct, which reported higher evaluation losses despite efficient training, highlight that achieving optimal performance with LoRA can depend on careful selection of model architectures and fine-tuning configurations.

3) Quantized Low-Rank Adaptation (QLoRA):

QLoRA combines quantization and low-rank adaptation to further reduce computational overhead while maintaining competitive model performance. Experiments conducted using Google Colab Pro with NVIDIA A100 GPUs, in conjunction with Huggingface Transformers, PEFT, and TRL

libraries, demonstrate that QLoRA can effectively balance efficiency and effectiveness. For instance, Mistral-7b-instruct-v0.3 achieved a low evaluation loss of 0.3219 with reduced training loss (0.1512), indicating strong generalization capabilities under low-constrained computational conditions. The PEFT library's ability to drastically reduce the number of trainable parameters while retaining performance further emphasizes QLoRA's potential for optimizing model deployment in environments with limited GPU resources.

The results indicate that LoRA and QLoRA fine-tuning techniques are particularly suited for cybersecurity and IT support applications, where large-scale processing, low GPU memory usage, balance in precision and real-time adaptability are crucial. Models like Phi-2, which achieved competitive evaluation losses (0.4189) with shorter training durations, underscore the potential of QLoRA in efficiently handling domain-specific tasks without significant GPU resources.

B. IDENTIFYING THE BEST OPEN-SOURCE LLM ARCHITECTURE FOR CYBERSECURITY AND IT SUPPORT

The selection of the most suitable open-source LLM architecture for fine-tuning in cybersecurity and IT support contexts involves evaluating various models based on their performance across SFT, LoRA, and QLoRA techniques.

Llama: Llama models, particularly Llama2-7b variants, have demonstrated robust performance across both SFT and QLoRA fine-tuning methods. The Llama2-7b-chat-hf model, with an accuracy of 0.9096 and low evaluation loss under SFT, and its instruct variants under QLoRA, highlight their adaptability and strong generalization capabilities. The scalability provided by HPE MLDES and GenAI Studio further enhances Llama models' suitability for specialized domains requiring high accuracy and contextual understanding.

Mistral: Mistral models, such as Mistral-7b-instruct-v0.3, show promising results in balancing computational efficiency with model performance, particularly under QLoRA fine-tuning. The model achieved the lowest evaluation loss (0.3219) with a competitive training loss, making it a strong candidate for environments requiring efficient deployment on limited hardware. The integration of Predibase's LoRAX with Ludwig also facilitates rapid model development and deployment, making Mistral models highly adaptable for real-time cybersecurity applications.

Phi: Phi models, such as Phi-3-mini-4k-instruct, demonstrated the best overall performance in LoRA fine-tuning experiments, achieving the lowest evaluation loss (0.6484). The models' effectiveness in adapting to domain-specific requirements with minimal computational resources makes them highly suitable for tasks involving real-time monitoring and rapid response, such as cybersecurity threat detection and IT support ticket resolution. The streamlined development environment provided by Predibase and Ludwig further supports the scalability and flexibility of Phi models [62].

Gemma: Gemma models, such as Gemma2-9b-instruct, performed well across both SFT and LoRA fine-tuning, with reasonable evaluation losses and efficient training times.

However, the variability in performance across different instruct variants suggests that additional optimization may be required to fully harness their potential in specialized domains. The adaptability of Gemma models, especially in instruct variants, makes them viable for incremental fine-tuning strategies, which could be beneficial in evolving cybersecurity and IT support domains.

VII. CONCLUSION

This study presents a thorough evaluation of three fine-tuning techniques—Supervised Fine-Tuning (SFT), Low-Rank Adaptation (LoRA), and Quantized LoRA (QLoRA) for optimizing LLMs in the domains of cybersecurity and IT support. The findings highlight several important considerations:

- **SFT** delivers high precision but at the cost of substantial computational overhead, particularly for models like Llama2-7B, making it less practical for resource-limited or production environments.
- **LoRA** maintains a balance by significantly reducing computational demands while maintaining performance close to that of SFT, as demonstrated by models like Phi-3-mini-4k-instruct and Zephyr-7B, which exhibit strong efficiency, accuracy, and inference capabilities.
- **QLoRA** offers the best trade-off between computational efficiency and model effectiveness, combining the advantages of quantization and low-rank adaptation. The Mistral-7b-instruct-v0.3 model illustrates its potential for real-world applications, where resource constraints and model effectiveness must be balanced.

These results underscore the importance of selecting the appropriate fine-tuning technique and model architecture when deploying LLMs in specialized fields like cybersecurity and IT support. Both LoRA and QLoRA emerge as the most feasible options for production environments, where computational efficiency and precision are paramount. Additionally, integrating tools such as HPE GenAI Studio, Ludwig, LoRAX, and Hugging Face Transformers into experimental setups enhances the flexibility and scalability of these fine-tuning approaches, making them more adaptable to the specific needs of real-world applications.

