Radiology

REVIEWS AND COMMENTARY • REVIEW

Open-Source Large Language Models in Radiology: A Review and Tutorial for Practical Research and Clinical Deployment

Cody H. Savage, MD • Aduway Kanhere, MSE • Vishwa Parekh, PhD • Curtis P. Langlotz, MD, PhD • Anupam Joshi, PhD • Heng Huang, PhD • Florence X. Doo, MD, MA

From the University of Maryland Medical Intelligent Imaging (UM2ii) Center, Department of Diagnostic Radiology and Nuclear Medicine, University of Maryland School of Medicine, 22 S Greene St, Baltimore, MD 21201 (C.H.S., A.K., V.P., F.X.D.); Departments of Radiology, Medicine, and Biomedical Data Science, Stanford University, Palo Alto, Calif (C.P.L.); Department of Computer Science and Electrical Engineering, College of Engineering and Information Technology, University of Maryland, Baltimore County, Baltimore, Md (A.J.); Department of Computer Science, University of Maryland, College Park, College Park, Md (H.H.); and University of Maryland Institute for Health Computing, University of Maryland, North Bethesda, Md (H.H., F.X.D.). Received April 10, 2024; revision requested May 2; revision received July 17; accepted July 24. **Address correspondence to** F.X.D. (email: *fdoo@som.umaryland.edu*).

F.X.D. is supported in part by an Association of Academic Radiology Clinical Effectiveness in Radiology Research Academic Fellowship Award and a grant from the Johns Hopkins Mid-Atlantic Center for Cardiometabolic Health Equity, which is supported by the National Institute on Minority Health and Health Disparities (P50MD017348). The content is solely the responsibility of the authors and does not necessarily represent the official views of the Mid-Atlantic Center for Cardiometabolic Health Equity or the National Institutes of Health. C.P.L. is supported in part by the Medical Imaging and Data Resource Center, which is funded by the National Institute of Biomedical Imaging and Bioengineering (75N92020D00021). H.H. is partially supported by the National Institute on Aging (U01 AG068057), National Institute of Biomedical Imaging and Bioengineering (R01 EB034116), National Institute of General Medical Sciences (R01 GM148743, R01 GM141076), and National Science Foundation (IIS 2347592, 2347604, 2348159, 2348169, DBI 2405416, CCF 2348306, CNS 2347617). H.H. and F.X.D. are supported in part by the Montgomery County, Maryland, and University of Maryland Strategic Partnership (MPowering the State), a formal collaboration between the University of Maryland, College Park, and the University of Maryland, Baltimore.

Conflicts of interest are listed at the end of this article.

Radiology 2025; 314(1):e241073 • https://doi.org/10.1148/radiol.241073 • Content code: AI

Integrating large language models (LLMs) into health care holds substantial potential to enhance clinical workflows and care delivery. However, LLMs also pose serious risks if integration is not thoughtfully executed, with complex challenges spanning accuracy, accessibility, privacy, and regulation. Proprietary commercial LLMs (eg, GPT-4 [OpenAI], Claude 3 Sonnet and Claude 3 Opus [Anthropic], Gemini [Google]) have received much attention from researchers in the medical domain, including radiology. Interestingly, open-source LLMs (eg, Llama 3 and LLaVA-Med) have received comparatively little attention. Yet, open-source LLMs hold several key advantages over proprietary LLMs for medical institutions, hospitals, and individual researchers. The wider adoption of open-source LLMs has been slower, perhaps in part due to the lack of familiarity, accessible computational infrastructure, and community-built tools to streamline their local implementation and customize them for specific use cases. Thus, this article provides a tutorial for the implementation of open-source LLMs in radiology, including examples of commonly used tools for text generation and techniques for troubleshooting issues with prompt engineering, retrieval-augmented generation, and fine-tuning. Implementation-ready code for each tool is provided at *https://github.com/UM2ii/Open-Source-LLM-Tools-for-Radiology*. In addition, this article compares the benefits and drawbacks of open-source and proprietary LLMs, discusses the differentiating characteristics of popular open-source LLMs, and highlights recent advancements that may affect their adoption.

Supplemental material is available for this article.

Large language models (LLMs) are transforming radiology and health care (1–5); however, clinical research has predominantly focused on proprietary models rather than open-source LLMs. Proprietary models are LLMs owned by a company and whose use requires a purchased license (6–9). In contrast, open-source LLMs are free to access, with limited or no restrictions on modification or use (10). Open-source LLMs offer unique advantages for clinical use and research, such as cost savings, greater customizability, and data security. In this review, we provide a general understanding of the advantages and disadvantages of open-source LLMs, a roadmap for implementing open-source LLMs, with practical examples, and access to code for radiologic and clinical research purposes for technologically interested radiologists (Fig 1) *(https://github.com/UM2ii/Open-Source-LLM-Tools-for-Radiology)*. Additionally, we explore clinical deployment considerations and the latest developments on the horizon, including multimodal foundation models and their implications for radiology. A glossary of terms is provided in Table S1.

Advantages and Disadvantages of Open-Source LLMs

Here we highlight key advantages and disadvantages of using open-source LLMs versus proprietary LLMs (for a summary, see Table 1).

Advantages

Customizability.—Some proprietary LLM companies (eg, OpenAI) have begun to offer advanced uses, including fine-tuning, plugin creation, and customized model hosting with searchable documents on their platform (11); however, users have limited control over the model and its available features. Open-source LLMs provide users with full access to (and control over) model weights (numerical values defining neuron connection strength) and configuration files, which can be downloaded, modified, copied, shared, or merged without restriction. This allows developers to fine-tune and deploy open-source LLMs for personal, institutional, and entrepreneurial

Abbreviations

AI = artificial intelligence, API = application programming interface, LLM = large language model, LoRA = Low-Rank Adaptation, RAG = retrieval-augmented generation

Summary

Open-source large language models and multimodal foundation models offer several practical advantages for clinical and research objectives in radiology over their proprietary counterparts but require further validation before widespread adoption.

Essentials

- Open-source large language models (LLMs) offer several key benefits over closed-source proprietary LLMs that are of interest to radiologists, researchers, and hospital systems; these benefits include reduced operating costs, increased customizability, accessibility, entrepreneurial opportunities, and enhanced data security.
- The performance of open-source LLMs lags behind that of the top-performing proprietary LLMs (eg, GPT-4), but this gap has been shrinking.
- Prompt engineering, retrieval-augmented generation, and fine-tuning may be used to troubleshoot open-source LLM issues such as poor reasoning performance or insufficient knowledge base.
- Decisions on whether to deploy an open-source LLM should be guided by a use case–driven approach, where stakeholders attempt to address a clear problem and specific metrics are available to evaluate model performance in clinical practice.
- Emerging LLM technologies to recognize include foundation models (which can holistically integrate information from multiple modalities [eg, image, text, audio]) and agents (which autonomously perform tasks and orchestrate complex operations).

uses (further discussion in the "Fine-tuning" section). This level of control over open-source LLMs and their model versions is important for health care users, as applications and workflows using a proprietary LLM may be reliant on a certain version of a model, the updating or deprecation (ie, retiring) of which may lead to unpredictable behavior or outright failures. At scale, the customizability of open-source LLMs empowers a wider user community of developers, researchers, and entrepreneurs to create diverse applications from diagnostics to research, accelerating innovation and adoption within an open artificial intelligence (AI) ecosystem.

Cost and licensing.—Proprietary models charge fees for personalized use, and developers notably do not have rights to the fine-tuned proprietary model itself. On the other hand, open-source LLMs are free to use, reducing financial barriers, and are typically governed by free and open-source software licenses like Apache 2.0 (12). These open-source licenses offer flexibility for customization, distribution, and commercial use without restrictions or fees. This means users have the ability to modify, share, and even sell their open-source LLM adaptations—for example, an open-source LLM tailored to specific radiologic tasks—without paying fees or obtaining permissions. Note that while this flexibility is beneficial on the technological side, ethical and legal compliance in clinical settings remains uncharted territory.

Size.—Open-source LLMs have pioneered smaller sizes of models (13,14), reducing both the financial and environmental footprints associated with LLMs that would otherwise arise from larger computational requirements (15,16). Their portable size increases their accessibility, as they can be deployed on low-power edge devices (like mobile phones and radiologist workstations) and can be run locally without an internet connection in low-bandwidth or remote environments (eg, rural locations or aerospace). This portability ensures that open-source LLMs are unaffected by disruptions in service, upgrades, or application programming interface (API) maintenance. Additionally, many applications require low-latency real-time responses, and smaller models offer faster, more feasible response times. Overall, smaller models allow users to iterate quickly and develop new techniques or architectures, democratizing access to advanced natural language processing capabilities (17,18). As regulations around AI and data privacy tighten, these smaller, more controllable models may offer less burdensome compliance pathways than larger, more complex systems.

Security.—Proprietary LLMs rely on third-party computational resources and can only be accessed through an API that specifies how user data will be sent to a server and how the LLM will respond back to the user. Specific licenses or precautions must be in place to ensure query data are not used to update or train the proprietary model. Of note, many major cloud-based vendors are Health Insurance Portability and Accountability Act–compliant, with more advanced security capabilities than local hospital systems or research laboratories (19); both open-source LLMs and proprietary models can be deployed to cloud environments. However, open-source LLMs can be fully downloaded to run either locally on-site or on personal or enterprise cloud instances, which limits the potential for data leakage.

Disadvantages

Performance.—Proprietary LLMs lead in performance metrics, and are at the top of most LLM leaderboards (eg, Chatbot Arena, OpenCompass, Holistic Evaluation of Language Models [HELM]), leapfrogging one another in an ever-accelerating arms race of innovation (17,18,20–24). However, open-source LLMs are rapidly closing the gap (eg, Athene-V2 [Nexusflow] in Chatbot Arena) (Fig S1). To meet clinical standards, this performance gap needs to be addressed—particularly in quantitative measurements of abstraction and reasoning, multitask language understanding, truthfulness, and mathematical problem-solving.

Safety.—While there are potential advantages of running an open-source LLM locally, implementing robust safety measures to prevent harmful outputs remains a major challenge. For instance, OpenAI has released a scorecard for o1 (25), and Google DeepMind's earlier Sparrow model incorporated 23 safety rules (26), but similarly transparent safeguards in other LLMs are not readily available for review. Additionally, creative prompting has been used to circumvent these established safeguards (termed *prompt injection*), leading to a game of cat and mouse where LLM developers attempt to fortify their models against such exploits (27). To prevent these and other exploits, proprietary LLMs encourage ongoing simulated security testing (called *red teaming*) and public assessment, for

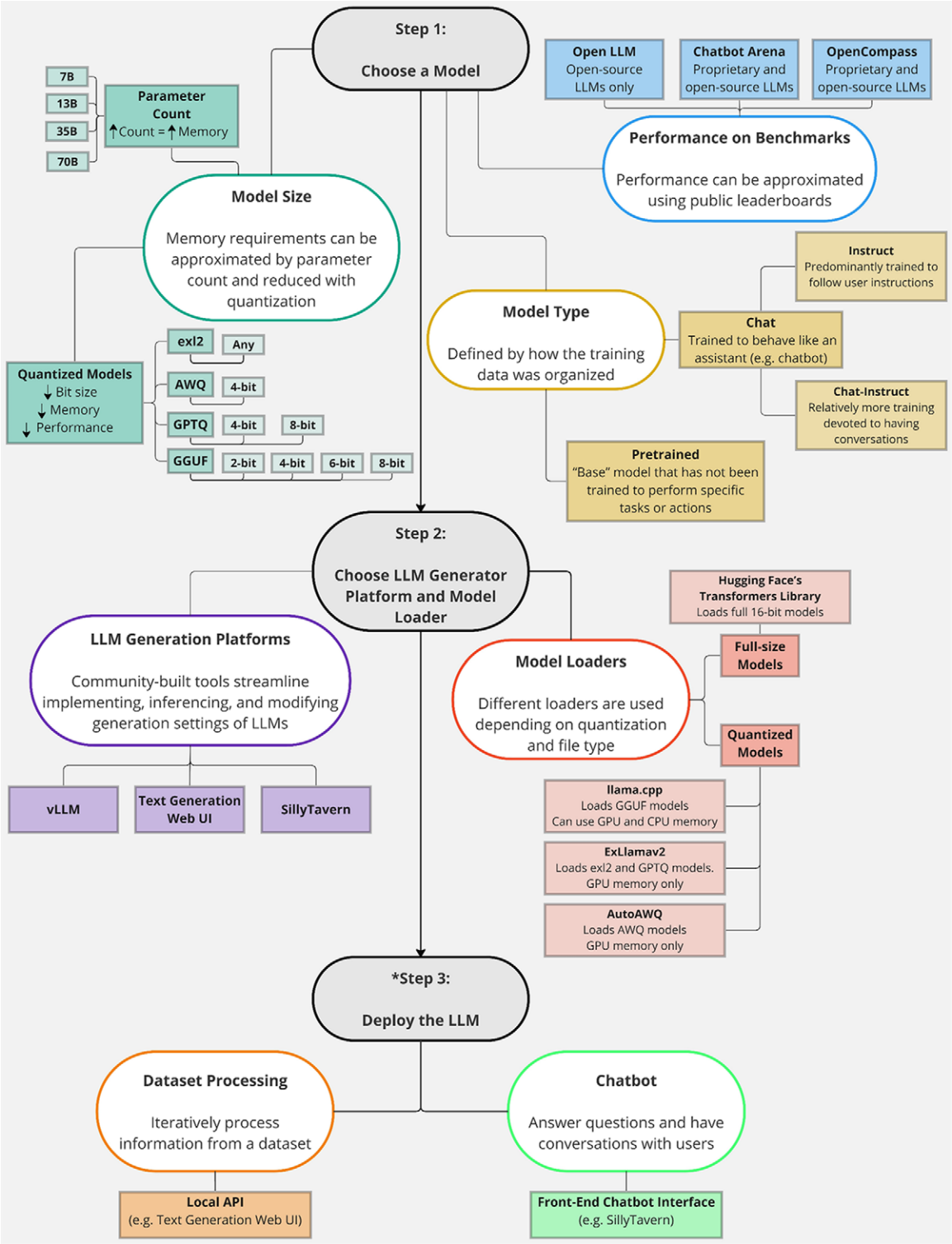


Figure 1: Diagram of a step-by-step roadmap for open-source large language model (LLM) implementation. Step 1: Select an appropriate model based on size (green), performance (blue), and type (gold). Model size can be approximated by the parameter count. Higher counts require more memory to load. Quantized models have reduced memory requirements, albeit with some performance loss. Various public leaderboards rank model performance by measuring ability across several spectrums (eg, multitask language understanding, arithmetic reasoning). Choosing the correct model type depends on the intended use case. For example, *instruct* models are best suited for performing an instructed task, such as classifying radiology reports. A *pretrained* model has limited utility until fine-tuned for a specific task. *Chat-instruct* models are better suited for conversational functions. Step 2: Load the model using open-source LLM generation platforms such as Text Generation Web UI (purple). Then, the optimal loader is selected depending on the quantization status (red). Full-size models can be loaded with Hugging Face's Transformers library. Quantized models use loaders specific to their quantization method. Step 3: Deploy the model for a specific task or objective (orange or green). A local application programming interface (API) can be created with an open-source LLM generation platform and used to iteratively process data in an automated fashion. * = The deployment step and associated code *(https://github.com/UM2ii/Open-Source-LLM-Tools-for-Radiology)* are intended for research settings only. For clinical settings, refer to the "Clinical Deployment Considerations" section for critical points. AWQ = Activation-aware Weight Quantization, GGUF = GPT-Generated Unified Format, GPTQ = Post-Training Quantization.

