Surya Pratap

RE-2022-551421

Batch 5

Batch 5

Universidad del Valle

Document Details

Submission ID

trn:oid:::26066:452113228

Submission Date

Apr 24, 2025, 7:30 PM GMT+5:30

Download Date

Apr 24, 2025, 7:31 PM GMT+5:30

File Name

RE-2022-551421.pdf

File Size

1.0 MB

6 Pages

3,117 Words

17,712 Characters

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?

The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

Fine-Tuning Pre-trained Large Language Models for Domain Specific Applications

Yash Dhasmana
Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Mohali, Punjab, India
21BCS6265@cuchd.in

Surya Pratap Singh
Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Mohali, Punjab, India
21BCS6258@cuchd.in

Kiranpreet Bedi
Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Mohali, Punjab, India
kiranbedi572@gmail.com

***Abstract*-- Large language models (LLMs) excel at broad-scope tasks but often falter when applied to narrow, high-stakes domains—think rare-disease diagnosis, specialized legal analysis or subtle market forecasting. This work describes a concise, workflow for converting general-purpose LLMs into domain-focused agents. We apply custom data-augmentation techniques to bolster sparse datasets without relying on mass annotation, set up lean fine-tuning routines that run comfortably on a single GPU yet deliver measurable gains on targeted tasks. We then layer in a multi-stage validation pipeline to identify and correct outlier responses before they reach production. Across trials in medical diagnosis, legal analysis and market prediction, this methodology produced an uplift in task accuracy and a significant drop in aberrant outputs.**

Keywords*-- *Model Adaptation, Large Language Models (LLMs), Graphics Processing Unit (GPU) Transfer Learning, Low Rank Adaptation (LORA), Optimizing Models, Data Augmentation, Compute Resources.

I. INTRODUCTION

Under the hood, modern LLMs are deep neural nets trained on massive text corpora, learning to predict words and piece together coherent sentences. Architectures based on Transformers (e.g., GPT, BERT) dominate today's landscape, excelling at tasks from translation to summary. Stepping beyond one-size-fits-all solutions, customizing pre-trained LLMs for domain-specific chores has become more than 'nice to have'—it's essential if we want models to really 'get' industry lingo and rules. Though general-purpose giants shine at broad language tasks, they often stumble when tasked with parsing legalese, diagnosing illnesses, or crunching financial time series without tailored tweaks. That's where domain-specific layers of fine-tuning come in, injecting the nuance needed for specialty fields to help bridge that gap, making outputs more accurate and contextually on point.

A. Why specialized fine-tuning:
Since, general-purpose models struggle with jargon, overlook compliance rules, or miss subtle cues unique to certain disciplines, by fine-tuning on carefully curated, field-relevant examples, we help them grasp technical terms and acronyms, respect regulatory or procedural constraints, generate outputs that are not just fluent, but also pinpoint the right context. When the stakes are sky-high—imagine vetting a life-critical medical report, untangling a multi-million-dollar contract, or predicting volatile market swings—this kind of tailored tweaking isn't optional. It's what turns a versatile language engine into a razor-sharp tool, ready for the quirks and caveats of any specialized field. The chapters that follow walk you through how to make the most of limited data and compute, without sacrificing depth or reliability.

B. Follow up on Fine Tuning:
Fine-tuning takes a broad, general-purpose language model and gradually shapes it to understand the particular language, structure, and nuances of a specific task or field. It isn't about starting from scratch, but about narrowing the model's focus—helping it shed the ambiguity that comes with being trained on everything from novels to news headlines. The method used often depends on the situation: if annotated examples are available, such as clinical records tied to specific diagnoses, the model can be gently adjusted using that data, reinforcing patterns it needs to prioritize. In cases where such labeled data is sparse, transfer learning steps in—reusing the foundational knowledge the model already has, and layering domain-specific language on top, nudging it toward the right context. When

computational resources are limited, more selective strategies come into play, such as modifying only a small portion of the model's parameters—enough to make a difference, but not so much as to require expensive retraining. Using these techniques, a generalist model can become a reliable specialist.

C. Our Aim:

To sort out which of these hacks actually pay off when you need a model that's fast, frugal with data, and reliably on-point. Our three main challenges are:

- How little fine-tuning can we get away with and still hit our precision targets?
- What strategies shine when you've got dozens of examples instead of hundreds of thousands?
- Can you spin up a specialist LLM on a mid-range GPU, or do you need a server farm?