Future research will explore advanced fine-tuning techniques such as Direct Preference Optimization (DPO), Reinforcement Learning from Human Feedback (RLHF), and Odds Ratio Preference Optimization (ORPO) to further enhance model performance and adaptability.

REFERENCES

- [1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners, July 2020. arXiv:2005.14165 [cs].
- [2] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel

- Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open Pre-trained Transformer Language Models, June 2022. arXiv:2205.01068 [cs] version: 4.
- [3] Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. Memory-Efficient Fine-Tuning of Compressed Large Language Models via sub-4-bit Integer Quantization, October 2023. arXiv:2305.14152 [cs].
 - [4] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. Instruction Tuning for Large Language Models: A Survey, March 2024. arXiv:2308.10792 [cs].
 - [5] Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning, March 2023. arXiv:2303.15647 [cs].
 - [6] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, October 2021. arXiv:2106.09685 [cs].
 - [7] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs, May 2023. arXiv:2305.14314 [cs].
 - [8] Soroursadat Fatemi and Yuheng Hu. A Comparative Analysis of Fine-Tuned LLMs and Few-Shot Learning of LLMs for Financial Sentiment Analysis, December 2023. arXiv:2312.08725 [cs].
 - [9] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling Language Modeling with Pathways, October 2022. arXiv:2204.02311 [cs].
 - [10] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, February 2023. arXiv:2302.13971 [cs].
 - [11] Nick Mecklenburg, Yiyou Lin, Xiaoxiao Li, Daniel Holstein, Leonardo Nunes, Sara Malvar, Bruno Silva, Ranveer Chandra, Vijay Aski, Pavan Kumar Reddy Yannam, Tolga Aktas, and Todd Hendry. Injecting New Knowledge into Large Language Models via Supervised Fine-Tuning, April 2024. arXiv:2404.00213 [cs].
 - [12] Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. AdaMix: Mixture-of-Adaptations for Parameter-efficient Model Tuning, November 2022. arXiv:2205.12410 [cs].
 - [13] Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. QA-LoRA: Quantization-Aware Low-Rank Adaptation of Large Language Models, October 2023. arXiv:2309.14717 [cs].
 - [14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision, February 2021. arXiv:2103.00020 [cs].
 - [15] Imanol Schlag, Sainbayar Sukhbaatar, Asli Celikyilmaz, Wen-tau Yih, Jason Weston, Jürgen Schmidhuber, and Xian Li. Large Language Model Programs, May 2023. arXiv:2305.05364 [cs].
 - [16] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A Comprehensive Overview of Large Language Models, April 2024. arXiv:2307.06435 [cs] version: 9.
 - [17] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large Language Models: A Survey, February 2024. arXiv:2402.06196 [cs].
 - [18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, September 2013. arXiv:1301.3781 [cs].
 - [19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. Publisher: Nature Publishing Group.
 - [20] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. volume 14, pages 1532–1543, January 2014.
 - [21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019. arXiv:1810.04805 [cs].
 - [22] Christopher Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, May 1999. Google-Books-ID: YiFDxbEX3SUC.
 - [23] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997.
 - [24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, June 2017.
 - [25] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd edition, 2024.
 - [26] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units, June 2016. arXiv:1508.07909 [cs].
 - [27] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, April 2021. arXiv:2005.11401 [cs].
 - [28] Vladimir Karpukhin, Barlas Ögüz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense Passage Retrieval for Open-Domain Question Answering, September 2020. arXiv:2004.04906 [cs].
 - [29] Oded Ovadia, Menachem Brief, Moshik Mishaël, and Oren Elisha. Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs, January 2024. arXiv:2312.05934 [cs].
 - [30] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 739–753, May 2019. arXiv:1812.00910 [cs, stat].
 - [31] Jie Zhang, Haoyu Bu, Hui Wen, Yu Chen, Lun Li, and Hongsong Zhu. When LLMs Meet Cybersecurity: A Systematic Literature Review, May 2024. arXiv:2405.03644 [cs].
 - [32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
 - [33] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know What You Don't Know: Unanswerable Questions for SQuAD, June 2018. arXiv:1806.03822 [cs].
 - [34] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
 - [35] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Abubakr Babiker, Nathanael Schärli, Aakanksha Chowdhery, Philip Mansfield, Dina Demner-Fushman, Blaise Agüera y Arcas, Dale Webster, Greg S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, August 2023. Publisher: Nature Publishing Group.
 - [36] Elliot Bolton, Abhinav Venigalla, Michihiro Yasunaga, David Hall, Betty Xiong, Tony Lee, Roxana Daneshjoui, Jonathan Frankle, Percy Liang, Michael Carbin, and Christopher D. Manning. BioMedLM: A 2.7B Parameter Language Model Trained On Biomedical Text, March 2024. arXiv:2403.18421 [cs].
 - [37] Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. Large Language Models in Finance: A Survey, July 2024. arXiv:2311.10723 [cs, q-fin].
 - [38] Eirini Kalliamvakou. Research: quantifying GitHub Copilot's impact on developer productivity and happiness, September 2022.
 - [39] Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification, May 2018. arXiv:1801.06146 [cs, stat].