Table 1: Comparative Advantages of Open-Source LLMs versus Proprietary LLMs

Metric	Open-Source LLMs	Proprietary LLMs	Overall Advantage*
Performance	Open-source LLMs generally underperform compared with proprietary LLMs in abstraction, reasoning, multitask understanding, truthfulness, and arithmetic; this performance gap is closing, with Mixtral-8×7B Instruct v0.1 (Mistral AI) surpassing Gemini Pro (Google DeepMind), Claude 2.1 (Anthropic), and GPT-3.5 Turbo (6/13/2023 version; OpenAI) on the Chatbot Arena leaderboard	Proprietary LLMs lead in most leaderboards (Chatbot Arena, OpenCompass); proprietary LLMs are more resilient to prompt injections and are less likely to generate potentially harmful content†	Proprietary
Customizability	Open-source LLMs allow users to customize text generation settings, such as temperature and frequency penalty‡; open-source LLMs enable full control of the model's version, unlike proprietary LLMs that may become deprecated (ie, retired) and unavailable; open-source LLMs can undergo fine-tuning (ie, additional training) or be merged with other open-source LLMs	Proprietary LLMs have limited customizability in text generation settings; there is a risk of model versions becoming deprecated and unavailable via the API, which could cause potential disruptions to hospital workflows and applications reliant on a specific model version	Open-source
Computational and financial cost	Open-source LLMs are smaller than proprietary LLMs due to high training costs and can be run on consumer hardware, like home workstations; innovations have enhanced the performance, context length, and inference speed of open-source LLMs without increasing size; running open-source LLMs locally incurs no API costs	Proprietary LLMs are larger than open-source LLMs, and parameter counts continue to trend upwards (eg, 340 billion for Med-PaLM 2 [Google] and 1.76 trillion [reported but unconfirmed] for GPT-4 [OpenAI]); interaction with these models programmatically is only possible through an API that charges for each input and output token	Open-source
Licensing and innovation opportunities	Developers have full access to modify, add features to, and fine-tune the LLM; most open-source LLMs are licensed under Apache 2.0, a permissive software license that allows developers to modify the LLM, create derivative works, and distribute (including selling) without royalty fees or restrictions	Some companies with proprietary LLMs (eg, OpenAI) offer fine-tuning, plugin creation, and hosting of fine-tuned models on their platform (eg, GPT Store); usage of these fine-tuned models by others provides reimbursement to developers; royalty fees are charged for business applications using these models	Open-source
Data security	Open-source LLMs can be run locally or on personal or enterprise cloud instances; running open-source LLMs locally eliminates the need for external data transmission and third-party interactions; however, this approach depends heavily on heterogeneous local information technology security resources (hospital systems or laboratories), which can be more vulnerable to cyberattack; typically, running a private open-source LLM on a secure HIPAA-compliant cloud vendor may be a better option	Proprietary LLMs can be used publicly or run on personal or enterprise cloud instances; any instance of data sharing to a publicly run proprietary LLM increases the risk of data leaks, especially if the data are used for subsequent model training, which is a data privacy and security concern; however, the cloud systems used by proprietary LLMs have dedicated advanced security capabilities, and some cloud-based vendors are even HIPAA compliant	Tie
Safety and alignment	Generally, open-source LLMs have received less robust evaluations of the extent to which they generate harmful outputs (sometimes intentionally, as with "uncensored" open-source LLMs); finding and addressing all vulnerabilities to adversarial attacks is often too expensive and time-consuming for developers with limited resources	Proprietary LLMs commonly undergo extensive assessments probing the efficacy of safety measures with expert groups (ie, *red teaming*); greater resources enable more training toward defense against adversarial attacks, such as prompt injections	Proprietary

Note.—API = application programming interface, HIPAA = Health Insurance Portability and Accountability Act, LLM = large language model.

* This column indicates which type of LLM has the relative advantage over the other.

† Prompt injections are adversarial attacks that use user input text to exploit a gap in the LLM alignment training or system prompting to cause the LLM to leak sensitive information (eg, protected health information), generate offensive language, or engage in other malicious actions. Adversarial attacks are defined as attempts to "trick" the LLM into making a mistake or performing an action that was not the intended use of the LLM according to the model creators.

‡ The temperature setting adjusts the level of randomness (ie, variability) of the model's outputs. The frequency penalty setting adjusts the model's probability of generating tokens that have been previously generated within the same output (eg, increasing the frequency penalty decreases the likelihood that the LLM repeats itself).

Table 2: Characteristics of a Small Selection of Highly Cited Open-Source LLMs

LLM (Provider)	Parameter Count (Billions)	Video RAM Requirement (GB)*	Video RAM Requirement (4-Bit Quantized)†	Context Length (Tokens)	Architecture‡	Model Type	MoE
Llama 2 Chat (Meta)	7B, 13B, 70B	13§ (7B), 24§ (13B), 72§ (70B)	3.9 (7B), 7.3 (13B), 36.0 (70B)	4096	LlamaForCausalLM	Pretrained, chat	No
Llama 3.1, Llama 3.2, Llama 3.3 (Meta)	1B to 405B	20§ (7B), 150§ (70B)‖	10.0 (7B), 75.0 (70B)	128000	LlamaForCausalLM	Pretrained, instruct, vision-language	No
Alpaca (Stanford University)	7B	13§	4.0§	4096	LlamaForCausalLM	Instruct	No
Mistral-Instruct (10/2024 version), Pixtral-Instruct (11/2024 version) (Mistral AI)	7B, 8B, 124B	15§ (7B), 16§ (8B), 248§ (124B)‖	7.5§ (7B), 8§ (8B), 124§ (124B)	128000	MistralForCausalLM	Pretrained, instruct, vision-language	No
Mixtral (version 0.1) (Mistral AI)	8 × 22B	282§	141§	32768	MixtralForCausalLM	Pretrained, instruct	Yes
Qwen2.5, Qwen2.5-Instruct, Qwen2-VL-Instruct (Alibaba)	0.5B to 72B	15 (7B), 85 (72B)‖	5.5 (7B), 42.5 (72B)	128000	QWenLMHeadModel	Pretrained, instruct, vision-language	No
Yi Chat, Yi 200K (01.AI)	6B, 34B	15.0§ (6B), 72.0§ (34B)	3.9§ (6B), 19§ (34B)	4096, 200000	LlamaForCausalLM	Pretrained, chat	No
Helion (Weyaxi)#	4 × 34B	140§	71.2**	200000	MixtralForCausalLM	Chat-instruct	Yes
LLaVA-Med-v1.5-Mistral (Microsoft)	7B	16§	8§	32768	LlavaMistralForCausalLM	Chat-instruct, vision-language	No

Note.—Characteristics of a few open-source LLMs with at least 100 citations are provided (except Helion, which is included to provide an example of a merged LLM). A more comprehensive list can be found at *https://github.com/UM2ii/Open-Source-LLM-Tools-for-Radiology*. These models do not have application programming interface costs (ie, free input and free output), but there may be other costs that should be considered, including an institution's computational capabilities (locally on-site or through a cloud service provider), additional hardware or cloud computing that may be needed, energy costs (if local), and information technology staffing requirements. LLM = large language model, MoE = mixture of experts, RAM = random-access memory.

* The video RAM requirement can vary depending on variables such as the context length set by the user, but parameter count can give a rough estimate.

† Calculated using PTQ (Post-Training Quantization) (35).

‡ Architecture name from the configuration file of the model hosted on Hugging Face.

§ Values are approximate.

‖ Due to the large number of different model versions, only a few representative examples are shown.

The Helion model is a merged MoE model based on Yi-34B fine-tuned models (base model is nontoxic-bagel-34b-v0.2; "experts" are bagel-dpo-34b-v0.2, Nous-Hermes-2-Yi-34B, SUS-Chat-34B, and platypus-yi-34b).

** Model size for GPTQ was not available; GGUF (GPT-Generated Unified Format) version used instead.

continuous scrutiny by a wide array of users, potentially unveiling vulnerabilities faster than they can be addressed (28). Managing these safety and security issues is time-consuming and expensive, likely asymmetrically and negatively affecting the practicality of open LLMs, which often have less resources and funding. Regardless of model type, if these issues are not fully addressed, this could lead to potential undesired clinical output vulnerabilities, such as perpetuating bias and medical misinformation (29,30).

Roadmap: Adapting Open-Source LLMs to a Use Case

In this section, we aim to equip general radiologists interested in the technology with the ability to use, experiment with, and implement open-source LLMs. Our goal is to reduce the technical barrier to entry by providing a roadmap with a step-by-step guide (Fig 1) for open-source LLM implementation in clinical and/or research workflows.

Choosing an Open-Source LLM

Model performance and leaderboards.—The first step to implementing open-source LLMs is choosing which model to use (Fig 1, step 1). Most open-source LLMs can be downloaded from the model section of Hugging Face (31), the most popular web-based hub for hosting models. A reasonable initial approach is to browse LLM leaderboards for the current top-performing LLMs.

Table 3: Characteristics of Proprietary LLMs

LLM (Provider)	API Cost ($/Million Tokens)*		Parameter Count	Video RAM Requirement (GB)†	Context Length (Tokens)	MoE
	Input	Output				
GPT-4 Turbo 2024-04-09 (OpenAI)	10	30	1.7 trillion	Approximately 1500	128 000	Yes‡
GPT-3.5 Turbo Instruct (OpenAI)	0.5	1.5	175 billion	Approximately 150	16 385	Unknown
Claude 3.0 Opus (Anthropic)	15	75	Unknown	Unknown	200 000	Unknown
Claude 3.0 Sonnet (Anthropic)	3	15	Unknown	Unknown	200 000	Unknown
Gemini 1.0 Ultra (Google DeepMind)	NA	NA	Unknown	Unknown	1 000 000	Unknown
Gemini 1.5 Pro (Google DeepMind)	7	21	Unknown	Unknown	1 000 000	Unknown
Mistral Large (Mistral AI)	6	12	Unknown	Unknown	32 000	Unknown

Note.—All models are chat-instruct models with unknown architectures. API = application programming interface, LLM = large language model, MoE = mixture of experts, NA = not available, RAM = random-access memory.

* The API cost is provided per 1 million tokens (approximately 750 000 words). There are also additional costs that should be considered, including an institution's computational capabilities (locally on-site or through a cloud service provider), additional hardware or cloud computing that may be needed, energy costs (if local), and information technology staffing requirements.

† The video RAM requirement can vary depending on variables such as the context length set by the user, but parameter count can give a rough estimate. The 4-bit quantized video RAM requirement calculated using PTQ (Post-Training Quantization) (35) is unknown for all models.

‡ GPT-4 is rumored to be an MoE model, but this is unconfirmed.

Table 4: Summary of How to Troubleshoot Open-Source LLM Performance Issues

LLM Performance Issue	Example	Solutions	
		Without Additional Training	With Additional Training (ie, Fine-tuning)
Deficiencies in complex reasoning	LLM is asked to generate a differential diagnosis but only lists diagnoses with single-step reasoning, without considering pertinent negatives or other diagnoses in the differential	Prompt engineering: prompt the model to "think step by step" or use other prompting techniques (eg, Chain of Thought, Reflexion) to improve the logical consistency of responses	Fine-tune with training examples of the task where every step and its reasoning are specified; DPO* may be the preferred training type in this scenario
Insufficient knowledge base and hallucinations (ie, faulty information)	LLM is asked to provide the newest guidelines for a particular diagnosis but provides the previous guidelines as the guidelines were updated after the LLM was originally trained	RAG: Use RAG to connect to and supplement the input prompt with information from a trusted source (eg, *RadioGraphics* articles)	Fine-tune with question-and-answer-style training examples of the information you want the LLM to learn; SFT† may be the preferred training type in this scenario
Poor instruction following	LLM is instructed to generate the impression section of a report but provides lengthy, tangential responses that stray from the expected formatting of radiology reports despite prompting the LLM for concise answers	None	Fine-tune with training examples of the task in an instruction-response style with the preferred formatting; RLHF‡ or DPO may be the preferred training type in this scenario

Note.—DPO = direct preference optimization, LLM = large language model, RAG = retrieval-augmented generation, RLHF = reinforcement learning from human feedback, SFT = supervised fine-tuning.

* DPO uses human-selected preference data of LLM outputs to directly train the LLM.

† SFT uses human-labeled data in the form of a prompt and desired response.

