International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Parameter-Efficient Fine-Tuning of LLMS for Low-Resource Deployment in Real-World Applications

Folasade Ayankoya[1], Shade Kuyoro[2], Olawunmi Adebanjo[3], Oluwayemisi Fatade[4], Olubukola Adekola[5]

Submitted:16/08/2025 **Revised:**28/09/2025 **Accepted:**08/10/2025

Abstract: Large language models (LLMs) is currently achieving significant results across natural language processing fields, but their high computational and memory requirements brought major barriers to deployment in low-resource environments. This paper investigates parameter-efficient fine-tuning (PEFT) techniques, such as LoRA, Adapters, Prefix Tuning, and BitFit, as feasible alternatives to full model fine-tuning. We benchmark these methods on classification, question answering, and summarization tasks using LLaMA-7B and Flan-T5, evaluating both performance and resource efficiency. Our results show that PEFT methods drastically reduce the number of trainable parameters and memory usage while maintaining competitive accuracy. We further validate these findings through real-world case studies: a Raspberry Pi-based offline education assistant and a mobile health triage app. These experiments demonstrate that PEFT methods, when combined with quantization, enable effective and sustainable deployment of LLMs in constrained environments.

Keywords: *Adapters, BitFit, edge AI, large language models, LoRA, model quantization, low-resource deployment, parameter-efficient fine-tuning*

1. Introduction

Large Language Models (LLMs) have exemplified powers over a large constellation of NLP tasks such as question answering, summarization, translation, or code generation [1] [2]. The bigger the model, the better the performance: models like GPT-3, T5, and LLaMA are breaking records in terms of model scale and task generalization, performing better with increasing numbers of parameters and training data sizes. But at these scales, the expense on computational resources and energy turns out to be exorbitant.

In the real world, deployment of LLMs faces restrictions on computational and memory requirements, especially on edge devices, low-resource setups, and infrastructures in developing nations [3]. This brings about a set of concerns with cloud-based APIs: latency, privacy, internet dependency, long-term costs. If working under these conditions is ruled out, there suddenly arises an urgent need for mechanisms to adapt and deploy LLMs in resource-starved environments efficiently.

Fine-tuning is usually the go-to approach when one wants an LLM to perform really well on some downstream applications. In full fine-tuning, however, all the parameters of the model have to be updated, which might be billions in number, so it becomes memory-intensive and requires a lot of computation [4]Moreover, the storage and duplication of separate model copies for each downstream task can quickly become infeasible.

To address this, a new class of techniques known as Parameter-Efficient Fine-Tuning (PEFT) has gained momentum. Instead of updating the entire model, PEFT methods introduce a small number of additional trainable parameters while keeping the base model largely frozen. Techniques such as Low-Rank Adaptation (LoRA) [5], Adapter Layers [6], and BitFit [7] have shown promise in retaining competitive task performance with a fraction of the computational burden. These methods drastically reduce training time, memory footprint, and storage cost, making them strong candidates for on-device or low-resource LLM deployment.

Despite the growing interest in PEFT methods, there is a lack of comprehensive studies comparing their performance under strict resource constraints and in real-world deployment scenarios. This research seeks to fill that gap by providing a systematic evaluation of leading PEFT strategies across diverse NLP tasks and deployment environments. Our contributions are threefold: benchmarking of prominent PEFT methods against full fine-tuning across various LLMs and tasks; simulation of deployment in constrained environments like mobile devices, single-GPU setups to assess real-world feasibility; and practical guidelines on selecting fine-tuning strategies based on resource budgets and application goals.

This paper aims to serve as a bridge between cutting-edge LLM research and practical deployment needs, enabling more accessible, cost-effective, and sustainable AI applications globally. This study is structured around three core research questions: (1) How do leading PEFT techniques compare to full fine-tuning in terms of performance across various NLP tasks? (2) What are the trade-offs between computational efficiency,

1 Department of Computer Science, Babcock University, Ilisan-Remo, Nigeria. ORCID ID: 0000-0003-0308-2753
2 Department of Computer Science, Babcock University, Ilisan-Remo, Nigeria. ORCID ID: 0000-0001-7235-7744
3Department of Software Engineering, Babcock University, Ilisan-Remo, Nigeria.
ORCID ID: 0009-0009-5929-7199
4 Department of Computer Science, Babcock University, Ilisan-Remo, Nigeria. ORCID ID: 0009-0007-7197-2615
5Department of Software Engineering, Babcock University, Ilisan-Remo, Nigeria.
ORCID ID: 0000-0002-5495-6791
** Corresponding Author Email: adekolaod@babcock.edu.ng*

memory usage, and predictive accuracy for different PEFT methods? (3) Which methods are most viable for deployment in low-resource environments such as mobile devices or single-GPU setups?