- [40] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D. Cubuk, and Quoc V. Le. Rethinking Pre-training and Self-training, November 2020. arXiv:2006.06882 [cs, stat].
- [41] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping, February 2020. arXiv:2002.06305 [cs].
- [42] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't Stop Pre-training: Adapt Language Models to Domains and Tasks, May 2020. arXiv:2004.10964 [cs].
- [43] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods, 2022. Publication Title: GitHub repository.
- [44] Drew A. Hudson and Christopher D. Manning. Compositional Attention Networks for Machine Reasoning, April 2018. arXiv:1803.03067 [cs].
- [45] En Li, Liekang Zeng, Zhi Zhou, and Xu Chen. Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing, October 2019. arXiv:1910.05316 [cs].
- [46] R OpenAI and others. GPT-4 technical report. *ArXiv*, 2303:08774, 2023.
- [47] Ensheng Shi, Yanlin Wang, Hongyu Zhang, Lun Du, Shi Han, Dongmei Zhang, and Hongbin Sun. Towards Efficient Fine-tuning of Pre-trained Code Models: An Experimental Study and Beyond, April 2023. arXiv:2304.05216 [cs].
- [48] Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer Learning in Natural Language Processing. In Anoop Sarkar and Michael Strube, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [49] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The Synthetic Data Vault. pages 399–410, October 2016.
- [50] Pluto. Pluto, August 2024. original-date: 2023-12-04T21:54:37Z.
- [51] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kavin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kudipudi, Anyana Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Mnyikwa, Suraj Nair, Avani Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the Opportunities and Risks of Foundation Models, July 2022. arXiv:2108.07258 [cs] version: 3.
- [52] Chunyuan Li, Zhe Gan, Zhengyuan Yang, Jianwei Yang, Linjie Li, Lijuan Wang, and Jianfeng Gao. Multimodal Foundation Models: From Specialists to General-Purpose Assistants, September 2023. arXiv:2309.10020 [cs].
- [53] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent Abilities of Large Language Models, October 2022. arXiv:2206.07682 [cs].
- [54] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M. Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesh Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask Prompted Training Enables Zero-Shot Task Generalization, March 2022. arXiv:2110.08207 [cs].
- [55] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Scao, Sylvain Gugger, and Alexander Rush. Transformers: State-of-the-Art Natural Language Processing, pages 38–45, January 2020.
- [56] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? . In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, pages 610–623, New York, NY, USA, March 2021. Association for Computing Machinery.
- [57] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Václav Cívek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. ETC: Encoding Long and Structured Inputs in Transformers, October 2020. arXiv:2004.08483 [cs, stat].
- [58] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluís-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon Emissions and Large Neural Network Training, April 2021. arXiv:2104.10350 [cs].
- [59] Emily M. Bender and Alexander Koller. Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online, July 2020. Association for Computational Linguistics.
- [60] Hewlett Packard Enterprise. Gen AI Studio, 2024.
- [61] Preditbase. Preditbase: The Developers Platform for Fine-tuning and Serving LLMs, 2024.
- [62] Piero Molino, Yaroslav Dudin, and Sai Sumanth Miryala. Ludwig: a type-based declarative deep learning toolbox, September 2019.
- [63] LoRAX. LoRAX, September 2024. original-date: 2023-10-20T18:19:49Z.
- [64] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. TRL: Transformer Reinforcement Learning, 2020. Publication Title: GitHub repository.
- [65] Lukas Biewald. Experiment Tracking with Weights and Biases, 2020.
- [66] Haidong Li, Jiongcheng Li, Xiaoming Guan, Binghao Liang, Yuting Lai, and Xinglong Luo. Research on Overfitting of Deep Learning. In *2019 15th International Conference on Computational Intelligence and Security (CIS)*, pages 78–81, December 2019.
- [67] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, September 2023. arXiv:1910.10683 [cs, stat].



CHAITHANYA VAMSHI SAI received a Master of Science (MSc) degree in Artificial Intelligence from Birmingham City University, Birmingham, United Kingdom, in 2023. He holds a Bachelor of Technology (BTech) degree in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad, India, in 2018.