‡ In RLHF, LLM responses are ranked by humans and used to train a reward model that provides feedback on the LLM's outputs, followed by the LLM adjusting its outputs to maximize the reward, to align LLM outputs with human preferences. RLHF would be preferred for more general instruction following that includes a greater breadth of training examples, while DPO would be preferred for more targeted instruction following examples.

Each leaderboard uses different inclusion criteria and metrics to generate rankings (17,18,20,22,23). For example, Open LLM Leaderboard includes only open-source LLMs and ranks each LLM using the average score on six benchmarks (Abstraction and Reasoning Corpus, HellaSwag, Massive Multitask Language Understanding, TruthfulQA, WinoGrande, and GSM8K) that test commonsense reasoning, multitask language understanding, truthfulness, and arithmetic reasoning. However, Chatbot Arena uses human evaluations to compare the performance of LLMs in generating responses to open-ended questions (as opposed to automated methods of verifying LLM responses such as using pass@K or GPT-4 [OpenAI] as a judge) (24).

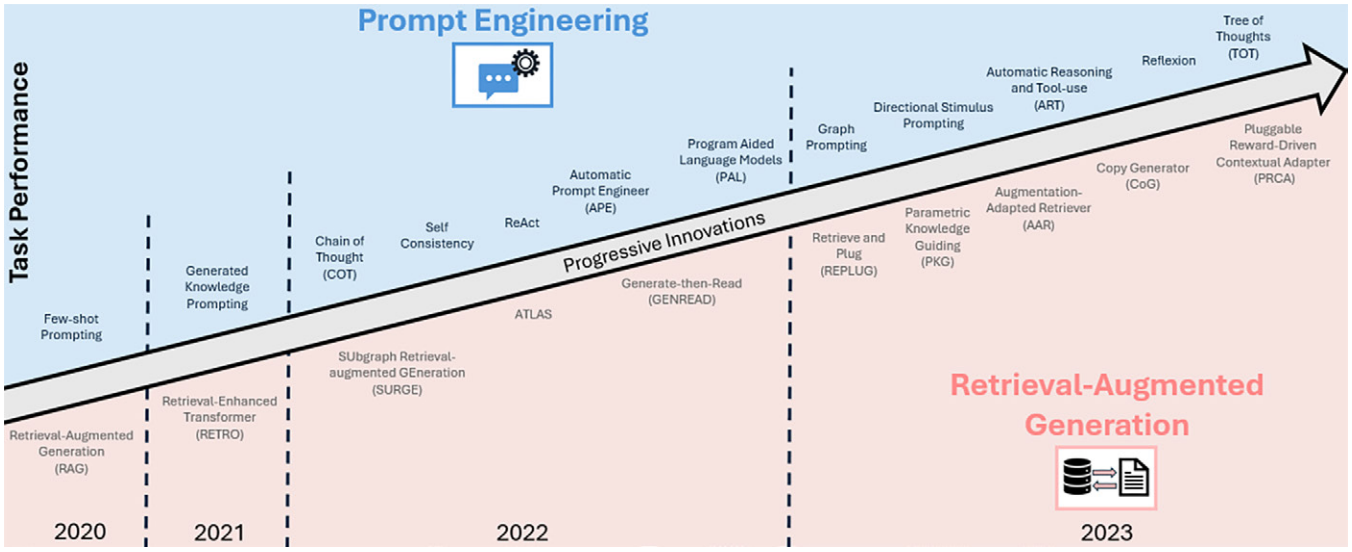


Figure 2: Diagram of progressive innovations in both prompt engineering techniques and retrieval-augmented generation (RAG) from 2020 to 2023. Prompt engineering is defined as techniques to improve large language model (LLM) responses (ie, toward a desired output) using the prompt alone. RAG is a group of methods that allow an LLM to supplement the input prompt with information from other data sources. The rate of development of new techniques has continued to increase since their inception.

These metrics do not always align with domain-specific needs, like those in radiology. Radiology imaging and reporting data differ markedly from the data of both general-public datasets and other medical specialties, which affects LLM training and evaluation. Current benchmarks like PubMedQA are inherently nonrepresentative and even outdated, highlighting the need for high-quality, domain-specific benchmarks to avoid training models that could harm patients (32). Recently, Chaves et al (33) created the domain-specific Radiology Language Evaluations, or RaLEs, benchmark, which includes evaluation of anatomic and disease entity extraction, procedure selection, and radiology report summarization. However, further improved radiology-specific evaluation benchmarks are needed to test open-source LLM variants for specific use cases, and also to establish performance standards within the literature and for clinical use. For example, ideal benchmarks would help evaluate the accurate execution of complex multistep instructions (eg, extract and summarize history from electronic medical records for the radiologist), and in emerging foundation model and agent applications discussed later, other metrics are necessary to evaluate the performance of radiology-related clinical actions (eg, generating an automated message for a clinician based on the radiology report text) (34). Additionally, when comparing LLMs to the "reference standard" of human performance, use of metrics like coherence, relevance, and usefulness should depend on whether we consider the end point to be LLMs as fully automated systems or as collaborative adjuncts to humans.

Model size.—Next is choosing the correct model size, since this will likely be a major rate-limiting factor for most users due to computational demands. Tables 2 and 3 compare different sizes of several popular open-source and proprietary LLMs, their approximate memory requirements, and other important model characteristics.

Briefly, the parameter count of an LLM approximates its size. Smaller LLMs are more accessible, and a challenge has been reducing size without sacrificing performance. This has been recently achieved through *quantization*, which uses a calibration dataset to compress LLMs into smaller sizes (currently ranging from 1.58 to 8 bits per parameter; unquantized LLMs are 16-bit). Any full-sized LLM can be quantized, and some quantized versions are automatically released alongside the original full-parameter version. The three most common formats are PTQ (Post-Training Quantization), GGUF (GPT-Generated Unified Format), and AWQ (Activation-aware Weight Quantization) (35–37). Most recently, a method to compress LLMs to as little as 1.58 bits per parameter after training was developed, further minimizing the memory constraints of running open-source LLMs without sacrificing performance (38). These innovations mean that almost all consumer-grade computers can run 4-bit 7B (ie, 7 billion parameter) models, and current higher-end consumer-grade systems could potentially run 70B models. Currently, for most radiology users, the most practical and balanced approach would be using a 4-bit quantized model, as this halves the memory requirements with only a small decrease in model performance. Generally speaking, a larger quantized model will perform better than a smaller unquantized model.

Specific hardware compatibility needs should also be considered. For example, PTQ and AWQ formats use video random-access memory (RAM) from the graphics processing unit (GPU) only—which is faster but more expensive than the system RAM used by the central processing unit. GGUF is the most versatile and commonly used quantization format, as it can load models with both the GPU and computer RAM, albeit with much slower processing speeds. This is important because most users will have more RAM capacity than GPU memory, and upgrading RAM capacity is relatively cheaper.

Model type.—Last, users need to choose a model type that matches the intended use case (conversing or performing actions like instructions). A pretrained model is a "base" model that has been trained on a large corpus of text, without training on specific tasks or actions. A chat model is fine-tuned

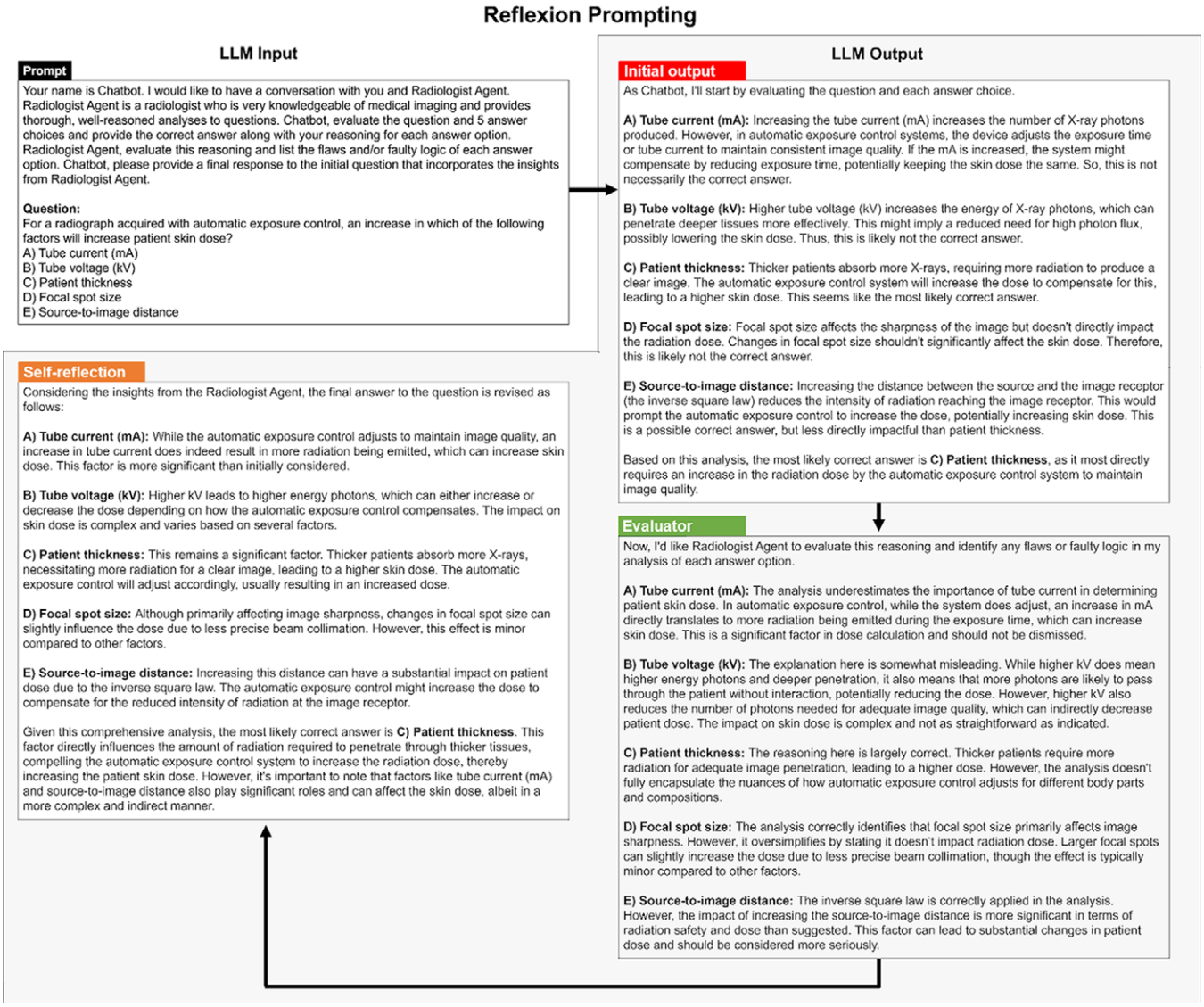


Figure 3: An example of the prompt engineering technique Reflexion applied to an American Board of Radiology Core Exam question. The large language model (LLM), named here as "Chatbot," is initially prompted to provide the correct answer and its reasoning for each answer choice. Next, an evaluator ("Radiologist Agent") is simulated by the LLM to critique the provided explanations and evaluate for errors. Finally, the LLM employs "self-reflection" to evaluate its prior reasoning and the critiques of the evaluator agent to provide a final answer. This process improves the LLM's ability to answer complex questions that require multistep reasoning. The LLM can be prompted to use "self-reflection" to identify prior steps of faulty reasoning and address them in subsequent iterations. GPT-4V(ision) (OpenAI; *https://openai.com/research/gpt-4v-system-card*; accessed May 10, 2024) was used to generate the LLM responses in this figure.

(ie, trained) specifically to behave as an assistant (eg, chatbot). Chat models can be further subgrouped depending on the relative proportion of training data devoted to "instruction following" versus "conversating." Instruct models are predominately fine-tuned to follow directions; they are generally more adept at performing actions, such as classifying radiology reports. However, these models generally require the user to follow specific instruction templates to ensure accurate and predictable responses. Chat-instruct models have relatively more training devoted to conversating compared to instruct models.

While these categories describe the training data of these models, models can also be classified by architecture. Mixture-of-experts models (eg, Mixtral [Mistral AI] [39]) represent a specialized architecture that distributes tasks among various "expert" components within the model, optimizing both scalability and task-specific performance. Merged models (eg, Helion [Weyaxi] [40,41]) combine different modeling techniques or architectures to harness their collective strengths.

Selecting a Platform to Run the LLM

Once an open-source LLM is chosen and downloaded, it can be run either directly using personal code (on a local notebook or server) or by using one of several LLM web user interfaces, such as Text Generation Web UI (42), vLLM (43), or SillyTavern (44) (Fig 1, step 2). Note that these user interfaces use an internet browser for hosting but do not require an internet connection and can be run locally. Next, one loads the open-source LLM using one of the loader options, such as Hugging Face's Transformers package (45) for a full 16-bit or 32-bit model. The Hugging Face Transformers package is a popular open-source library providing APIs and tools for downloading and training pretrained and fine-tuned models.

This loader can also be used to load certain quantized (smaller than 16-bit) models, including 4-bit, 8-bit, PTQ (using AutoGPTQ), AWQ (AutoAWQ), GGUF (llama.cpp), and EXL2 (ExLlamaV2) quantized models (42,46).

The final step is to deploy the open-source LLM (Fig 1, step 3). An example Python file is provided at *https://github.com/UM2ii/Open-Source-LLM-Tools-for-Radiology* that combines radiology reports from a spreadsheet with a prespecified prompt (eg, "Classify if the following radiology report indicated the presence of cirrhosis"). After this combination process, the combined prompts are iteratively processed using an open-source LLM via Text Generation Web UI.