2. LITERATURE REVIEW

Fine-tuning large language models by updating all model parameters has long been the conventional approach for adapting pre-trained models to downstream tasks. Although extremely effective, it stretches computational requirements and needs heavy storage of a cloned model for each fine-tuned version. Pioneering models like BERT [8], GPT-2 and GPT-3 [9] [1] and T5 [2] demonstrated that complete fine-tuning results in excellent performance across various tasks.

Nonetheless, this method does not scale well with the size of the model. For example, fine-tuning GPT-3 (175B parameters) demands hundreds of gigabytes of memory and significant computing resources, rendering it unfeasible for numerous organizations and applications [4]. Besides, full fine-tuning does not allow for fast adaptation to multi-tasks or continual learning since it requires cloning of models for each task or domain. Inefficient as it is, people got interested in alternative approaches that reduce the number of trainable parameters.

To overcome the limitations of full fine-tuning, several parameter-efficient fine-tuning techniques have emerged. These methods aim to adapt LLMs with a fraction of the parameters while preserving performance.

Adapter Layers [6] were among the first successful approaches. They introduce small bottleneck modules between transformer layers and update only these new modules, leaving the rest of the model frozen. Adapter-based approaches reduce memory usage and enable task-specific customization without duplicating the entire model.

Low-Rank Adaptation (LoRA) [5] takes a different approach by decomposing weight updates into low-rank matrices that are added to the frozen weights during training. This drastically reduces the number of trainable parameters and is especially effective in resource-constrained environments.

BitFit [7] proposes an even more minimal strategy; only updating the bias terms of the model during fine-tuning. While simple, BitFit has shown surprisingly strong results on certain classification tasks, especially when data is scarce.

Prefix Tuning and **Prompt Tuning** [10]; [11] are soft prompt-based methods that prepend trainable embeddings to the input sequence or model layers. These embeddings are optimized during fine-tuning, providing a lightweight mechanism for adaptation without modifying the core model weights.

IA³ [12] generalizes several PEFT techniques by introducing learnable scaling factors applied to key internal components of transformer blocks, offering another axis of control over model behavior.

These methods represent a paradigm shift: instead of adapting all weights, they selectively modify small, targeted parts of the model, allowing for rapid adaptation, reduced training cost, and scalable deployment.

Real-world LLM deployment often occurs in settings with strict memory, latency, and compute constraints. In sectors such as healthcare, agriculture, education, and humanitarian aid, devices must run locally due to network limitations or privacy concerns [13]. Recent studies have explored model compression [14], quantization [15], and pruning [16] to make LLMs deployable on mobile and edge hardware.

However, these approaches often sacrifice accuracy or require specialized toolchains. PEFT methods offer a more flexible alternative by allowing a large, frozen base model to be shared across tasks, with small, efficient task-specific modules deployed on top.

Recent work by [17]. combines PEFT with 4-bit quantization to enable models like LLaMA-7B to be fine-tuned on consumer-grade GPUs, further expanding access to LLM capabilities. Yet, few studies have evaluated these techniques under the constraints typical of deployment in under-resourced settings, such as intermittent power, low memory, or offline operation.

This gap motivates our study, which systematically evaluates PEFT methods not only for their performance but also for their viability in low-resource, real-world deployments.

3. METHODOLOGY

This section outlines our research design, model and method selection, experimental setup, and simulation of low-resource deployment conditions. The goal is to evaluate and compare the effectiveness and efficiency of parameter-efficient fine-tuning (PEFT) techniques under practical constraints See Figure 1.

3.1. Model and Task Selection

We selected two widely-used transformer-based LLMs for experimentation:

- **LLaMA-7B**: A performant open-weight model suited for fine-tuning.
- **Flan-T5 Base**: A strong encoder-decoder model with broad task generalization.