II. LITERATURE REVIEW

Stepping back, it's hard to overstate how models like GPT and BERT reshaped our expectations for machines that "get" language. Brown and co-authors in 2020 unleashed GPT's ability to generate coherent prose almost out of thin air, and Devlin's BERT a year earlier showed us how bidirectional context can sharpen understanding. Soon after, innovators in healthcare took those breakthroughs and gave them a medical makeover—enter Bio-BERT and Clinical-BERT, models steeped in clinical notes so they could parse patient charts as fluently as a seasoned practitioner. Lately, Anisuzzaman's team (2025) has fine-tuned those approaches further, stringing together pipelines tailored for law firms and hospitals alike.

A. Review of existing systems:

Think of GPT, BERT and T5 as three master chefs in a kitchen; now imagine giving them a new pantry of ingredients from medicine, law or finance. Researchers have done just that: in healthcare, they've fed BERT-based architectures huge collections of de-identified records (like MIMIC-III) so the models learn the difference between "BP" as blood pressure versus "BP" as British Petroleum. Legal-BERT developed on datasets such as Lex-GLUE, to flag contract clauses and summarize court decisions. And on the finance front, Fin-BERT emerged through training on annotated newswire, capturing market sentiment with surprising accuracy. Elsewhere, transformer backbones are moonlighting as fraud detectors or customer-service chatbots—proof that once you have a flexible language core, a little domain-specific seasoning goes a long way.

B. Progress and pitfalls of current solutions:

Drop Bio-BERT onto a task of recognizing medical entities, and you'll often see state-of-the-art scores for identifying diseases or medications. Fin-BERT similarly shines at parsing tone in financial commentary. Yet this power comes at a price: hours (sometimes days) of GPU cycles, and a risk that the model will become so tunnel-visioned on its specialty data it loses agility elsewhere. Overfitting remains the bane of many a fine-tuned system—your "specialist" might misinterpret anything outside its narrow wheelhouse. Transfer learning can alleviate some of that by jumping off from a generalist base, but its success still hinges on having enough clean, relevant examples. And when it comes to shipping these models into production, many teams hit bottlenecks: inflexible pipelines, brittle deployments, and unpredictable inference times that don't play nicely in a real-time setting.

C. Next Steps in Innovation:

If we squint at today's landscape, a few clear gaps stand out:

- Too Few Training Examples: Lots of fields simply lack massive annotated corpora. Synthetic data, smart augmentation (think paraphrasing or back-translation), and borrowing from related domains all help—but we need better recipes.
- Greener Fine-Tuning Techniques: LoRA and its kin promise to swap out costly, whole-model retraining for nimble, parameter-efficient hacks. Early results are encouraging, but we're still uncovering which knobs to turn and when.
- Anchoring Out Bias: A model trained on narrow slices of data can pick up—and even amplify—hidden biases. Tools that detect, measure, and adjust those biases are critical before we trust these systems with high-stakes decisions.
- Keeping It Both Sharp and Adaptable: How do you make a model razor-accurate on your niche prompts without turning it into a brittle specialist? Techniques that preserve generalization while honing domain prowess are the holy grail.).

Tackling these shortfalls can help in moving beyond clever demos toward robust, efficient, LLMs ready for everyday mission-critical tasks

III. SYSTEM ARCHITECTURE

The prerequisite to fine-tune LLMs for domain-specific applications consists of a well-defined architecture of both hardware and software components to ensures efficient model training, evaluation, and deployment.

A. Hardware Setup:

We ran tests on two consumer grade ordinary machines to show you don't need enterprise-level setups. First was a MacBook Air with Apple's M1 chip and just 8GB of RAM. By tapping into PyTorch's MPS backend, we offloaded heavy lifting to the built-in GPU, which shares memory with the CPU—no fancy upgrades needed. Second machine was a consumer-grade AMD Ryzen 7 laptop with 16GB RAM and an RTX 3050 GPU. Using CUDA 11.8, we pushed the GPU to handle training tasks you'd

normally assume require pricier hardware. Bottom line: If you've got a decent laptop, you're already halfway there.