His professional experience includes working as a Data Analyst at Cendyn, India, from 2018 to 2020. During this time, he led data integration and data analytics projects for clients in the USA, UK, Australia, and India. From 2020 to 2021, he served as a Business Intelligence Analyst at CBRE Group, India, where he developed KPI metric-driven dashboards for the project management department, enabling stakeholders to derive business insights and create innovative strategies.

Currently, he is a Data Scientist at Birmingham City University and Managed Enterprise Technologies Limited (METCLOUD) in the UK, as part of a Knowledge Transfer Partnership (KTP) project funded by Innovate UK Research and Innovation (UKRI) which is the United Kingdom's innovation agency. He is a Member of the British Computer Society (MBCS) Professional in the UK. His research interests include Artificial Intelligence, Generative AI, Explainable AI, Machine Learning, Data Analytics, Business Intelligence, and Cybersecurity.



DR IAIN RICE received a BSc Business and Applied Mathematics in 2012 and a PhD in Mathematics in 2015 from Aston University, Birmingham. He was a Postdoctoral Research Fellow contributing to the RAPID project in conjunction with Birmingham Children's Hospital creating algorithms for advanced clinical prediction of deterioration, following which he joined Arden University eventually becoming the Head of School Computing and IT in 2019. Since 2019 he has

worked at Birmingham City University as an Associate Professor leading AI teaching and research in the healthcare, finance and engineering domains.



DR NOUH ELMITWALLY is a PhD scholar specializing in Data Science. With extensive experience in both academia and industry, he has a strong background in Applied AI within medical imaging and Natural Language Processing (NLP). He earned his MSc in Computer Science from Cairo University in 2005 and completed his PhD in Computer Science at the University of Surrey, United Kingdom, in 2014.

Currently, he is a Lecturer with the School of Computing and Digital Technology, Birmingham City University, Birmingham, U.K., and the Faculty of Computers and Artificial Intelligence, Department of Computer Science, Cairo University, Egypt. His research interests include Data Science, Big Data, and Artificial Intelligence, with a particular focus on Machine Learning, Deep Learning, Large Language Models, and Generative AI. Elmitwally has conducted numerous research studies on cloud computing, NLP, and AI utilization for healthcare and has published his findings in prestigious international journals and conferences.



DR HAITHAM MAHMOUD received his PhD from Birmingham City University (BCU) in Electrical Engineering in 2022 and is a Marie-curie fellow on the IoT4win project at BCU between 2018 and 2021. His main research interests and expertise span a broad range of areas in intelligent networks and utilising AI on digital systems. As a research fellow at the Future Communication research cluster, he is currently actively coordinating research activities and overseeing major projects in BCU, including Intelligent warehouse tracking, intelligent B5G/6G radio access networks, radio access network resource optimisation, Cognitive radio networks, massive MIMO and future open networks. His research has been supported by H2020, UK DSIT, UKRI, NGI, BC and others in Egypt (STDF and NTI). He has been the Work Package Leader of H2020 IoT4win, Co-PI for BC TSNE with Egypt and academic supervisor for AKT.



IAN VICKERS In 2020, Ian Vickers launched the METCLOUD brand. The organisation has gone on to be recognised as one of the most secure cybersecurity platforms not only in the UK but has won dozens of internationally recognised awards for innovation. Through Ian's leadership the business has developed into a multi-million-pound organisation and is currently planning on International expansion in 2025. Ian is also keen to contribute to the community and has been Chair of the West Midlands Cyber Security Resilience Centre (WMCRC). The WMCRC is a Government backed initiative establishing ten regional Cyber Resilience Centres working in collaboration with NCSC, Police Cyber Teams (RCCU), Businesses and Academia.

Ian is an active member of the Computer Trade Industry Association (CompTIA) and in the UK, was a member of the Executive Council between 2015 and 2018 creating and supporting diversity initiatives. Ian is a member of the Microsoft Partner Advisory Council, which is made-up of twenty global Microsoft Partners to assist in the development of the Microsoft Partner program. In May 2020, Ian became a member of The World Economic Forum (Invitation only membership). The World Economic Forum is an organisation for Public-Private co-operation. The Forum engages the foremost political, business and other leaders of society to shape global, regional and industry agendas.



XAVIER SCHMOOR received an MSc in Information Systems and Computer Science in e-health at Ecole d'Ingénieurs ISIS in Castres, France, in 2017, and graduated in 2018 with a MSc in Innovation and Research in Informatics with Data Mining and Business Intelligence specialisation at the Universitat Politècnica de Catalunya, in Barcelona, Spain. Between 2018 and 2021, Xavier worked with Birmingham City University and Birmingham City Council on a series of data science

projects supporting businesses and local planning. Since 2021, Xavier has been working as a Data Scientist at METCLOUD on the development of AISOC, a tool speeding up the triage of cybersecurity alerts. His research interests include data science, cybersecurity, and business intelligence.

...