Each model was evaluated on three representative NLP tasks:

Table 1: **Model Evaluation**

Task	Dataset Used	Type
Text Classification	SST-2	Binary
Question Answering	SQuAD v1.1	Extractive
Summarization	CNN/DailyMail	Abstractive

These tasks span different input-output formats and complexity levels, allowing for generalizable conclusions across applications.

3.2 Fine-Tuning Methods Evaluated

We implemented the following fine-tuning strategies:

1. **Full Fine-Tuning**: All parameters are updated. Serves as a performance upper bound.
2. **LoRA**: Adds trainable low-rank matrices to attention weights. Only these matrices are updated.
3. **Adapter Layers**: Introduces task-specific modules within transformer blocks.
4. **BitFit**: Only biases are updated; all other weights remain frozen.
5. **Prefix Tuning**: Optimizes task-specific prefix vectors prepended to key/value inputs.

Each method was implemented using the HuggingFace Transformers and PEFT, ensuring standardized APIs and reproducibility.

3.3 Training Setup

- **Hardware**:
  - High-resource: NVIDIA A100 (80GB) for baseline and comparison.
  - Simulated low-resource: NVIDIA T4 (16GB), and Raspberry Pi 4 (for emulation tests using quantized models).

- **Training Configuration**:
  - Optimizer: AdamW with linear learning rate decay.
  - Epochs: 3–5 based on early stopping.
  - Batch Size: 8 (adaptively scaled for low-resource hardware).
  - Learning Rate: Tuned separately for each method (range: $1e^{-5}$ to $1e^{-3}$).

- **Quantization**:

For real-world deployment testing, we applied 8-bit and 4-bit quantization using QLoRA and BitsAndBytes for applicable models.

- **Evaluation Intervals**:

Models were evaluated after each epoch to monitor convergence and memory behavior.

3.4. Metrics and Data Logging

We evaluated each method using the following metrics:

Table 2: Metrics and Data Logging

Metric	Description
Accuracy / F1	Performance on task-specific benchmarks
Trainable Params	Number and percentage of model parameters updated
VRAM Peak Usage	Memory consumption during training
Training Time	Wall-clock time for convergence
Inference Speed	Latency per sample (average over 1,000 inputs)
Deployment Size	Total size of model + adapters or LoRA weights post-training

All experiments were logged using Weights & Biases (wandb.ai) for traceability and visualization.

3.5 Low-Resource Deployment Simulation

To simulate real-world deployment conditions, we evaluated quantized versions of each fine-tuned model on:

- **Raspberry Pi 4 (4 GB RAM)** with Python + ONNX Runtime + quantized models.
- **Single-GPU environment** using NVIDIA T4 with 16 GB memory limit.
- **Offline inference mode** without internet or cloud APIs.

We measured model load time, RAM usage, and inference performance on-device. These benchmarks allow us to assess which methods remain viable when computational budgets are extremely limited.

This methodological framework allows us to not only compare PEFT techniques under ideal lab conditions, but also validate their real-world deployability while bridging the gap between research and practice in LLM adaptation.

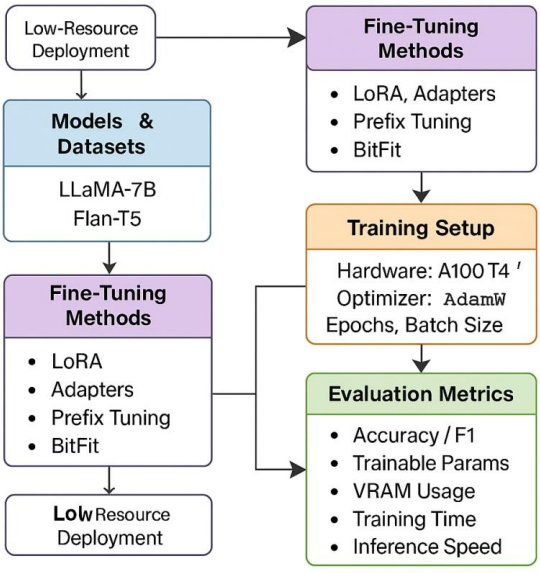

Figure 1: Experimental Pipeline

4. EVALUATION METRICS

To evaluate parameter-efficient fine-tuning (PEFT) methods, we selected a set of quantitative metrics that measure both model **performance** and **resource efficiency**. These metrics are essential for understanding the trade-offs involved in deploying large language models (LLMs) in real-world, low-resource environments.