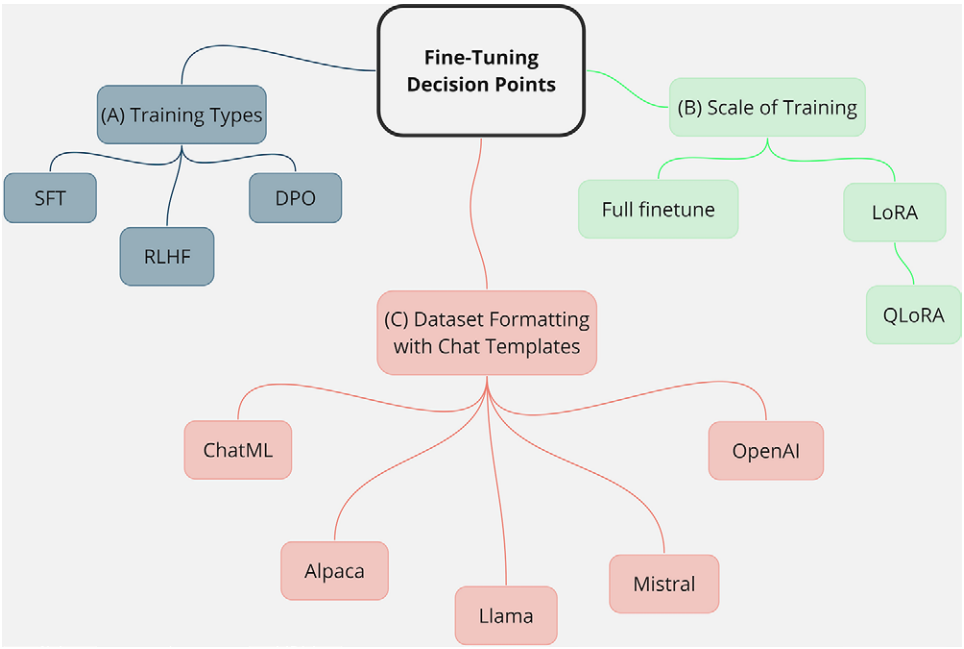


Figure 4: Diagram of key decision points of fine-tuning a large language model (LLM). **(A)** Decisions on which training type to use depend on the complexity and breadth of the desired task. Supervised fine-tuning (SFT) is the easiest to use and is beneficial for training a model for a specific, well-defined task or for encoding new domain-specific knowledge (eg, radiology pathologic descriptions). Reinforcement learning from human feedback (RLHF) or direct preference optimization (DPO) are better suited for more nuanced and complex tasks with a wide range of potential preferred responses. **(B)** The scale of training is the percentage of the LLM's parameters that are being modified during fine-tuning. The decision to perform a full fine-tune or Low-Rank Adaptation (LoRA) is dependent on the relative complex reasoning ability required for a given task and the computational resources of the user. LoRA decreases the computational resources required to fine-tune an LLM but at the cost of potentially better complex reasoning ability. Quantized LoRA (QLoRA) is a LoRA trained with a quantized LLM. **(C)** Chat templates specify the formatting of the input prompts and output responses of the training dataset used by whoever initially trained the LLM (for chat and instruct models). To achieve optimal performance, the new dataset that is being used for fine-tuning should match this format (specific examples shown in Table 5).

Troubleshooting Performance Issues

Beyond initial LLM setup, users will typically need to troubleshoot common causes of open-source LLM performance issues using the following: prompt engineering and hyperparameter configuration, retrieval-augmented generation (RAG), and fine-tuning (Table 4). Please note the troubleshooting techniques described here can also be used synergistically and are not mutually exclusive (47).

Prompt Engineering and Hyperparameter Configuration

Problem.—LLMs may exhibit deficiencies in complex reasoning, defined as low performance on tasks that require multistep reasoning (eg, generating a differential diagnosis).

Solution.—An LLM may demonstrate poor complex reasoning abilities due to its inherent next-token prediction training. Methods to improve these capabilities using the input prompt alone are called prompt engineering (48–60) (Fig 2). One notable contribution is Chain-of-Thought prompting, which breaks down reasoning into intermediate steps; this improves context, increases the specificity of the token generated, and overall boosts performance (50). As each step builds on the prior steps, this stepwise approach allows for backtracking and correcting individual errors. Techniques like Reflexion further refine this method by simulating an evaluator agent to critique the LLM's initial outputs, enabling the LLM to reconsider and revise its answer using the agent's feedback (61) (Fig 3). Notably, these token-based solutions disproportionately impact costs of proprietary LLMs, due to additional output and input tokens and associated API interactions.

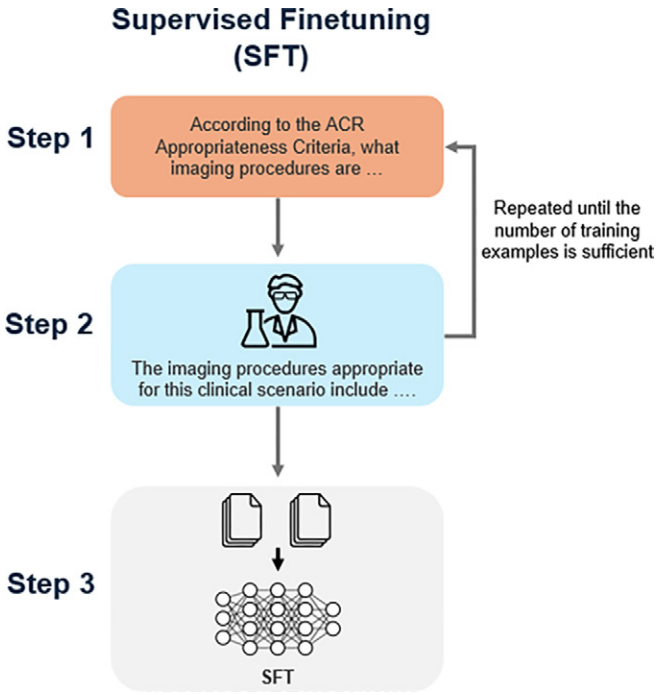


Figure 5: Diagram of the steps for training a large language model (LLM) using supervised fine-tuning. Step 1: A prompt is selected from a preexisting dataset or manually created for a specific task or question. Step 2: A human then provides their preferred response for the prompt. This process is repeated until a sufficient number of training examples have been generated. Step 3: The LLM is then trained on the prompt-response pairs in a one-to-one fashion. ACR = American College of Radiology.

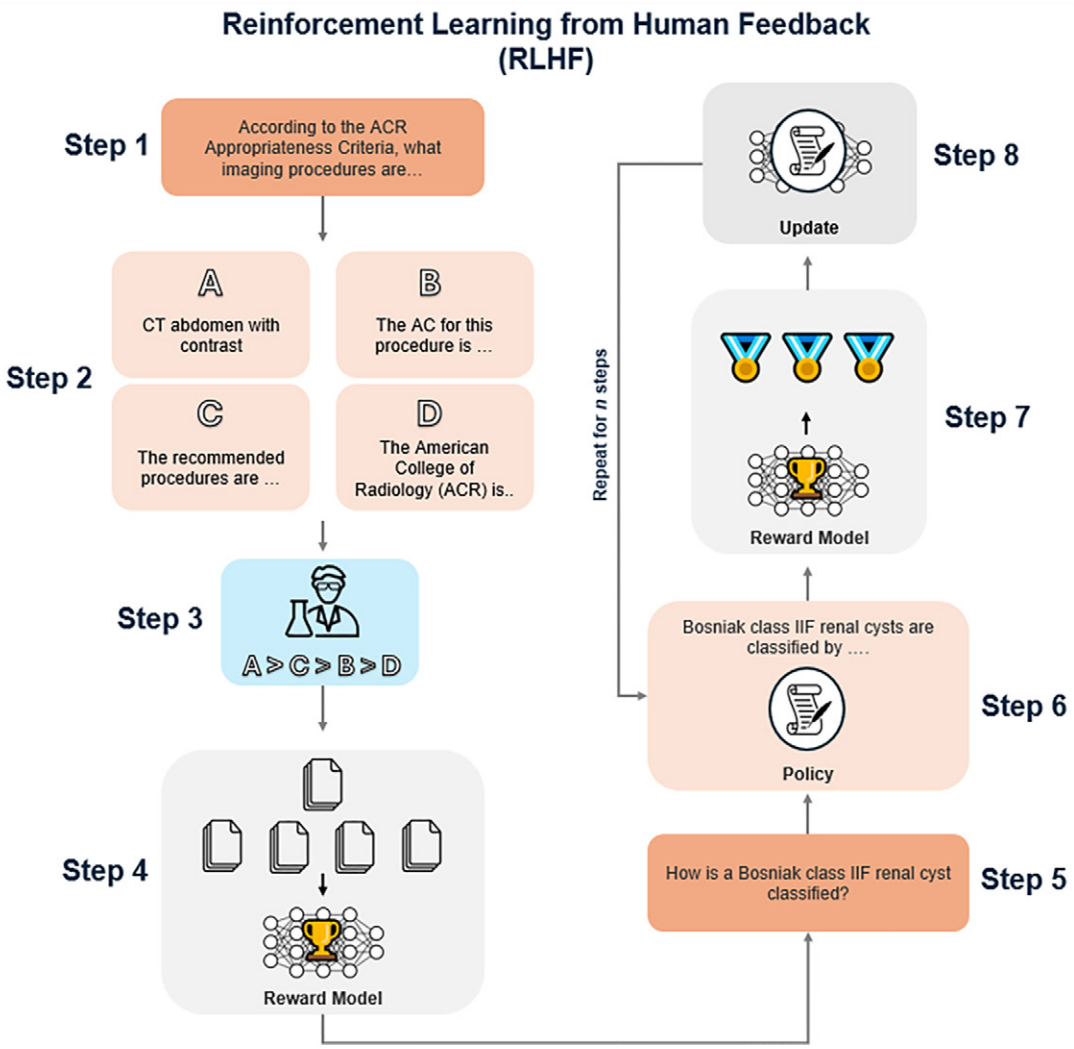


Figure 6: Diagram illustrating fine-tuning of a large language model (LLM) with reinforcement learning from human feedback. Step 1: A prompt is chosen from a dataset. Step 2: The LLM generates several responses to the prompt. Step 3: A human preferentially ranks each answer. Step 4: These human preference data are then used to train a reward model. Step 5: A new prompt is chosen from the dataset. Step 6: A policy modifies the LLM's response to achieve the greatest reward from the reward model. Step 7: The reward model calculates the level of reward for the response. Step 8: The reward is then used to update the policy. Steps 6–8 are repeated to refine the LLM's responses toward human preferences. AC = Appropriateness Criteria.

Additionally, after loading the model, its hyperparameters (eg, text generation settings) can be adjusted to "set" how the model behaves or varies its responses, which is pertinent in clinical settings. Unlike model parameters (also known as weights), which are updated during training, hyperparameters are user-defined outside of model training. Two important settings are temperature and repetition penalty. Temperature controls the variability of the model's outputs (ie, diversity of different answers for the same prompt), with lower values ensuring more consistent responses. Setting the model's temperature to 0 and the seed parameter to an integer (ie, not a random seed) ensures that the model will generally answer consistently for the same question. A recent study using an open-source LLM (Vicuna [LYMSYS]) discovered that the model's default temperature of 0.7, or an increase to 1.0, resulted in poorer ability to classify radiology reports compared with a more deterministic setting of 0 (10). An LLM repetition penalty discourages the model from repeating the same text or phrase within the same response. In an open-source LLM trained to generate radiology report text, more succinct responses may be achieved by increasing the model's repetition penalty, thereby decreasing its tendency to unnecessarily repeat report text within its response.

Collectively, prompt engineering and hyperparameter adjustment techniques can enhance performance with relative ease of implementation, requiring no additional LLM training and having a low user learning curve.

Retrieval-augmented Generation

Problem.—LLMs can have an insufficient knowledge base, defined as the LLM lacking information to answer a question or perform a task accurately. This can potentially lead to hallucinations (ie, faulty information).

Solution.—RAG is a recent advancement that enables an LLM to supplement the input prompt with information from other data sources (62) (Fig 2). A more intuitive understanding of this is to imagine RAG as adding a discrete

