

LLM Post-Training: A Deep Dive into Reasoning Large Language Models

Komal Kumar*, Tajamul Ashraf*, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip H.S. Torr, Fahad Shahbaz Khan, Salman Khan

Abstract—Large Language Models (LLMs) have transformed the natural language processing landscape and brought to life diverse applications. Pretraining on vast web-scale data has laid the foundation for these models, yet the research community is now increasingly shifting focus toward post-training techniques to achieve further breakthroughs. While pretraining provides a broad linguistic foundation, post-training methods enable LLMs to refine their knowledge, improve reasoning, enhance factual accuracy, and align more effectively with user intents and ethical considerations. Fine-tuning, reinforcement learning, and test-time scaling have emerged as critical strategies for optimizing LLMs performance, ensuring robustness, and improving adaptability across various real-world tasks. This survey provides a systematic exploration of post-training methodologies, analyzing their role in refining LLMs beyond pretraining, addressing key challenges such as catastrophic forgetting, reward hacking, and inference-time trade-offs. We highlight emerging directions in model alignment, scalable adaptation, and inference-time reasoning, and outline future research directions. We also provide a public repository to continually track developments in this fast-evolving field: <https://github.com/mbzuai-oryx/Awesome-LLM-Post-training>.

Index Terms—Reasoning Models, Large Language Models, Reinforcement Learning, Reward Modeling, Test-time Scaling

arXiv:2502.21321v2 [cs.CL] 24 Mar 2025

1 Introduction

CONTEMPORARY Large Language Models (LLMs) exhibit remarkable capabilities across a vast spectrum of tasks, encompassing not only text generation [1, 2, 3] and question-answering [4, 5, 6, 7], but also sophisticated multi-step reasoning [8, 9, 10, 11]. They power applications in natural language understanding [12, 13, 14, 15, 16, 17], content generation [18, 19, 20, 21, 22, 23, 24, 25], automated reasoning [26, 27, 28, 29], and multimodal interactions [30, 31, 33]. By leveraging vast self-supervised training corpora, these models often approximate human-like cognition [34, 35, 36, 37, 38], demonstrating impressive adaptability in real-world settings.

Despite these impressive achievements, LLMs remain prone to critical shortcomings. They can generate misleading or factually incorrect content (commonly referred to as “hallucinations”) and may struggle to maintain logical consistency throughout extended discourse [41, 42, 43, 44, 45, 46]. Moreover, the concept of reasoning in LLMs remains a topic of debate. While these models can produce responses that appear logically coherent, their reasoning is fundamentally distinct from human-like logical inference [47, 34, 48, 49]. This distinction is crucial, as it helps explain why LLMs can

- *Equal contribution. Corresponding authors (Email: komal.kumar@mbzuai.ac.ae, tajamul.ashraf@mbzuai.ac.ae)
- Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, and Salman Khan are with Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE.
- Mubarak Shah is with the Center for Research in Computer Vision at the University of Central Florida, Orlando, FL 32816, USA.
- Ming-Hsuan Yang is with the University of California at Merced, Merced, CA 95343 USA, and also with Google DeepMind, Mountain View, CA 94043, USA.
- Phillip H.S. Torr is with the Department of Engineering Science, University of Oxford, Oxford OX1 2JD, UK.

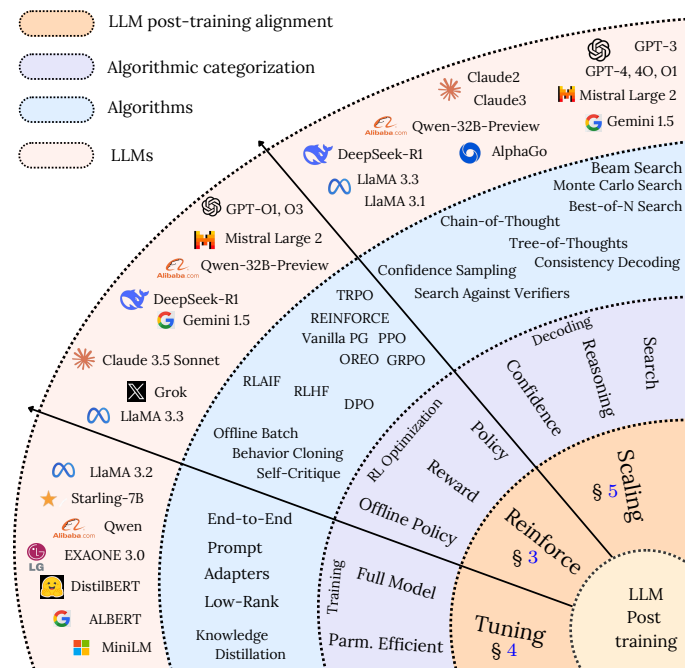


Fig. 1: A taxonomy of post-training approaches for LLMs (LLMs), categorized into Fine-tuning, Reinforcement Learning, and Test-time Scaling methods. We summarize the key techniques used in recent LLM models, such as GPT-4 [39], LLaMA 3.3 [13], and Deepseek R1 [40].

produce compelling outputs while still stumbling on relatively simple logical tasks. Unlike symbolic reasoning that manipulates explicit rules and facts, LLMs operate in an implicit and probabilistic manner [50, 42, 51]. For the scope of this work,

‘reasoning’ in LLMs refers to their ability to generate logically coherent responses based on statistical patterns in data rather than explicit logical inference or symbolic manipulation. Additionally, models trained purely via next-token prediction can fail to align with user expectations or ethical standards, especially in ambiguous or malicious scenarios [4, 52]. These issues underscore the need for specialized strategies that address reliability, bias, and context sensitivity in LLM outputs.