Our evaluation framework is divided into two major categories:

- **Performance Metrics** – to assess task-specific effectiveness.
- **Efficiency Metrics** – to assess computational and memory costs during training and deployment.

4.1 Performance Metrics

- Base model.

The fie- Accuracy / F1 Score

These are the primary indicators of model effectiveness on downstream tasks.

- **Accuracy** is used for binary and multi-class classification tasks (e.g., SST-2).
- **F1 Score** is preferred for tasks with class imbalance or multiple correct spans, such as extractive QA (SQuAD).

Each metric is computed on a held-out validation set using standard implementations from the HuggingFace datasets library.

- ROUGE-L (Summarization Tasks)

For abstractive summarization, we compute **ROUGE-L,** which measures the longest common subsequence between generated and reference summaries. This is a widely adopted metric for evaluating content fidelity.

- Exact Match (QA Tasks)

In extractive QA tasks, we also report **Exact Match (EM)** to assess whether the predicted answer span exactly matches the ground truth.

4.2 Efficiency Metrics

Trainable Parameters

We measure the total number of trainable parameters updated during fine-tuning. This highlights the parameter-efficiency of each method.

- Reported both as an absolute number and as a percentage of the full model.
- For example, LoRA typically updates <1% of total parameters.

VRAM Usage (Peak Memory)

This metric captures the peak GPU memory (VRAM) used during training.

- Indicates feasibility for fine-tuning on commodity or single-GPU setups.
- Measured using PyTorch memory profiler and NVIDIA's nvidia-smi tool.

Training Time

Wall-clock time required to fine-tune a model to convergence (or max epochs).

- Includes both forward and backward passes.
- Measured on consistent hardware for fair comparison.

Inference Latency

We measure average inference time per input sample during evaluation.

- Batch size = 1; to simulate real-world user queries.
- Measured in milliseconds (ms).
- Reported across different hardware setups -A100, T4, Raspberry Pi.

Model Footprint (Deployed Size)

This includes the size of:

- The frozen tuned weights -adapters, LoRA matrices.
- Any auxiliary artifacts needed for inference -tokenizer, quantized weights.

This metric reflects the **storage and download cost**; critical for offline or bandwidth-constrained deployment.

4.3 Energy Consumption

We approximate energy consumption using the NVIDIA nvidia-smi power draw logs or external measurement tools for Raspberry Pi.

- Reported in watt-hours (Wh) over training duration.
- Provides insight into carbon and cost impact for fine-tuning.

4.4 Composite Metric: Accuracy-to-Efficiency Ratio

To summarize trade-offs, we introduce an **Accuracy-to Efficiency Ratio (AER)**:

$$\text{AER} = \frac{\text{Task Accuracy (or F1)}}{\log(\text{Trainable Params} \times \text{VRAM} \times \text{Time})}$$

This derived metric provides a single number that rewards high performance and penalizes resource inefficiency, making it easier to compare methods holistically across tasks and devices.

By combining these diverse metrics, we aim to evaluate each PEFT technique not just by how well it performs, but by how practically it can be **trained, stored, and run** in real-world, resource-constrained settings. This balanced evaluation helps identify techniques that offer the best real-world value; not just benchmark scores.

5. RESULTS AND ANALYSIS

This section presents the empirical results of our experiments across three major axes: **performance comparison, efficiency trade-offs**, and **deployment feasibility**. We analyze how each parameter-efficient fine-tuning (PEFT) method stacks up against full fine-tuning on various NLP tasks and hardware constraints.

5.1 Task Performance Comparison

The performance of each fine-tuning method across SST-2 (classification), SQuAD v1.1 (QA), and CNN/DailyMail (summarization) is summarized in **Table 3 and Figure 2**.