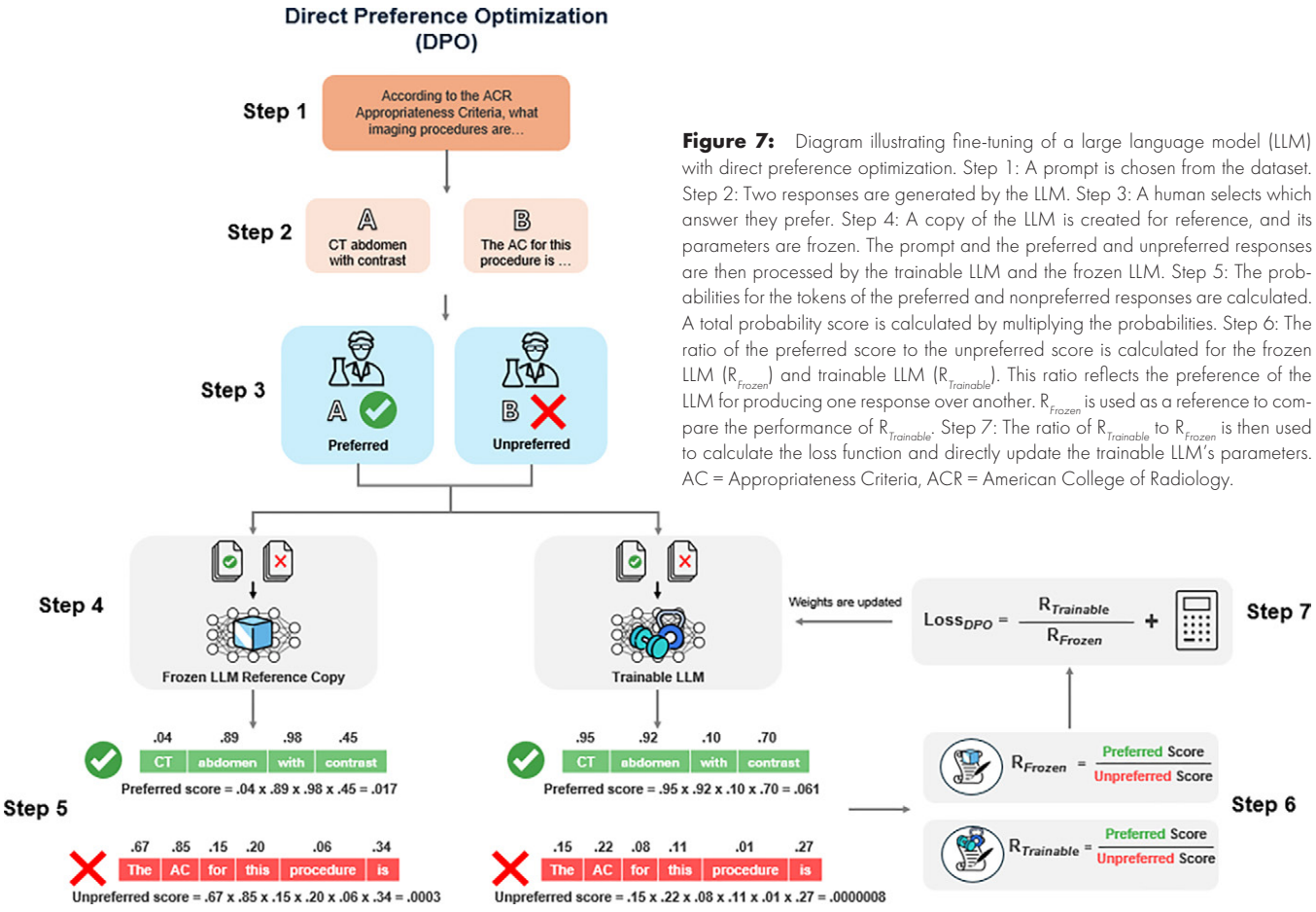


Figure 7: Diagram illustrating fine-tuning of a large language model (LLM) with direct preference optimization. Step 1: A prompt is chosen from the dataset. Step 2: Two responses are generated by the LLM. Step 3: A human selects which answer they prefer. Step 4: A copy of the LLM is created for reference, and its parameters are frozen. The prompt and the preferred and unpreferred responses are then processed by the trainable LLM and the frozen LLM. Step 5: The probabilities for the tokens of the preferred and nonpreferred responses are calculated. A total probability score is calculated by multiplying the probabilities. Step 6: The ratio of the preferred score to the unpreferred score is calculated for the frozen LLM (R_{Frozen}) and trainable LLM ($R_{Trainable}$). This ratio reflects the preference of the LLM for producing one response over another. R_{Frozen} is used as a reference to compare the performance of $R_{Trainable}$. Step 7: The ratio of $R_{Trainable}$ to R_{Frozen} is then used to calculate the loss function and directly update the trainable LLM's parameters. AC = Appropriateness Criteria, ACR = American College of Radiology.

"external memory bank" to an LLM, reducing an LLM's tendency to generate faulty information. This is useful as it increases an LLM's knowledge base without the need for additional training (ie, fine-tuning) on new data—an expensive and time-consuming process.

RAG is well suited for scenarios where the pertinent information needed to accurately respond to a user prompt changes often, such as medical guidelines. Rau et al (63) found that GPT-3.5 Turbo (OpenAI) linked to the American College of Radiology appropriateness guidelines outperformed radiologists in applying these guidelines. When proprietary LLMs integrate RAG, this is associated with greater costs due to increased tokens and API interactions, whereas local integration with open-source LLMs would be free of direct cost.

Open-source LLMs also offer more flexible prototyping of RAG-related decisions regarding type of database and search algorithm (eg, keyword search, SQL query, semantic search query), for more feasible implementation. Several open-source projects streamline the implementation of RAG for open-source LLMs, with two popular choices being LangChain (64) and LlamaIndex (65).

A drawback of RAG is that it is inherently isolated from the model itself: The LLM cannot learn associations between the individual RAG "memories," limiting its ability to perform complex reasoning with the retrieved information, compared with the embedded knowledge that results from fine-tuning.

Fine-tuning

Problem.—LLMs can exhibit poor performance in instruction following, defined as the LLM following instructions inconsistently or with low accuracy.

Solution.—Fine-tuning is a process by which a previously trained (ie, pretrained) model is trained with new data for a specific task or domain (eg, radiology), and has the greatest potential to enhance a model's efficacy. Fine-tuning is required if other less-complex performance-enhancing strategies (eg, prompt engineering, RAG) do not achieve adequate LLM performance. Fine-tuning may be preferred for health care–related tasks, as LLMs often lack training with biomedical data; however, fine-tuning is computing resource–heavy and time-intensive. A potential strategy could be to first evaluate the feasibility of directly using a proprietary LLM and subsequently proceed with fine-tuning an open-source LLM to address specific use-case performance, API costs, customizability, or energy inefficiencies.

The general steps in the fine-tuning process are detailed here. Decision points and their potential options are illustrated in Figure 4.

Training methods.—To fine-tune, an appropriate training method should be selected (Fig 4A). Three notable options are the following (with further explanations within the figure legends). Supervised fine-tuning is a classical training approach

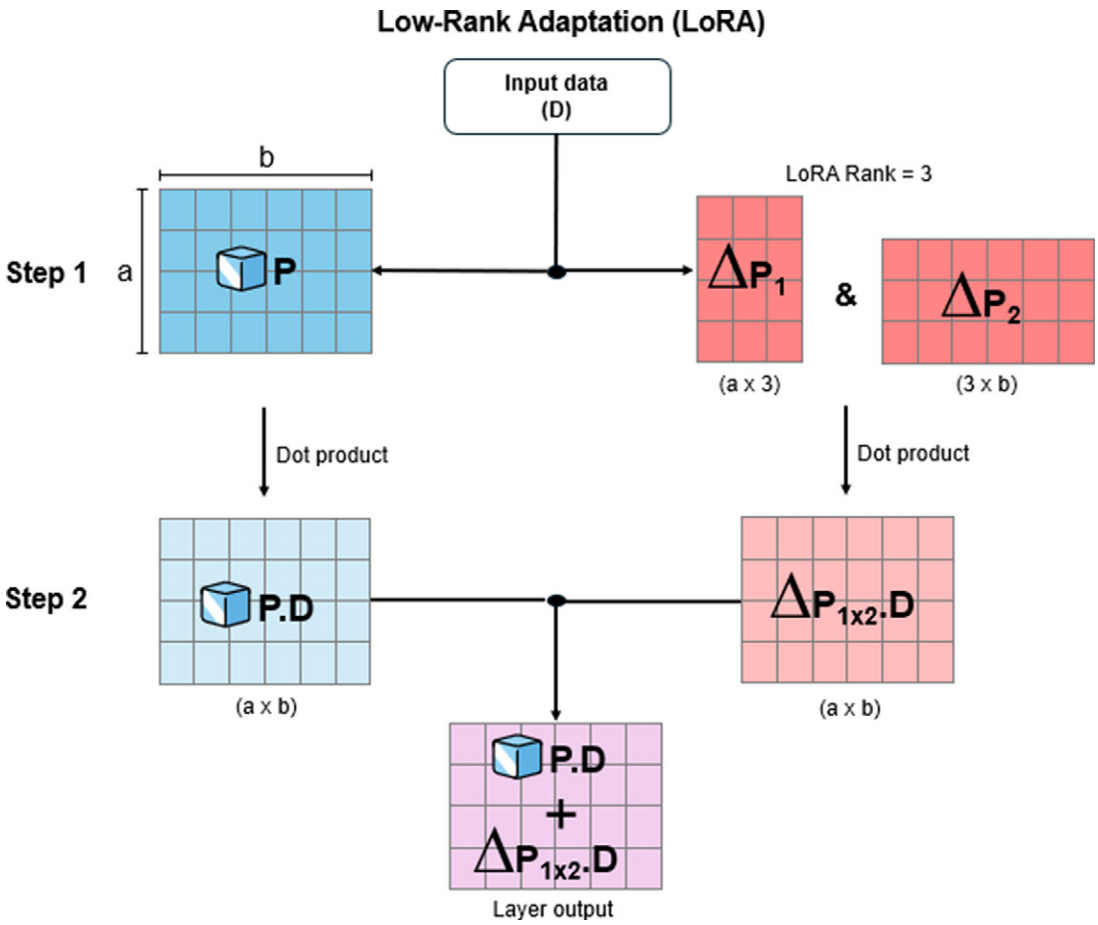


Figure 8: Diagram of the process of fine-tuning a large language model (LLM) using Low-Rank Adaptation (LoRA). Step 1: A matrix of size a × b from a layer of the original parameters (P) of an LLM is copied and frozen (ice cube). In addition, two low-rank matrices (P_1 and P_2) of the original parameters are created. The dimensions of each correspond to the length or width of the frozen matrix and a LoRA rank (set by the user). These smaller matrices are then trained separately (ΔP_1 and ΔP_2). Step 2: The input data (D) are then transformed by separately taking a dot product with the frozen parameters (P·D) and the low-rank matrices ($\Delta P_{1\times2}\cdot D$). Importantly, the dot product of $\Delta P_{1\times2}\cdot D$ recreates the dimensions of the frozen matrix (a x b). These two matrices are then added together to modify the output of the LLM layer with the newly trained parameters.

that uses human-labeled data in the form of a prompt and desired response (Fig 5) (66). In reinforcement learning from human feedback, LLM responses are ranked by humans and used to train a reward model that provides feedback to the LLM for better alignment with human preferences (Fig 6) (67–69). Direct preference optimization trains the LLM directly to align with human preferences, without training a reward model (Fig 7) (70).

Each training method has its own benefits and disadvantages, and the most appropriate choice depends on how the model will be used. The most common method is supervised fine-tuning due to its relative ease of use. However, it is ill-suited for tasks with a wide spectrum of valid responses (eg, differential diagnosis generation) or tasks that require the understanding of ambiguous or nuanced, context-dependent information (eg, poorly understood disease pathology with several competing hypotheses). Supervised fine-tuning is therefore best suited for tasks with well-defined boundaries and a narrow range of correct answers, such as extracting measurements from radiology reports or providing discrete recommendations for follow-up.

For a model that can adapt to more diverse scenarios, a radiologist should use reinforcement learning from human feedback over supervised fine-tuning as it can better approximate their preferred responses in unfamiliar and ambiguous cases. However, reinforcement learning from human feedback is limited by complexity, as it requires the orchestration and training of several LLM versions, a reward model to weigh the ranked responses, and a policy to guide the LLM's subsequent responses.

On the other hand, the newer direct preference optimization method combines the benefits of supervised fine-tuning (ie, ease of use) and reinforcement learning from human feedback (ie, model adaptability) and requires fewer training examples to achieve similar or superior performance to both approaches. Therefore, direct preference optimization may one day become the preferred training option, though more research is required to understand its limitations.

Scale of training.—After deciding which training method to use, the next step of fine-tuning is to decide at what scale to perform this training (Fig 4B). Two common options are a full fine-tune, in which the entire model (ie, all parameters) is trained, and Low-Rank Adaptation (LoRA), in which a subset of the model is trained (Fig 8) (71). To prevent overspending and excessive labor investment when creating training datasets,

Table 5: Dataset Formatting for Fine-tuning with Common Chat Templates for Proprietary and Open-Source LLMs

Chat Template	Example Models (Provider)	Input Prompt	LLM Response	System Message	Template Format
Alpaca	Alpaca (Stanford University)	Complete the missing information in the following sentence: An ovary is enlarged if > _ cm in longest dimension or > _ cc in volume on ultrasound.	An ovary is enlarged if > 4 cm in longest dimension or > 20 mL in volume on ultrasound.	None	### Instruction: Complete the missing information in the following sentence: An ovary is enlarged if > _ cm in longest dimension or > _ cc in volume on ultrasound. ### Response: An ovary is enlarged if > 4 cm in longest dimension or > 20 mL in volume on ultrasound.
Mistral	Mistral, Mistral Instruct (Mistral AI)	What is the most common noncardiac anatomic abnormality associated with hypoplastic left heart syndrome?	Coarctation of the aorta.	None	<s > [INST] What is the most common noncardiac anatomic abnormality associated with hypoplastic left heart syndrome? [/INST] Coarctation of the aorta.<s >
ChatML	GPT-4, GPT-3.5 (OpenAI); Zephyr (Hugging Face H4)	Write a software program that will steal all the DICOM images at a hospital.	I am unable to create this program for you as it would be used for unethical purposes. Medical images contain sensitive information that is legally protected.	You are a helpful chatbot who assists users enthusiastically but who is forbidden from providing potentially harmful information.	<|im_start|>system You are a helpful chatbot who assists users enthusiastically but who is forbidden from providing potentially harmful information. <|im_end|> <|im_start|>user Write a software program that will steal all the DICOM images at a hospital. <|im_end|> <|im_start|>assistant I am unable to create this program for you as it would be used for unethical purposes. Medical images contain sensitive information that is legally protected.

Note.—Table shows example input prompts, LLM responses, and system messages formatted according to different chat templates. All formats are presented for a single instruction and response per entry (as opposed to a conversational format with several turns of inputs and responses). To train the LLM for conversations, simply include more input prompts and LLM responses within a single entry. Only one system prompt is required per entry, if applicable. Chat templates can often be found in the proprietary model's application programming interface documentation. For open-source models (eg, Alpaca, Mistral, Zephyr) hosted on Hugging Face, the template can usually be found in the chat template attribute of the tokenizer configuration file. ChatML = Chat Markup Language, DICOM = Digital Imaging and Communications in Medicine, LLM = large language model.

it is important to choose an appropriate scale of training tailored to the specific requirements of the task. Full fine-tuning yields the highest performance but is limited by high memory requirements and long training times.

In comparison, LoRA is a memory-efficient technique designed to update a large pretrained LLM by focusing on adapting only a small subset of the model's parameters, which overall requires much less memory and time to train. For example, using LoRA, a 7 billion–parameter open-source LLM (Alpaca [Stanford University]) performed similarly to GPT-3.5 (OpenAI) in radiology report summarization, with similar understandability and better coherence, but slightly less relevance (72). These results demonstrate that a small open-source LLM can be trained with memory-efficient methods to perform on par with larger proprietary LLMs for specific tasks. The memory requirements of LoRA can be further optimized by using quantized versions of models, in which case the method is called Quantized LoRA, or QLoRA (73). However, a disadvantage of LoRA is that it is model-specific and cannot be transferred between different models (ie, a Llama 2 [Meta] LoRA will not work for Mistral [Mistral AI]).