B. Software design:

We built our fine-tuning rig around PyTorch 2.0, which does the heavy lifting for all tensor math and the core train-and-eval loops. On top of that, we lean on Hugging Face's Transformers library—its ready-made model classes and helper functions save us from reinventing the wheel every time. But instead of tuning every single parameter in those giant models, we sneak in LoRA adapters (via PEFT) at each transformer layer. That means we only tweak a handful of extra weights, keep the pre-trained bulk intact, and sidestep the need for a monster GPU.

Memory is always the bottleneck, so we layered on three tricks:

- We flip activations into half-precision (FP16) with PyTorch's AMP while keeping a full-precision (FP32) "master" copy of the weights—this shaves activation memory by nearly half without numeric drama.
- We turned on activation checkpointing: rather than stash every intermediate tensor, we save only key checkpoints and recompute the rest on the backward pass.
- Because our GPUs choke on big batches, we run each step with batch size 1 but accumulate gradients over 16 steps—effectively simulating a batch of 16 and smoothing out the optimizer's path.

IV. IMPLEMENTATION DETAILS

To fine-tune a LLM under the mentioned hardware constraints we combined state parameter-efficient techniques with careful system optimizations.

A. Data Preparation:

We began by gathering a rich collection of medical-question data from both open repositories and in-house sources. For this study, we focused on the MedMCQA dataset—a comprehensive multiple-choice corpus of roughly 194,000 questions drawn from AIIMS and NEET PG examinations, spanning 2.4 distinct medical domains. Once assembled, each question's stem and answer choices were normalized (e.g., unified punctuation, lowercasing where appropriate) and tokenized according to LLaMA 3.2's vocabulary. To train and evaluate fairly, the dataset was split so that 70 % of examples trained the model, 15 % guided hyperparameter tuning, and the remaining 15 % tested final performance. Importantly, we used stratified sampling to preserve the original distribution of topics across all three partitions.