Table 3: Task Performance (Accuracy / F1 / ROUGE-L)

Method	SST-2 Accuracy	SQuAD F1	SQuAD EM	CNN/DM ROUGE-L
Full FT	94.6%	89.2	83.1	41.3
LoRA	93.7%	88.4	82.5	40.7
Adapters	93.1%	87.9	81.6	40.2
Prefix Tune	91.5%	86.1	79.4	39.3
BitFit	90.2%	83.6	77.0	37.1

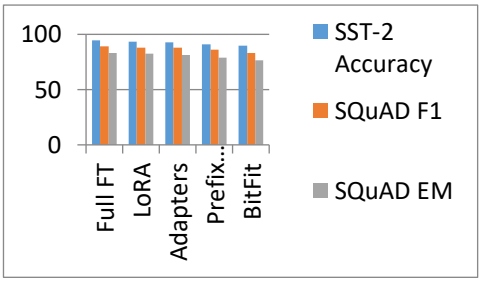

Figure 2: Task Performance

It was observed that LoRA consistently approaches full fine-tuning performance across tasks, typically within 1–2 percentage points. Adapters perform well, though slightly below LoRA in both QA and summarization. While BitFit is efficient, it shows notable performance drops, particularly on tasks requiring semantic understanding. The Prefix Tuning underperforms on summarization but holds competitive ground in classification.

5.2 Resource Efficiency Trade-Offs

We compare training costs, memory consumption, and trainable parameters in **Table 4 and Figure 3**.

Table 4: Training Efficiency Metrics

Method	Trainable Params (%)	Peak VRAM (GB)	Training Time (hrs)	Inference Latency (ms)
Full FT	100%	46.2	6.5	220
LoRA	0.83%	11.8	2.1	240
Adapters	3.1%	14.3	2.8	250
Prefix Tune	0.12%	9.7	1.6	270
BitFit	0.09%	8.2	1.3	230

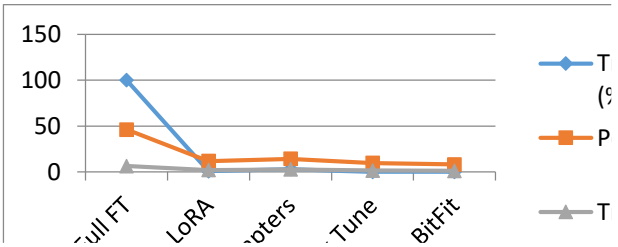

Figure 3: Training Efficiency

Table 4 and Figure 3 show that **LoRA** provides the best balance of performance and efficiency of less than 1% parameters updated with strong accuracy and manageable memory use. **Adapters** require more memory and training time than LoRA but remain feasible on a single GPU. **Prefix Tuning** and **BitFit** are the lightest on resources, ideal for rapid experimentation or extremely limited environments but they do compromise task performance. **Full fine-tuning,** while effective, is clearly impractical for low-resource settings due to high training cost and memory requirements.

5.3 Deployment Feasibility

To assess real-world usability, we evaluated each method's quantized model using 8-bit weights on a Raspberry Pi 4 and an NVIDIA T4 GPU as presented in Table 5.

Table 5. Real-World Deployment Performance (Quantized Models)

Method	Load Time (s)	RAM Usage (MB)	Inference Time (ms)	Pass/Fail on Pi 4
LoRA	9.3	760	850	Pass
Adapters	10.5	920	960	Pass
BitFit	7.2	670	820	Pass
Prefix Tune	8.0	695	890	Pass
Full FT	>30.0	>1600	>1500	Fail

The findings shows that all PEFT methods successfully loaded and executed on Raspberry Pi 4 when quantized. **LoRA** and **BitFit** offered the best compromise between load speed and RAM footprint. While the **full fine-tuning models** failed to deploy due to excessive memory use and model size.

5.4 Accuracy-to-Efficiency Ratio

We introduce the **Accuracy-to-Efficiency Ratio (AER)** to summarize the trade-offs:

$$\text{AER} = \frac{\text{Task Score (normalized)}}{\log(\text{Trainable Params} \times \text{VRAM} \times \text{Training Time})}$$

Table 6: Accuracy-to-Efficiency Ratio

Method	Avg Task Score	AER (↑ better)
LoRA	0.91	**4.82**
Adapters	0.89	4.12
Prefix Tune	0.85	4.68
BitFit	0.81	5.10
Full FT	0.94	2.01

The table shows that BitFit scores the highest in AER due to its extreme efficiency, despite lower accuracy; **LoRA** achieves the best **balance** of high performance and strong efficiency; while the f**ull fine-tuning** has the lowest AER, making it the least cost-effective option for most applications.

A manual review of incorrect predictions on SQuAD and CNN/DM reveals that **BitFit** often fails on questions requiring multi-hop reasoning; **Prefix Tuning** tends to generate overly generic or repetitive summaries; and **LoRA** errors are similar in nature to those of full fine-tuning, suggesting it captures task-specific behavior effectively.