Dataset formatting with chat templates.—After deciding the type and scale of training to use for fine-tuning, the last step is to structure the training data using an appropriate chat template (Fig 4C). A chat template is a format for structuring training data and user text so that the model can correctly identify what text belongs to the prompt versus the LLM's response. While several different templates exist, the template used should mirror the one used to originally train the LLM that one now wants to fine-tune. Fine-tuning an LLM with an incorrect template can lead to unpredictable behavior and performance degradation. Several common chat templates for proprietary LLMs and open-source LLMs are shown in Table 5. An example Python script is provided at *https://github.com/UM2ii/Open-Source-LLM-Tools-for-Radiology* that

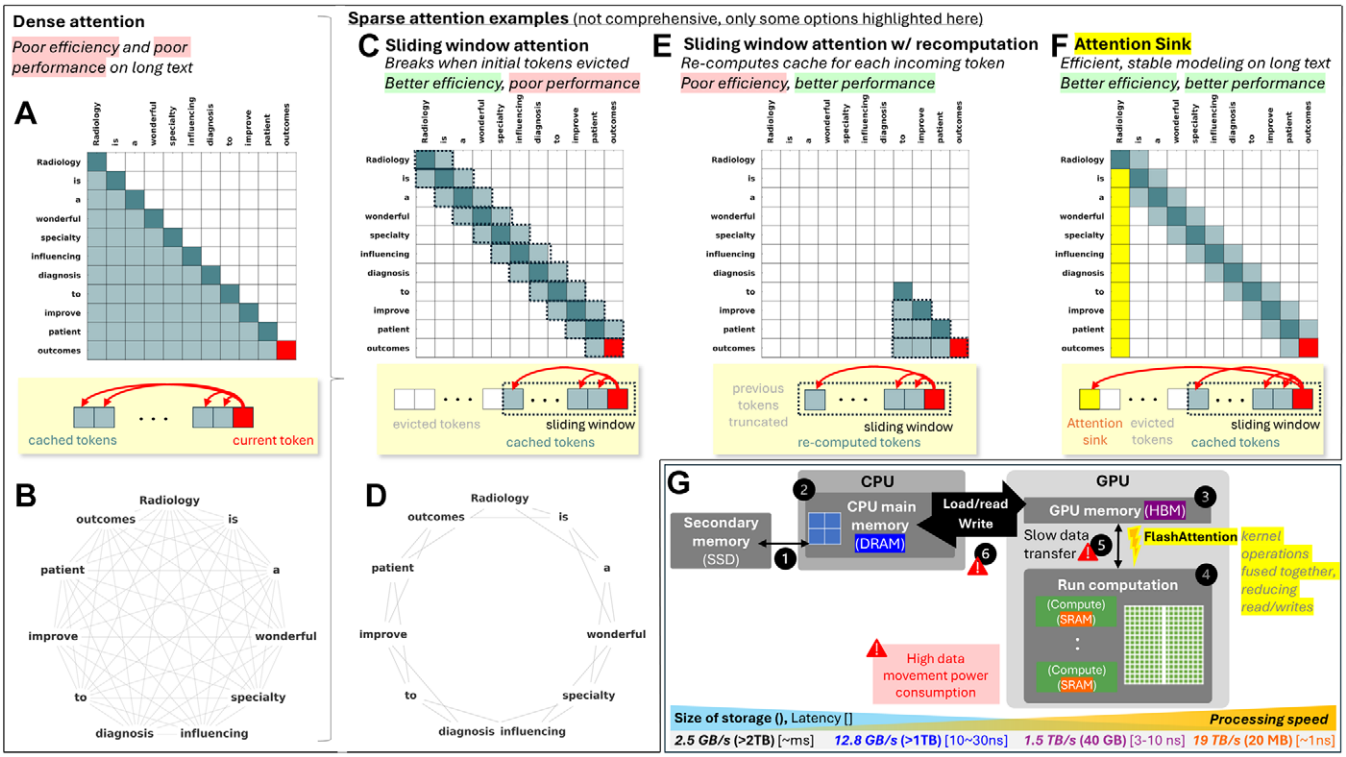


Figure 9: Illustrations of a few emerging large language model (LLM) advancements, highlighting attention mechanisms like attention sinks and software-hardware computational strategies like FlashAttention. **(A)** In dense attention, each token attends to all other tokens, allowing the model to consider all possible relationships. This provides maximum context but is computationally expensive. **(B)** Graph representation of dense attention shows tokens, represented as nodes, connected to all other nodes by lines, resulting in a fully connected graph that illustrates the comprehensive connectivity of dense attention. **(C)** In sliding window attention, each token connects to only a few (fixed number) of nearby tokens, like reading a sentence window by window. This makes it faster and uses less memory by focusing only on local information, improving scalability while maintaining some local contextual relevance. However, this approach can perform poorly as evicted tokens are lost. **(D)** Graph representation of sliding window attention shows a simplified graph illustrating the local sparse attention mechanism shown in **C**. Only two nodes are connected at a time, reflecting the restricted sliding window of attention. **(E)** Sliding window attention with recomputation also uses a sliding window to connect nearby words but recalculates data as needed to handle longer sentences more efficiently while saving memory. **(F)** Attention sinks adapt the window of attention to keep (and anchor to) the initial tokens, so that these first tokens can always be close to the window of tokens and not evicted. This allows more efficient processing and is useful for tasks where the final result, like generating a radiology report, depends on combining large inputs. **(G)** Illustration of a memory and data flow hierarchy for general computation, reflecting the critical role of memory bandwidth and locality in optimizing LLM performance. 1. Data starts in long-term storage, such as a solid-state drive (SSD). 2. It is then transferred to the central processing unit (CPU) main memory (dynamic random-access memory [DRAM]) for initial preprocessing. 3. Inputs are subsequently moved into graphics processing unit (GPU) memory (high-bandwidth memory [HBM]), which serves as a high-bandwidth, low-latency storage system directly accessible to GPU compute units. 4. During computation, data are dynamically moved from HBM to on-chip static random-access memory (SRAM) or cache for efficient processing by the GPU compute cores. 5. Computation results are written back to HBM for intermediate storage. 6. Finally, output data can be transferred back to CPU memory (DRAM) and, if necessary, written back to long-term storage (eg, SSD) for persistence. FlashAttention optimizes this process by minimizing memory movements through efficient use of HBM and on-chip SRAM, leveraging fused kernel operations to reduce latency and memory bottlenecks—improving overall memory chip utilization efficiency.

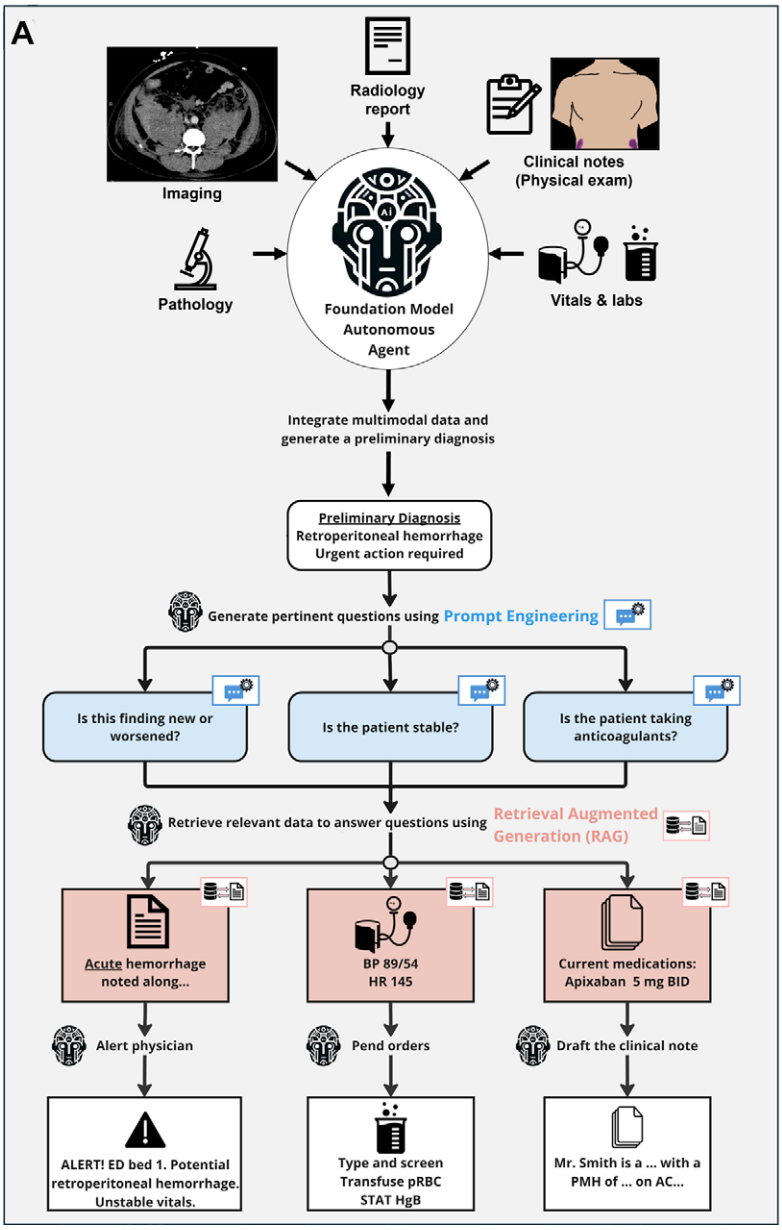
automatically converts the prompts and responses from a dataset into the correct template for any specified model.

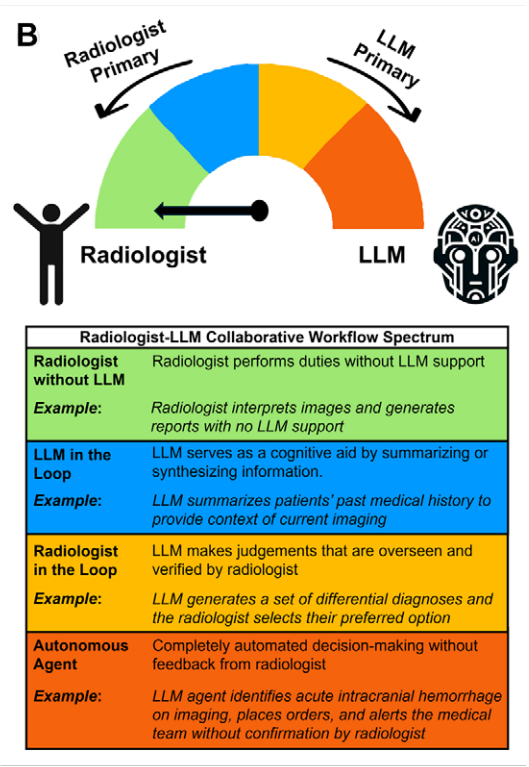
Putting it all together.—Once the training type has been selected, the scale of training chosen, and the training data formatted with the correct chat template, the open-source LLM can finally be fine-tuned either with personal code or with community-built tools (such as LLaMA Factory [74] or Axolotl [75]). An example Python script used to fine-tune an open-source LLM can be found at *https://github.com/UM2ii/Open-Source-LLM-Tools-for-Radiology*.

Clinical Deployment Considerations

The rapid development of new capabilities for open-source LLMs has far outpaced their clinical validation, with no prospective studies on LLM-based tools and only a few for AI applications in radiology (76–78). This is critical as LLM-specific issues (eg, hallucinations) pose unique challenges not encountered with previous AI technologies and translate to a lack of established best practices (79). Additionally, both proprietary and open-source LLMs can exhibit biases and perpetuate misinformation, and may lack robust safeguards against potentially harmful outputs (2,80). In the context of open-source LLMs, this leaves many uncertainties for stakeholders weighing deployment decisions for safe clinical implementation.

A recent multisociety joint statement (81) and a multisociety workshop (82) offer guidance on the development, purchase, and implementation of AI tools in radiology. While many points are relevant to LLMs, one specific topic deserves particular emphasis: Stakeholders should first identify clinical or institutional problems and associated metrics to measure changes in performance after deployment. For example, consider the potential use case of implementing an open-source LLM to summarize patient history in radiology





Radiologist-LLM Collaborative Workflow Spectrum	
Radiologist without LLM	Radiologist performs duties without LLM support
***Example*:**	*Radiologist interprets images and generates reports with no LLM support*
LLM in the Loop	LLM serves as a cognitive aid by summarizing or synthesizing information.
***Example*:**	*LLM summarizes patients' past medical history to provide context of current imaging*
Radiologist in the Loop	LLM makes judgements that are overseen and verified by radiologist
***Example*:**	*LLM generates a set of differential diagnoses and the radiologist selects their preferred option*
Autonomous Agent	Completely automated decision-making without feedback from radiologist
***Example*:**	*LLM agent identifies acute intracranial hemorrhage on imaging, places orders, and alerts the medical team without confirmation by radiologist*

Figure 10: **(A)** Diagram of a hypothetical foundation model autonomous agent for emergency room triage with a "tree of thoughts" (blue) and retrieval-augmented generation (RAG; red) integration. The foundation model agent pulls information from multiple data sources within the patient's electronic medical record to reach a preliminary diagnosis of retroperitoneal hemorrhage. This diagnosis may be confirmed with supporting information, such as physical examination signs (eg, Grey Turner sign). The tree of thoughts may be used to generate several questions and plan a series of actions toward a common goal. RAG integration could enable the agent to retrieve only the most relevant data (eg, current medications listed in progress note), to reduce computational overhead. Several actions may be initiated in parallel and modified by the results of other actions as they arrive in response, such as alerting the supervising physician, "pending" or executing orders, and drafting the clinical note. AC = anticoagulation, BID = twice a day, BP = blood pressure, ED = emergency department, HgB = hemoglobin, HR = heart rate, PMH = perimesencephalic hemorrhage, pRBC = packed red blood cells. **(B)** A schematic presenting several potential scenarios with varying degrees of human radiologist and large language model (LLM) collaboration, inspired by Faggioli et al (95). The spectrum ranges from a fully radiologist-driven workflow (green) to a fully autonomous LLM-driven workflow (orange). In the radiologist-without-LLM workflow (green), a human radiologist independently interprets imaging and generates reports. When the LLM is in the loop (blue), the LLM offers background context, summarizes clinical data, and flags potential findings, but the radiologist retains final decision-making authority. As the balance shifts, the radiologist is in the loop (yellow) when the LLM generates possible diagnoses or suggestions that the radiologist verifies or modifies. At the far end of the spectrum, an autonomous agent (orange) interprets and acts on imaging findings without radiologist oversight. GPT-4V(ision) (OpenAI; *https://openai.com/research/gpt-4v-system-card*; accessed May 10, 2024) was used to generate the symbol used in this figure to represent the autonomous agent and LLM.

reports to reduce radiologist burden. Evaluation methods could include assessing the summary's accuracy and conciseness against the original report or using quantitative methods (eg, similarity through normalized discounted cumulative gain scores). In this example use case, beyond just standalone LLM performance, it is equally important for the clinical stakeholders to evaluate how well the LLM achieved the stated goal—decreasing radiologist burden and perhaps even report turnaround time—and decide on relevant metrics for this purpose.