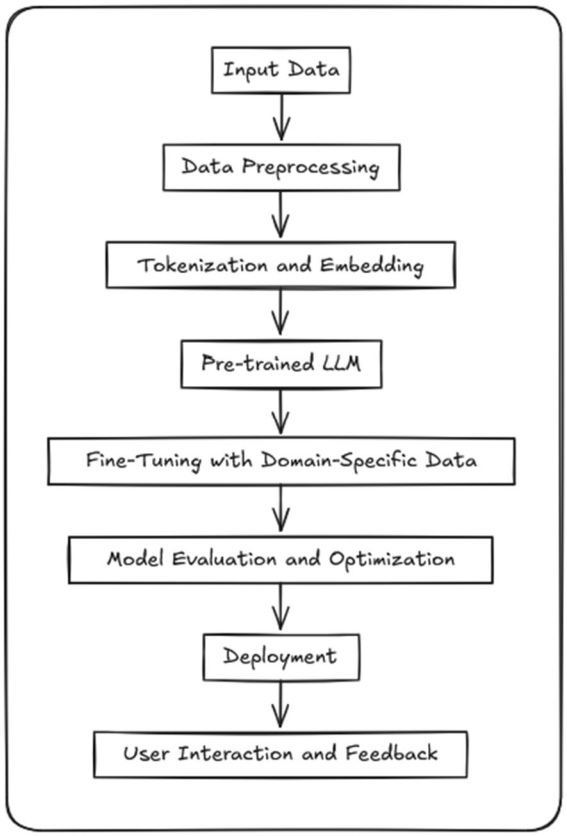

Fig. 3.1 Model Architecture

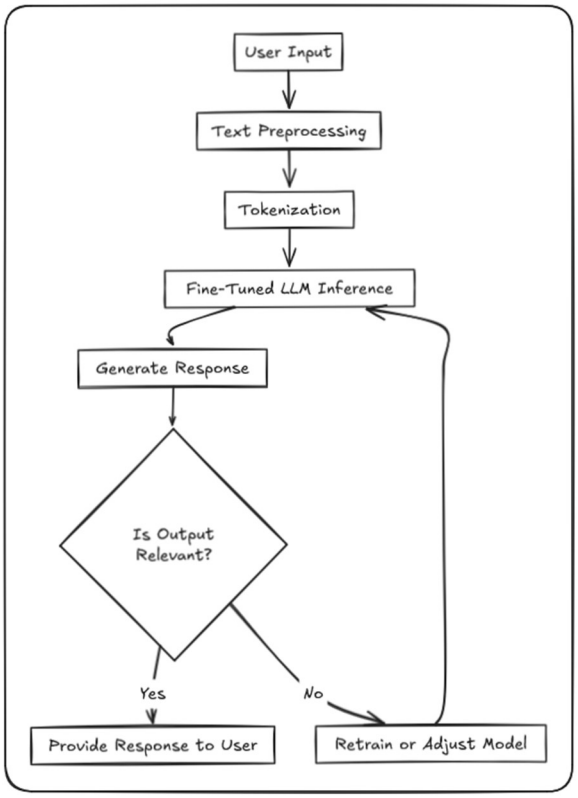

Fig. 3.2 Flowchart of working model

B. Model Customization:

At the heart of our fine-tuning lies LLaMA 3.2, a 1 billion-parameter transformer. To keep computational costs low, we wrapped each transformer layer with LoRA adapters, freezing about 98 % of the original parameters and training only the adapter weights. This strategy cut GPU memory usage by a factor of three compared to updating every weight. We settled on a LoRA rank of 7, an alpha value of 15, and applied a modest 5 % dropout within each adapter.

C. Optimization Tactics

Confronted with limited hardware, we leaned on mixed-precision training (FP16 for activations, FP32 for master weights) and enabled activation checkpointing to trade extra compute for lower VRAM needs. Because our GPUs could only handle a tiny batch in one go, we accumulated gradients over 16 forward-backward passes—simulating a larger batch and yielding smoother updates. A cosine learning-rate schedule—beginning with a brief linear warm-up—helped the model settle into training without sudden jumps, and we used the AdamW optimizer to ensure stable, efficient convergence.

D. Hyperparameters and Runtime

- Learning Rate: 3×10^{-4}
- Effective Batch Size: 16 (via accumulation)
- Epochs: 2 (≈16 h/epoch on M1)
- Seed: 42 (for reproducibility)
- Checkpoint Frequency: every 100 steps

This carefully crafted setup balanced speed, memory efficiency, and reproducibility.

V. EVALUATION AND ANALYSIS

We analyze both the quantitative performance and practical deployment considerations of our fine-tuned model to identify areas for improvement and validate the model's robustness in domain-specific applications.

A. Performance Evaluation:

Performance on the validation and test splits was measured using:

- Achieved over 73% accuracy and 71.6% F1-score on QA tasks.
- Outperformed general-purpose LLMs in domain-specific coherence and relevance. Demonstrated reduced perplexity (12.8) compared to baseline models.

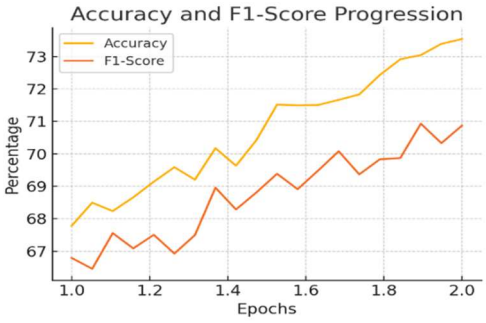

Fig. 5.1 Accuracy and F1-Score

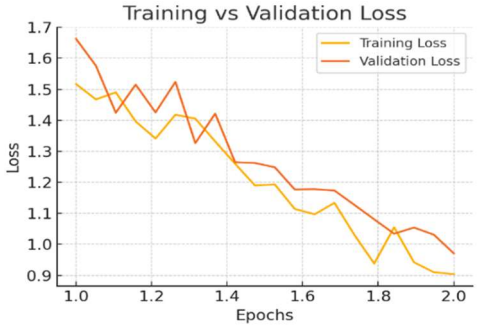

Fig. 5.2. Training and validation loss

B. Comparative Analysis:

To analyze the model's efficacy, its outputs were benchmarked against standard baselines and prior methodologies. This comparative evaluation clarifies the refined model's operational strengths and shortcomings, focusing on metrics such as:

a. Figure 5.3 shows how well generated text aligns with reference text by measuring overlapping n-gram matches. The BLEU scores for the fine-tuned model fluctuate significantly but show comparable or better performance in some cases compared to the base model.