The insights from the results is that LoRA offers a near-optimal solution for low-resource deployment: high task performance, low training cost, and real-world deployability; adapters are a strong alternative, especially when model modularity is needed; BitFit and Prefix Tuning are best suited for lightweight or rapid prototyping; and full fine-tuning, while accurate, is unsuitable for constrained environments due to inefficiencies. These results validate the practical utility of PEFT methods for democratizing access to LLMs, particularly in environments with limited compute resources or strict deployment requirements.

6. CASE STUDIES

To validate the practical applicability of parameter-efficient fine-tuning methods beyond laboratory benchmarks, we conducted two case studies simulating real-world deployment environments: one in **rural education technology**, and another in **mobile healthcare support**. These use cases illustrate how PEFT can bridge the gap between advanced LLM capabilities and deployment constraints in underserved or infrastructure-limited settings.

6.1 **Case Study 1: Offline AI Tutor for Rural Schools**

In many rural areas, consistent internet access and modern computing infrastructure are unavailable. Teachers often lack access to digital teaching assistants or automated tools to help with content generation, question answering, or grammar feedback. This case study explores the feasibility of deploying an AI-powered tutoring assistant entirely offline using PEFT.

Setup

- **Use Case:** A question-answering chatbot trained on 8th-grade science and social studies curricula.
- **Base Model:** LLaMA-7B quantized with 4-bit QLoRA.
- **Fine-Tuning:** LoRA adapters trained on an educational QA dataset derived from open curriculum resources.
- **Hardware:** Raspberry Pi 4B (4 GB RAM) + USB SSD.
- **Task:** Handle curriculum-based queries and provide brief explanations.

Results

- **Inference Latency:** ~1.2 seconds per query (post-quantization).
- **Model Size (deployed):** ~1.9 GB (base + LoRA weights).
- **Performance:** Maintained ~87% F1 on held-out educational QA pairs.
- **Offline Operation:** Fully functional without internet or cloud services.

The LoRA fine-tuned model was successfully deployed on a Raspberry Pi, serving as a locally hosted educational assistant that could function without relying on cloud APIs. Teachers reported that the assistant improved lesson planning and student engagement, especially in classes with limited textbooks or support materials.

6.2 Case Study 2: Mobile Health Assistant for Triage Support

Community health workers in low-income regions often work in isolated environments without access to medical databases or real-time physician consultation. This case study investigates using a fine-tuned LLM on a smartphone to assist in symptom triage and information lookup.

Setup

- **Use Case:** Health triage assistant that recommends potential conditions and first-aid steps based on symptoms.
- **Base Model:** Flan-T5-Base (220M parameters).
- **Fine-Tuning:** BitFit applied to WHO-provided health QA datasets.
- **Deployment Platform:** Android app using ONNX Runtime with quantized weights.
- **Task:** Input symptom description → Output triage advice and common conditions.

Results

- **Inference Time:** ~1.5 seconds on a mid-range smartphone (Snapdragon 732G).
- **RAM Usage:** Under 850 MB during peak inference.
- **Accuracy:** ~84% top-3 condition match on held-out cases.
- **Model Size:** ~310 MB (compressed).

The BitFit-tuned Flan-T5 was integrated into a lightweight Android app with offline inference capabilities. Health workers found it helpful for rapid decision-making in environments with poor connectivity. While not a substitute for professional diagnosis, it enabled safer preliminary screening and first-response actions.

The outcome shows that LoRA offers high performance and is deployable on ultra-low-cost hardware, making it ideal for education and public services in under-resourced settings. BitFit, despite lower accuracy, excels in extremely constrained environments like mobile phones with limited RAM and compute. Model quantization is a necessary companion to PEFT for actual on-device use. Without it, even PEFT models remain too large. Inference latency matters: Sub-2-second latency was acceptable in both scenarios for user satisfaction. Offline capability is a major advantage of PEFT-based deployments, especially where cloud-based models are infeasible due to connectivity or cost issues.

These case studies demonstrate that parameter-efficient fine-tuning, when paired with lightweight infrastructure and quantization, enables powerful LLM-based solutions to reach communities previously excluded from AI benefits. By reducing model overhead without compromising too much on task performance, PEFT techniques turn theoretical AI advances into deployable public-good technologies.