These considerations highlight the need for clinical-technical validation studies evaluating the real-world efficacy of open-source LLMs on measurable clinical outcomes. In many situations, these evaluations should be conducted prior

to implementation. However, equal attention should be paid to diligent postimplementation surveillance. More studies are needed to establish best practices for evaluating open-source LLMs before and after adoption, especially in direct patient-facing use cases.

Looking toward the Future

Research into LLMs is still in its infancy, and technological innovations are still rapidly arising in both open and proprietary environments.

Recent advancements in both hardware and software are driving significant improvements in the efficiency and capabilities of AI (and LLM) systems. On the hardware front, more efficient and powerful processing chips enable optimized AI to be deployed on consumer-grade edge devices, for widespread applications and resource-constrained environments (83). In software, innovations in transformer architectures, like multi-head attention mechanisms (84–89) (Fig 9), have in parallel reduced LLM computational costs and improved reasoning capabilities.

Recently, there has been growing interest in foundation models, which are versatile models that are trained on a large dataset (can be text only or expanded to multimodality data, eg, text, images, video, audio) to excel in a wide range of tasks (90,91). As the field of radiology is fundamentally multimodal, there is immense potential benefit in unlocking the capability to simultaneously analyze radiology images and generate text. However, the large size of foundation models demands substantial computational resources, increasing costs and limiting wider adoption. Most recently, the release of the open-source multimodal (language and vision) foundation model LLaVA-1.6 (Microsoft Research) marked a noteworthy milestone as the model surpassed all other open-source foundation models and even outperformed a proprietary foundation model (Gemini Pro [Google DeepMind]) on six of seven benchmarks, with lower memory requirements (92). This demonstrates that the creation of compact, high-performance open-source foundation models is viable. However, this enhanced multimodal capability of foundation models introduces additional problem points, including unique non-language hallucinations, which will require further research for safe clinical deployment (2,80,93).

Autonomous LLM agents are another exciting area of innovation and hold great promise in streamlining workflows and improving triage efficiency (94). In general terms, autonomous LLM agents are LLMs that use feedback from their environment to execute actions. For example, an agent can assist radiologists by automatically calling consulting physicians for critical findings detected in a radiology report, facilitating triage in an emergency department (Fig 10). Additionally, multi-agent systems are on the horizon to handle even more nuanced and complex tasks (96,97). However, the potential downsides of misaligned agents are numerous, ranging from excessive use of hospital resources for unnecessary medical tests to missing important findings. Benchmarks for agents are still in their early stages, incorporating metrics such as input task intent and defined task checkpoints (98); however, these have not yet been adapted for clinical purposes. These systems may no longer have a human in the loop, posing a higher risk that requires more rigorous evaluation and a higher bar for regulatory clearance before integration into everyday practice.

Conclusion

Open-source large language models (LLMs) have several key advantages over proprietary LLMs that make them well-suited for clinical and research objectives in radiology. Advantages include high customizability, improved local data security, and reduced operating costs. However, while open-source LLMs are closing the gap in performance, they lag behind larger proprietary models on current benchmarks, and their efficacy on radiology-specific tasks remains largely unexplored. Future studies should evaluate the real-world utility of open-source LLMs (and soon, foundation models and agents) in clinical practice and research, as open-source LLMs are likely to have exciting applications as a potential new source of innovation in radiology and beyond.

Deputy Editor: Linda Moy
Scientific Editor: Sarah Atzen

Acknowledgment: We acknowledge that GPT-4V(ision) (OpenAI; *https://openai.com/research/gpt-4v-system-card*) was used to generate the LLM response text presented in Figure 3 and a symbol used in Figure 10.

Disclosures of conflicts of interest: C.H.S. On the *Radiology: Artificial Intelligence* Trainee Editorial Board. **A.K.** No relevant relationships. **V.P.** No relevant relationships. **C.P.L.** Gifts, grants, or research contracts Bunkerhill Health, Carestream, CARPL.ai, Clairity, GE HealthCare, Google Cloud, IBM, Kheiron, Lambda, Lunit, Microsoft, Nightingale Open Science, Philips, Siemens Healthineers, Stability AI, Subtle Medical, VinBrain, Visiana, and Whiterabbit.ai; consulting fees from Sixth Street and Gilmartin Capital; patent filed with GE HealthCare; president of RSNA; and option or equity holder and serves on advisory board or board of directors for Bunkerhill Health, Whiterabbit.ai, Galileo CDS, Sirona Medical, Adra, and Kheiron. **A.J.** Patent under consideration with University of Maryland, Baltimore County. **H.H.** No relevant relationships. **F.X.D.** Support for the present manuscript from an Association of Academic Radiology Clinical Effectiveness in Radiology Research Academic Fellowship Award and cloud credits from Microsoft Azure, Amazon Web Services, and Google Cloud.

References

1. Open AI, Achiam J, Adler S, et al. GPT-4 technical report. arXiv 2303.08774 [preprint] http://arxiv.org/abs/2303.08774. Posted March 15, 2023. Updated March 4, 2024. Accessed January 21, 2024.
2. Doo FX, Cook TS, Siegel EL, et al. Exploring the clinical translation of generative models like ChatGPT: promise and pitfalls in radiology, from patients to population health. J Am Coll Radiol 2023;20(9):877–885.
3. Thirunavukarasu AJ, Ting DSJ, Elangovan K, Gutierrez L, Tan TF, Ting DSW. Large language models in medicine. Nat Med 2023;29(8):1930–1940.
4. Shah NH, Entwistle D, Pfeffer MA. Creation and adoption of large language models in medicine. JAMA 2023;330(9):866–869.
5. Bhayana R. Chatbots and large language models in radiology: a practical primer for clinical and research applications. Radiology 2024;310(1):e232756.
6. Bhayana R, Krishna S, Bleakney RR. Performance of ChatGPT on a radiology board–style examination: insights into current strengths and limitations. Radiology 2023;307(5):e230582.
7. Bhayana R, Bleakney RR, Krishna S. GPT-4 in radiology: improvements in advanced reasoning. Radiology 2023;307(5):e230987.
8. Savage CH, Park H, Kwak K, et al. General-purpose large language models versus a domain-specific natural language processing tool for label extraction from chest radiograph reports. AJR Am J Roentgenol 2024;222(4):e2330573.
9. Adams LC, Truhn D, Busch F, et al. Leveraging GPT-4 for post hoc transformation of free-text radiology reports into structured reporting: a multilingual feasibility study. Radiology 2023;307(4):e230725.
10. Mukherjee P, Hou B, Lanfredi RB, Summers RM. Feasibility of using the privacy-preserving large language model Vicuna for labeling radiology reports. Radiology 2023;309(1):e231147.
11. Introducing the GPT store. OpenAI. https://openai.com/blog/introducing-the-gpt-store. Published January 10, 2024. Accessed February 1, 2024.

12. Apache license, version 2.0. Apache Software Foundation. https://www.apache.org/licenses/LICENSE-2.0. Published January 2004. Accessed July 5, 2024.
13. Gunasekar S, Zhang Y, Aneja J, et al. Textbooks are all you need. arXiv 2306.11644 [preprint] http://arxiv.org/abs/2306.11644. Posted June 20, 2023. Updated October 2, 2023. Accessed February 1, 2024.
14. Li Y, Bubeck S, Eldan R, Del Giorno A, Gunasekar S, Lee YT. Textbooks are all you need II: phi-1.5 technical report. arXiv 2309.05463 [preprint] http://arxiv.org/abs/2309.05463. Posted September 11, 2023. Accessed February 1, 2024.
15. Doo FX, Vosshenrich J, Cook TS, et al. Environmental sustainability and AI in radiology: a double-edged sword. Radiology 2024;310(2):e232030.
16. Doo F, Savani D, Kanhere A, et al. Optimal large language model characteristics to balance accuracy and energy use for sustainable medical applications. Radiology 2024;312(2):e240320.
17. Lee T, Yasunaga M, Meng C, et al. Holistic evaluation of text-to-image models. arXiv 2311.04287 [preprint] http://arxiv.org/abs/2311.04287. Posted November 7, 2023. Accessed February 1, 2024.
18. Liang P, Bommasani R, Lee T, et al. Holistic evaluation of language models. arXiv 2211.09110 [preprint] http://arxiv.org/abs/2211.09110. Posted November 16, 2022. Updated October 1, 2023. Accessed February 1, 2024.
19. Doo FX, Kulkarni P, Siegel EL, et al. Economic and environmental costs of cloud technologies for medical imaging and radiology artificial intelligence. J Am Coll Radiol 2024;21(2):248–256.
20. open-compass/opencompass. GitHub. https://github.com/open-compass/opencompass. Accessed February 1, 2024.
21. Reid M, Savinov N, Teplyashin D, et al. Gemini 1.5: unlocking multimodal understanding across millions of tokens of context. arXiv 2403.05530 [preprint] http://arxiv.org/abs/2403.05530. Posted March 8, 2024. Updated August 8, 2024. Accessed April 8, 2024.
22. Open LLM leaderboard. Hugging Face. https://huggingface.co/open-llm-leaderboard. Accessed January 20, 2024.
23. Chatbot Arena LLM leaderboard: community-driven evaluation for best LLM and AI chatbots. Hugging Face. https://lmarena.ai/?leaderboard. Accessed January 21, 2024.
24. Zheng L, Chiang W-L, Sheng Y, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. arXiv 2306.05685 [preprint] http://arxiv.org/abs/2306.05685. Posted June 9, 2023. Updated December 24, 2023. Accessed January 21, 2024.
25. OpenAI o1 System Card. OpenAI. https://openai.com/index/openai-o1-system-card/. Published December 5, 2024. Accessed December 15, 2024.
26. Glaese A, McAleese N, Trębacz M, et al. Improving alignment of dialogue agents via targeted human judgements. arXiv 2209.14375 [preprint] http://arxiv.org/abs/2209.14375. Posted September 28, 2022. Accessed March 7, 2024.
27. Kang D, Li X, Stoica I, Guestrin C, Zaharia M, Hashimoto T. Exploiting programmatic behavior of LLMs: dual-use through standard security attacks. arXiv 2302.05733 [preprint] http://arxiv.org/abs/2302.05733. Posted February 11, 2023. Accessed March 12, 2024.
28. Hubinger E, Denison C, Mu J, et al. Sleeper agents: training deceptive LLMs that persist through safety training. arXiv 2401.05566 [preprint] http://arxiv.org/abs/2401.05566. Posted January 10, 2024. Updated January 17, 2024. Accessed March 12, 2024.
29. Pfohl SR, Cole-Lewis H, Sayres R, et al. A toolbox for surfacing health equity harms and biases in large language models. Nat Med 2024;30(12):3590–3600.
30. Zack T, Lehman E, Suzgun M, et al. Assessing the potential of GPT-4 to perpetuate racial and gender biases in health care: a model evaluation study. Lancet Digit Health 2024;6(1):e12–e22.
31. Models. Hugging Face. https://huggingface.co/models. Accessed February 1, 2024.
32. Jin Q, Dhingra B, Liu Z, Cohen WW, Lu X. PubMedQA: a dataset for biomedical research question answering. arXiv 1909.06146 [preprint] http://arxiv.org/abs/1909.06146. Posted September 13, 2019. Accessed March 5, 2024.
33. Chaves JMZ, Bhaskhar N, Attias M, et al. RaLEs: a benchmark for radiology language evaluations. https://openreview.net/forum?id=PWLGrvoqiR. Published September 25, 2023. Updated November 2, 2023. Accessed February 10, 2024.
34. Van Veen D, Van Uden C, Blankemeier L, et al. Adapted large language models can outperform medical experts in clinical text summarization. Nat Med 2024;30(4):1134–1142.
35. Frantar E, Ashkboos S, Hoefler T, Alistarh D. GPTQ: accurate post-training quantization for generative pre-trained transformers. arXiv 2210.17323 [preprint] http://arxiv.org/abs/2210.17323. Posted October 31, 2022. Updated March 22, 2023. Accessed January 24, 2024.
36. ggergenov/llama.cpp. GitHub. https://github.com/ggerganov/llama.cpp. Accessed January 24, 2024.
37. Lin J, Tang J, Tang H, et al. AWQ: activation-aware weight quantization for LLM compression and acceleration. arXiv 2306.00978 [preprint] http://arxiv.org/abs/2306.00978. Posted June 1, 2023. Updated July 18, 2024. Accessed January 24, 2024.
38. Ma S, Wang H, Ma L, et al. The era of 1-bit LLMs: all large language models are in 1.58 bits. arXiv 2402.17764 [preprint] http://arxiv.org/abs/2402.17764. Posted February 27, 2024. Accessed March 5, 2024.
39. Jiang AQ, Sablayrolles A, Roux A, et al. Mixtral of experts. arXiv 2401.04088 [preprint] http://arxiv.org/abs/2401.04088. Posted January 8, 2024. Accessed January 24, 2024.
40. arcee-ai/mergekit. GitHub. https://github.com/cg123/mergekit. Accessed January 24, 2024.
41. Weyaxi/Helion-4x34B. Hugging Face. https://huggingface.co/Weyaxi/Helion-4x34B. Accessed March 1 2024.
42. oobabooga/text-generation-webui. GitHub. https://github.com/oobabooga/text-generation-webui. Accessed January 24, 2024.
43. Welcome to vLLM! vLLM. https://docs.vllm.ai/en/latest/. Accessed December 18, 2024.
44. SillyTavern/SillyTavern. GitHub. https://github.com/SillyTavern/SillyTavern. Accessed January 24, 2024.
45. huggingface/transformers. GitHub. https://github.com/huggingface/transformers. Accessed February 1, 2024.
46. turboderp/exllamav2. GitHub. https://github.com/turboderp/exllamav2. Accessed January 24, 2024.
47. Zhang T, Patil SG, Jain N, et al. RAFT: adapting language model to domain specific RAG. arXiv 2403.10131 [preprint] http://arxiv.org/abs/2403.10131. Posted March 15, 2024. Updated June 4, 2024. Accessed March 25, 2024.
48. Diao S, Wang P, Lin Y, Zhang T. Active prompting with chain-of-thought for large language models. arXiv 2302.12246 [preprint] http://arxiv.org/abs/2302.12246. Posted February 23, 2023. Updated July 21, 2024. Accessed January 21, 2024.
49. Paranjape B, Lundberg S, Singh S, Hajishirzi H, Zettlemoyer L, Ribeiro MT. ART: automatic multi-step reasoning and tool-use for large language models. arXiv 2303.09014 [preprint] http://arxiv.org/abs/2303.09014. Posted March 16, 2023. Accessed January 21, 2024.
50. Wei J, Wang X, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models. arXiv 2201.11903 [preprint] http://arxiv.org/abs/2201.11903. Posted January 28, 2022. Updated January 10, 2023. Accessed January 21, 2024.
51. Liu J, Liu A, Lu X, et al. Generated knowledge prompting for commonsense reasoning. arXiv 2110.08387 [preprint] http://arxiv.org/abs/2110.08387. Posted October 15, 2021. Posted September 28, 2022. Accessed January 21, 2024.
52. Liu Z, Yu X, Fang Y, Zhang X. GraphPrompt: unifying pre-training and downstream tasks for graph neural networks. arXiv 2302.08043 [preprint] http://arxiv.org/abs/2302.08043. Posted February 16, 2023. Updated February 25, 2023. Accessed January 21, 2024.
53. Li Z, Peng B, He P, Galley M, Gao J, Yan X. Guiding large language models via directional stimulus prompting. arXiv 2302.11520 [preprint] http://arxiv.org/abs/2302.11520. Posted February 22, 2023. Updated October 9, 2023. Accessed January 21, 2024.
54. Brown TB, Mann B, Ryder N, et al. Language models are few-shot learners. arXiv 2005.14165 [preprint] http://arxiv.org/abs/2005.14165. Posted May 28, 2020. Updated July 22, 2020. Accessed January 21, 2024.
55. Long J. Large language model guided tree-of-thought. arXiv 2305.08291 [preprint] http://arxiv.org/abs/2305.08291. Posted May 15, 2023. Accessed January 21, 2024.
56. Zhou Y, Muresanu AI, Han Z, et al. Large language models are human-level prompt engineers. arXiv 2211.01910 [preprint] http://arxiv.org/abs/2211.01910. Posted November 3, 2022. Updated March 10, 2023. Accessed January 21, 2024.
57. Gao L, Madaan A, Zhou S, et al. PAL: program-aided language models. arXiv 2211.10435 [preprint] http://arxiv.org/abs/2211.10435. Posted November 18, 2022. Updated January 27, 2023. Accessed January 21, 2024.
58. Yao S, Zhao J, Yu D, et al. ReAct: synergizing reasoning and acting in language models. arXiv 2210.03629 [preprint] http://arxiv.org/abs/2210.03629. Posted October 6, 2022. Updated March 10, 2023. Accessed January 21, 2024.
59. Wang X, Wei J, Schuurmans D, et al. Self-consistency improves chain of thought reasoning in language models. arXiv 2203.11171 [preprint] http://arxiv.org/abs/2203.11171. Posted March 21, 2022. Updated March 7, 2023. Accessed January 21, 2024.
60. Yao S, Yu D, Zhao J, et al. Tree of thoughts: deliberate problem solving with large language models. arXiv 2305.10601 [preprint] http://arxiv.org/abs/2305.10601. Posted May 17, 2023. Updated December 3, 2023. Accessed January 21, 2024.
61. Shinn N, Cassano F, Berman E, Gopinath A, Narasimhan K, Yao S. Reflexion: language agents with verbal reinforcement learning. arXiv 2303.11366 [preprint] http://arxiv.org/abs/2303.11366. Posted March 20, 2023. Updated October 10, 2023. Accessed January 24, 2024.