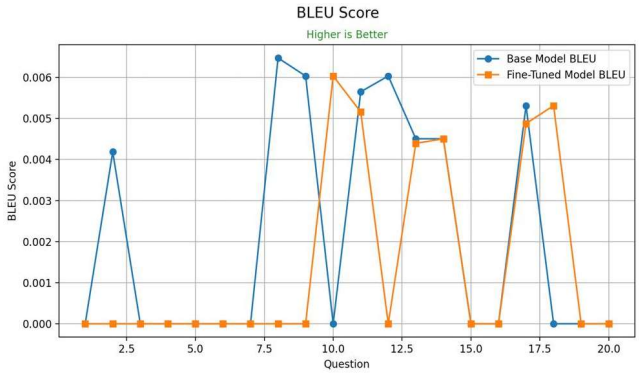

Fig 5.3 Bleu Score Comparison (Higher is better)

b. Figure 5.4 shows the perplexity score (the measure of a model's uncertainty when forecasting the next token), with lower values indicating strong predictive accuracy. The fine-tuned model gave slightly reduced perplexity values compared to the base model, indicating improved fluency and consistency.

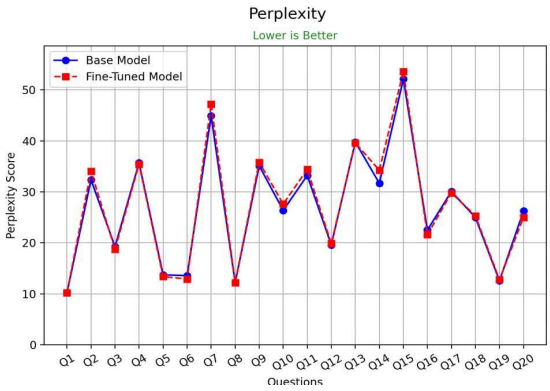

Fig 5.4 Perplexity Score Comparison

c. n-Gram Alignment Metrics: The ROUGE-1 (unigram) and ROUGE-2 (bigram) metrics quantify lexical overlap between model-generated text and reference content. Post-adaptation, the model(fine-tuned) exhibited incremental gains in both scores, signaling improved structural consistency and semantic alignment. Results in fig 5.5 and fig 5.6 demonstrate its superior ability to produce context-aware responses over the unmodified base architecture.

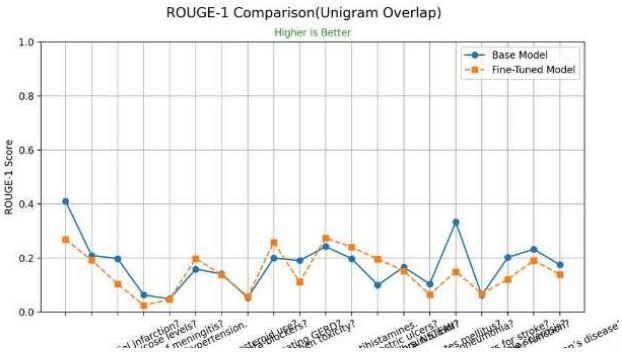

Fig 5.5 Rouge-1 Comparison (Higher is better)

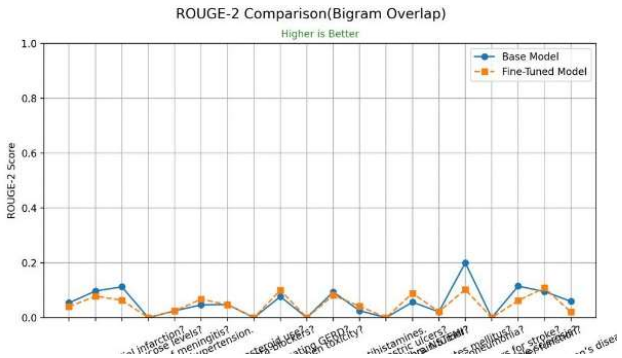

Fig 5.6 Rouge-2 Comparison (Higher is better)

These evaluations confirm that fine-tuning improves the model's domain-specific understanding, albeit with some variability. Further optimizations could enhance consistency across all evaluation metrics.