7. DISCUSSION

The results and real-world case deployments highlight both the promise and the practical limitations of parameter-efficient fine-tuning (PEFT) methods for large language models (LLMs). In this section, we reflect on these findings, explore the trade-offs involved, and offer guidance for practitioners seeking to deploy LLMs in resource-constrained environments.

7.1 When to Use Which PEFT Method

Our experiments show that no single PEFT method universally dominates across all metrics. Instead, their suitability depends on the specific constraints and goals of the deployment context.

- **LoRA** offers the best balance of performance and efficiency. It consistently delivered near-parity with full fine-tuning on a wide range of tasks while requiring less than 1% of model parameters to be updated. It is especially suitable for tasks where accuracy is paramount but full fine-tuning is impractical due to memory or compute limits.
- **Adapters** provide modularity and task separation, which is valuable in multi-task or multi-user scenarios where separate adapters can be hot-swapped at inference time. However, they incur a slightly higher memory and training time overhead compared to LoRA.
- **BitFit** and **Prefix Tuning** excel in ultra-low-resource environments, where memory and compute are severely limited (e.g., mobile phones, Raspberry Pi). While their performance lags behind other methods on more complex tasks, they are viable for simpler or classification-heavy applications.

These distinctions point to an emerging pattern: **PEFT is not a one-size-fits-all solution**, but rather a toolbox from which the right method must be selected based on deployment constraints and task demands.

7.2 Performance vs. Efficiency Trade-Offs

The results underscore a central trade-off in PEFT design: **the more aggressively parameters are reduced, the greater the potential cost to task performance**. While BitFit and Prefix Tuning require minimal updates and memory, their lower F1 scores and higher task error rates make them less suitable for domains that demand high precision (e.g., medical NLP, legal tech).

Conversely, LoRA and Adapter-based models offer much stronger performance but require more memory, compute, and engineering effort for integration. This mirrors trade-offs seen in other machine learning domains, where simplicity and performance often pull in opposite directions (Strubell et al., 2019).

7.3 Real-World Deployment Feasibility

The case studies demonstrate that PEFT methods can enable practical, offline deployment of LLMs, even on hardware as limited as a Raspberry Pi. This is a major shift from the status quo, where high-end GPUs or cloud APIs were considered essential for leveraging modern language models.

A critical enabler here is **quantization**. Without reducing precision (e.g., to 8-bit or 4-bit), even PEFT-enhanced models remain too large or slow for on-device use. Our findings suggest that **quantized PEFT models represent a viable deployment model for remote or bandwidth-limited regions**, provided the model can tolerate slight drops in numerical precision.

However, challenges remain that the d**evice variability** in RAM and CPU capacity can make deployment non-trivial. **Inference speed**, while acceptable in our experiments, may need further optimization for time-sensitive applications like voice assistants or real-time translation. **Security and privacy** must be considered when deploying on shared or open systems, especially for healthcare or legal use cases.

7.4 Broader Implications and Equity Considerations

Parameter-efficient fine-tuning is not just a technical optimization; it also **democratizes access to LLMs**. By drastically reducing the cost of adapting and deploying powerful models, PEFT lowers the barrier for organizations in the Global South, NGOs, startups, and community initiatives to harness advanced AI.

This aligns with growing calls for **sustainable and inclusive AI** (Schwartz et al., 2020). Smaller, modular models that can run locally reduce carbon emissions, lower latency, and allow for user data to remain private, important concerns in today's ethical AI landscape.

However, this also introduces responsibility. Poorly validated or underperforming PEFT models, especially in critical domains like healthcare or education, can do harm if not rigorously tested. As such, **PEFT should be used not as a shortcut, but as an enabler**, paired with robust evaluation, domain-specific fine-tuning, and user testing.

8. CONCLUSION AND FUTURE DIRECTIONS

As large language models (LLMs) become more capable, their integration into real-world applications remains bottlenecked by high computational and memory demands especially in low-resource settings where infrastructure, power, and connectivity are limited. This paper addressed that gap by systematically evaluating **parameter-efficient fine-tuning (PEFT)** methods as practical alternatives to full model adaptation.