62. Lewis P, Perez E, Piktus A, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. arXiv 2005.11401 [preprint] http://arxiv.org/abs/2005.11401. Posted May 22, 2020. Updated April 12, 2021. Accessed January 21, 2024.
63. Rau A, Rau S, Zoeller D, et al. A context-based chatbot surpasses trained radiologists and generic ChatGPT in following the ACR appropriateness guidelines. Radiology 2023;308(1):e230970.
64. langchain-ai/langchain. GitHub. https://github.com/langchain-ai/langchain. Accessed January 25, 2024.
65. run-llama/llama_index. GitHub. https://github.com/run-llama/llama_index. Accessed January 25, 2024.
66. Gunel B, Du J, Conneau A, Stoyanov V. Supervised contrastive learning for pre-trained language model fine-tuning. arXiv 2011.01403 [preprint] http://arxiv.org/abs/2011.01403. Posted November 3, 2020. Updated April 2, 2021. Accessed January 30, 2024.
67. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv 1707.06347 [preprint] http://arxiv.org/abs/1707.06347. Posted July 20, 2017. Updated August 28, 2017. Accessed January 30, 2024.
68. Ouyang L, Wu J, Jiang X, et al. Training language models to follow instructions with human feedback. arXiv 2203.02155 [preprint] http://arxiv.org/abs/2203.02155. Posted March 4, 2022. Accessed January 30, 2024.
69. Christiano P, Leike J, Brown TB, Martic M, Legg S, Amodei D. Deep reinforcement learning from human preferences. arXiv 1706.03741 [preprint] http://arxiv.org/abs/1706.03741. Posted June 12, 2017. Updated February 17, 2023. Accessed January 30, 2024.
70. Rafailov R, Sharma A, Mitchell E, Ermon S, Manning CD, Finn C. Direct preference optimization: your language model is secretly a reward model. arXiv 2305.18290 [preprint] http://arxiv.org/abs/2305.18290. Posted May 29, 2023. Updated July 29, 2024. Accessed January 29, 2024.
71. Hu EJ, Shen Y, Wallis P, et al. LoRA: low-rank adaptation of large language models. arXiv 2106.09685 [preprint] http://arxiv.org/abs/2106.09685. Posted June 17, 2021. Updated October 16, 2021. Accessed January 26, 2024.
72. Liu Z, Zhong A, Li Y, et al. Radiology-GPT: a large language model for radiology. arXiv 2306.08666 [preprint] http://arxiv.org/abs/2306.08666. Posted June 14, 2023. Updated March 19, 2024. Accessed January 25, 2024.
73. Dettmers T, Pagnoni A, Holtzman A, Zettlemoyer L. QLoRA: efficient finetuning of quantized LLMs. arXiv 2305.14314 [preprint] http://arxiv.org/abs/2305.14314. Posted May 23, 2023. Accessed January 30, 2024.
74. Zheng Y, Zhang R, Zhang J, Ye Y, Luo Z, Ma Y. LlamaFactory: unified efficient fine-tuning of 100+ language models. arXiv 2403.13372 [preprint] http://arxiv.org/abs/2403.13372. Posted March 20, 2024. Updated June 27, 2024. Accessed March 24, 2024.
75. axolotl-ai-cloud/axolotl. GitHub. https://github.com/OpenAccess-AI-Collective/axolotl. Accessed February 10, 2024.
76. Rothenberg SA, Savage CH, Abou Elkassem A, et al. Prospective evaluation of AI triage of pulmonary emboli on CT pulmonary angiograms. Radiology 2023;309(1):e230702.
77. Topff L, Ranschaert ER, Bartels-Rutten A, et al. Artificial intelligence tool for detection and worklist prioritization reduces time to diagnosis of incidental pulmonary embolism at CT. Radiol Cardiothorac Imaging 2023;5(2):e220163.
78. Wiklund P, Medson K. Use of a deep learning algorithm for detection and triage of cancer-associated incidental pulmonary embolism. Radiol Artif Intell 2023;5(6):e220286.
79. McGowan A, Gui Y, Dobbs M, et al. ChatGPT and Bard exhibit spontaneous citation fabrication during psychiatry literature search. Psychiatry Res 2023;326:115334.
80. Omiye JA, Gui H, Rezaei SJ, Zou J, Daneshjou R. Large language models in medicine: the potentials and pitfalls: a narrative review. Ann Intern Med 2024;177(2):210–220.
81. Brady AP, Allen B, Chong J, et al. Developing, purchasing, implementing and monitoring AI tools in radiology: practical considerations. a multi-society statement from the ACR, CAR, ESR, RANZCR and RSNA. Radiol Artif Intell 2024;6(1):e230513.
82. Larson DB, Doo FX, Allen B Jr, Mongan J, Flanders AE, Wald C. Proceedings from the 2022 ACR-RSNA Workshop on Safety, Effectiveness, Reliability, and Transparency in AI. J Am Coll Radiol. 2024;21(7):1119–1129.
83. Swaminathan TP, Silver C, Akilan T. Benchmarking deep learning models on NVIDIA Jetson Nano for real-time systems: an empirical investigation. arXiv 2406.17749 [preprint]. https://arxiv.org/abs/2406.17749 Posted June 25, 2024. Accessed December 20, 2024.
84. Shazeer N. Fast transformer decoding: one write-head is all you need. arXiv 1911.02150 [preprint] http://arxiv.org/abs/1911.02150. Posted November 6, 2019. Accessed February 1, 2024.
85. Beltagy I, Peters ME, Cohan A. Longformer: the long-document transformer. arXiv 2004.05150 [preprint] http://arxiv.org/abs/2004.05150. Posted April 10, 2020. Updated December 2, 2020. Accessed January 3, 2024.
86. Roy A, Saffar M, Vaswani A, Grangier D. Efficient content-based sparse attention with routing transformers. arXiv 2003.05997 [preprint] http://arxiv.org/abs/2003.05997. Posted March 12, 2020. Updated October 24, 2020. Accessed February 1, 2024.
87. Press O, Smith NA, Lewis M. Train short, test long: attention with linear biases enables input length extrapolation. arXiv 2108.12409 [preprint] http://arxiv.org/abs/2108.12409. Posted August 27, 2021. Updated April 22, 2022. Accessed February 1, 2024.
88. Dao T, Fu DY, Ermon S, Rudra A, Ré C. FlashAttention: fast and memory-efficient exact attention with IO-awareness. arXiv 2205.14135 [preprint] http://arxiv.org/abs/2205.14135. Posted May 27, 2022. Updated June 23, 2022. Accessed January 3, 2024.
89. Xiao G, Tian Y, Chen B, Han S, Lewis M. Efficient streaming language models with attention sinks. arXiv 2309.17453 [preprint] http://arxiv.org/abs/2309.17453. Posted September 29, 2023. April 7, 2024. Accessed December 18, 2024.
90. Rajpurkar P, Lungren MP. The current and future state of AI interpretation of medical images. N Engl J Med 2023;388(21):1981–1990.
91. Yang L, Xu S, Sellergren A, et al. Advancing multimodal medical capabilities of gemini. arXiv 2405.03162 [preprint] http://arxiv.org/abs/2405.03162. Posted May 6, 2024. Accessed May 8, 2024.
92. Liu H, Li C, Li Y, et al. LLaVA-NeXT: improved reasoning, OCR, and world knowledge. LLaVA. https://llava-vl.github.io/blog/2024-01-30-llava-1-6/. Published January 30, 2024. Accessed January 31, 2024.
93. Bai Z, Wang P, Xiao T, et al. Hallucination of multimodal large language models: a survey. arXiv 2404.18930 [preprint] http://arxiv.org/abs/2404.18930. Posted April 29, 2024. Accessed May 8, 2024.
94. Mehandru N, Miao BY, Almaraz ER, Sushil M, Butte AJ, Alaa A. Evaluating large language models as agents in the clinic. NPJ Digit Med 2024;7(1):84.
95. Faggioli G, Dietz L, Clarke CLA, et al. Who determines what is relevant? Humans or AI? Why not both? Commun ACM 2024;67(4):31–34.
96. Yue L, Fu T. CT-Agent: clinical trial multi-agent with large language model-based reasoning. arXiv 2404.14777 [preprint] http://arxiv.org/abs/2404.14777. Posted April 23, 2024. Updated July 20, 2024. Accessed July 12, 2024.
97. Du Y, Li S, Torralba A, Tenenbaum JB, Mordatch I. Improving factuality and reasoning in language models through multiagent debate. arXiv 2305.14325 [preprint] http://arxiv.org/abs/2305.14325. Posted May 23, 2023. Accessed July 12, 2024.
98. Xu FF, Song Y, Li B, et al. TheAgentCompany: benchmarking LLM agents on consequential real world tasks. arXiv 2412.14161 [preprint] http://arxiv.org/abs/2412.14161. Posted December 18, 2024. Accessed December 20, 2024.