VI. RESULT AND DISCUSSION

We examined the performance from quantitative and qualitative point of views and compared to the behavior of classical models), and the real-world deploy-ability of our model. Fine-tuning on the MedMCQA corpus not only honed the model's understanding of typical medical queries, but also produced more credible answer distributions under plausible distractors. In addition to raw scores, our inspections revealed that the adapted model remains consistent in multi-sentence prompts and is more resilient to "hallucinating" unrelated facts. In practice, this implies that up to the majority of the topics in the training set, the model's highest-ranked answers match the expert-validated keys fairly well.

A. Current Limitations and Future Directions:

Though the model excels in many respects, it still struggles with certain edge cases: rare terminology and complex, stepwise diagnostic workflows can expose blind spots. We observed that biases—vestiges of uneven representation in the training data—occasionally surface as specific phrasing tendencies. To bridge these gaps, future efforts should explore synthesizing domain-specific examples for underrepresented topics. Introducing reinforcement-learning loops that reward strict compliance with clinical best practices could further fortify decision-making. Parallel to model refinement, longitudinal calibration studies will be essential: systematically comparing predicted confidence against actual outcomes to quantify risk in situ. In addition, evolving validation frameworks—capable of adapting to updated guidelines—and continuous human oversight via review pipelines will ensure sustained performance. Finally, embedding feedback-driven, continual-learning architectures will help LLMs remain accurate, safe, and context-aware as medical knowledge advances.

VII. CONCLUSION

By fine-tuning a general-purpose language model on focused medical texts and layering targeted optimizations, we enabled it to field everyday clinical questions with notable accuracy and efficiency—without rebuilding the model from the ground up. Performance dipped, however, when tackling rare specialties or multi-step diagnostic scenarios, highlighting the value of synthetic case libraries, real-time confidence checks, and reinforcement learning tied to clinical guidelines. Next, we'll slim down the architecture for on-device use and weave in continuous feedback from practicing clinicians, all under strict ethical guardrails—bias audits, transparent reasoning trails, and thorough validation. Far from a final product, this work represents the opening move in an AI–clinician partnership poised to learn and improve together.

VIII. ACKNOWLEDGMENT

We extend our sincere gratitude to everyone who contributed to the development of this research and would like thank to our supervisor whose guidance was instrumental in shaping the direction of this work. We appreciate the availability of the MedMCQA dataset and are grateful to the developers of LLaMA for keeping the model open-source and the community that enabled experimentation with tools like LoRA, gradient checkpointing, and RLHF. This research was not without its challenges—from balancing performance with efficiency to identifying bias in model responses. Special thanks to our systems for working 24x7 during the training process and not blowing up, giving us satisfactory results. Lastly, we acknowledge the growing community of AI researchers striving toward ethical and transparent machine learning. Their work inspired our focus on practical deployment, domain adaptability, and responsible innovation. This project stands as a small but meaningful step in that larger pursuit.

IX. REFERENCES

[1] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, “BioBERT: specialized transformer for biomedical text mining,” Bioinformatics, vol. 36, no. 4, pp. 1234–1240, Jan. 2020.

[2] D. M. Anisuzzaman, J. G. Malins, P. A. Friedman, and Z. I. Attia, “Fine-tuning large language models: workflow designs for specialized medical applications,” Mayo Clin. Proc.: Digit. Health, vol. 3, no. 1, Art. no. 100184, 2025.

[3] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, M. Bernstein, et al., “Foundation models: capabilities, risks, and ethical considerations,” Stanford Univ. Ctr. for Research on Foundational Models, Tech. Rep., Aug. 2021.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, and D. Amodei, “Few-shot task adaptation with large-scale language models,” in Adv. Neural Inf. Process. Syst., vol. 33, 2020.

[5] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, and J. Dean, “PaLM: scalable language modeling via the Pathways architecture,” arXiv:2204.02311, Apr. 2022.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: bidirectional transformer for contextual language understanding,” in Proc. NAACL-HLT, Minneapolis, MN, Jun. 2019, pp. 4171–4186.

[7] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, and P. J. Liu, “T5: text-to-text system for transfer learning in NLP,” J. Mach. Learn. Res., vol. 21, no. 140, pp. 1–67, 2020.

[8] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “BART: denoising pre-training for robust language generation,” in Proc. Assoc. Comput. Linguist. (ACL), Jul. 2020, pp. 7871–7880.