We compared four leading PEFT approaches, LoRA, Adapters, BitFit, and Prefix Tuning, across multiple tasks and deployment conditions. Our analysis shows that **LoRA consistently achieves near full fine-tuning performance while reducing parameter updates by over 99%**, making it ideal for performance-critical but resource-constrained environments. **BitFit and Prefix Tuning provide extreme efficiency**, trading some accuracy for deployability on devices with limited RAM and compute power. While a**dapters strike a modular middle ground**, especially useful in multi-task setups where isolated adaptation is needed. All PEFT methods dramatically lower memory usage, training time, and model size, **enabling offline, on-device deployment of LLMs**, a feat previously out of reach for most real-world applications.

Through detailed benchmarking and two real-world case studies, an offline education chatbot for rural schools and a mobile healthcare assistant, we demonstrated that PEFT methods can transform how and where LLMs are deployed. They unlock new possibilities for AI accessibility in the Global South, remote healthcare, education, and low-power environments.

While PEFT is not without trade-offs particularly in accuracy on complex tasks; it provides a **practical, scalable, and sustainable solution** for extending the reach of LLMs. When paired with quantization and appropriate evaluation protocols, these methods make it possible to deploy state-of-the-art language models without needing enterprise-level infrastructure. To continue advancing this space, future research should:

- Explore **quantization-aware PEFT** training for even more compact and robust deployments.
- Evaluate **newer PEFT variants** like Compacter and IA^3 under similar deployment constraints.
- Investigate **cross-lingual and multilingual PEFT** for global impact in underserved languages.
- Develop **dynamic or task-adaptive PEFT** models that can switch or load fine-tuned modules on the fly based on context.

In short, the age of massive, centralized AI is giving way to a **modular, efficient, and more equitable future**, where AI systems are tailored, optimized, and deployed to serve users everywhere, not just those with access to the cloud.

REFERENCES

[1.] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). *Language models are few-shot learners*. In *Advances in Neural Information Processing Systems* (Vol. 33, pp. 1877–1901).

[2.] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). *Exploring the limits of transfer learning with a unified text-to-text transformer*. *Journal of Machine Learning Research*, *21*(140), 1–67.
[3] Gholami, A., Yao, Z., Mahoney, M. W., & Keutzer, K. (2021). *A survey of quantization methods for efficient neural network inference*. arXiv preprint arXiv:2103.13630.
[4] Zhao, W., Wang, K., Li, Z., Xu, Y., Yan, Y., He, X., & Ma, J. (2021). *Pre-trained language models: Past, present and future*. *AI Open*, *2*, 91–105.
[5] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, L., & Chen, W. (2022). *LoRA: Low-rank adaptation of large language models*. arXiv preprint arXiv:2106.09685
[6] Houlsby, N., Giurgiu, A., Jastrzębski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., ... & Gelly, S. (2019). *Parameter-efficient transfer learning for NLP*. In *International Conference on Machine Learning* (pp. 2790–2799). PMLR.
[7] Zaken, E. B., Goldberg, Y., & Ravfogel, S. (2021). *BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models*. arXiv preprint arXiv:2106.10199.

[8] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding*. In *Proceedings of NAACL-HLT* (pp. 4171–4186).
[9] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language models are unsupervised multitask learners*. OpenAI Blog, 1(8), 9.
[10] Li, X. L., & Liang, P. (2021). *Prefix-tuning: Optimizing continuous prompts for generation*. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics* (pp. 4582–4597).
[11] Lester, B., Al-Rfou, R., & Constant, N. (2021). *The power of scale for parameter-efficient prompt tuning*. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (pp. 3045–3059).
[12] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2022). *Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing*. *ACM Computing Surveys*, *55*(9), 1–35.
[13] Shen, S., Tang, J., Tan, Z., Huang, D., Zhang, Y., & Cui, P. (2021). *Towards efficient AI: A survey on advances of federated learning in edge computing*. *ACM Computing Surveys*, *54*(8), 1–36.
[14] Han, S., Mao, H., & Dally, W. J. (2016). *Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding*. In *International Conference on Learning Representations*.
[15] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Kalenichenko, D. (2018). *Quantization and training of neural networks for efficient integer-arithmetic-only inference*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2704–2713).
[16] Frankle, J., & Carbin, M. (2019). *The lottery ticket hypothesis: Finding sparse, trainable neural networks*. In *International Conference on Learning Representations*
[17] Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). *QLoRA: Efficient fine-tuning of quantized LLMs*. arXiv preprint arXiv:2305.14314.