LLMs training can be broadly categorized into two stages: pre-training, which generally relies on a next-token prediction objective over large-scale corpora, and post-training, encompassing multiple rounds of fine-tuning and alignment. Post-training mechanisms aim to mitigate LLMs limitations by refining model behavior and aligning outputs with human intent, mitigating biases or inaccuracies [53].

Adapting LLMs to domain-specific tasks often involves techniques like **fine-tuning** [54, 55, 56], which enables task-specific learning but risks overfitting and incurs high computational costs. To address these challenges, approaches such as **Reinforcement Learning (RL)** [57, 58, 59] enhance adaptability by leveraging dynamic feedback and optimizing sequential decision-making. Additionally, advances in **scaling** techniques, including Low-Rank Adaptation (LoRA) [60], adapters [365?], and Retrieval-Augmented Generation (RAG) [61, 62, 63], improve both computational efficiency and factual accuracy. These strategies, coupled with distributed training frameworks, facilitate large-scale deployment and further boost the usability of LLMs across diverse applications (Figure 1). Through these targeted post-training interventions, LLMs become better aligned with human intent and ethical requirements, ultimately enhancing their real-world applicability. Below, we summarize key post-training stages.

a) Fine-Tuning in LLMs: Fine-tuning adapts pre-trained LLMs to specific tasks or domains by updating parameters on curated datasets [64, 65, 66, 54, 55, 67, 56]. While LLMs generalize well after large-scale pretraining, fine-tuning enhances performance in tasks like sentiment analysis [68, 69], question answering, and domain-specific applications such as medical diagnosis [70, 71, 72]. This process, typically supervised, aligns models with task requirements but poses challenges like overfitting, high computational costs, and sensitivity to data biases [56, 31, 16]. To this end, parameter-efficient techniques like LoRA [60] and adapters learn task-specific adaptation by updating explicit parameters, significantly reducing computational overhead. As models specialize, they may struggle with out-of-domain generalization, underscoring the trade-off between specificity and versatility.



Fine-tuning tailors LLMs for specific tasks, improving performance but risking overfitting, high compute costs, and reduced generalization.

b) Reinforcement Learning in LLMs: In conventional RL, an agent interacts with a structured environment, taking discrete actions to transition between states while maximizing cumulative rewards [73]. RL domains—such as robotics, board games, and control systems—feature well-defined state-action spaces and clear objectives [74, 75]. RL in LLMs differs significantly. Instead of a finite action set, LLMs select tokens

from a vast vocabulary, and their evolving state comprises an ever-growing text sequence [16, 59, 76, 57]. This complicates planning and credit assignment, as the impact of token selection may only emerge later. Feedback in language-based RL is also sparse [77], subjective, and delayed, relying on heuristic evaluations and user preferences rather than clear performance metrics [78, 79, 58]. Additionally, LLMs must balance multiple, sometimes conflicting, objectives, unlike conventional RL, which typically optimizes for a single goal. Hybrid approaches combining process-based rewards (e.g., chain-of-thought reasoning) with outcome-based evaluations (e.g., response quality) help refine learning [8, 80, 81]. Thus, RL for LLMs requires specialized optimization techniques to handle high-dimensional outputs, non-stationary objectives, and complex reward structures, ensuring responses remain contextually relevant and aligned with user expectations.



Reinforcement in LLMs extends beyond conventional RL as it navigates vast action spaces, handles subjective and delayed rewards, and balances multiple objectives, necessitating specialized optimization techniques.

c) Test Time Scaling in LLMs: Test Time Scaling is optimizing model performance and efficiency without altering the core architecture. It enables better generalization while minimizing computational overhead. It is crucial for enhancing the performance and efficiency of LLMs. It helps improve generalization across tasks but introduces significant computational challenges [82, 83]. Balancing performance and resource efficiency requires targeted strategies at inference. Techniques like CoT [8] reasoning and Tree-of-Thought (ToT) [84] frameworks enhance multi-step reasoning by breaking down complex problems into sequential or tree-structured steps. Additionally, search-based techniques [85, 86, 87, 88] enable iterative exploration of possible outputs, helping refine responses and ensure higher factual accuracy. These approaches, combined with methods like LoRA [60], adapters, and RAG [61, 62, 89], optimize the model’s ability to handle complex, domain-specific tasks at scale. RAG enhances factual accuracy by dynamically retrieving external knowledge, mitigating limitations of static training data [62, 24, 90]. Distributed training frameworks leverage parallel processing to manage the high computational demands of large-scale models. Test-time scaling optimizes inference by adjusting parameters dynamically based on task complexity [83, 91]. Modifying depth, width, or active layers balances computational efficiency and output quality, making it valuable in resource-limited or variable conditions. Despite advancements, scaling presents challenges such as diminishing returns, longer inference times, and environmental impact, especially when search techniques are performed at test time rather than during training [82]. Ensuring accessibility and feasibility is essential to maintain high-quality, efficient LLM deployment.



Test-time scaling enhances the adaptability of LLMs by dynamically adjusting computational resources during inference.

1.1 Prior Surveys

Recent surveys on RL and LLMs provide valuable insights but often focus on specific aspects, leaving key post-training components underexplored [51, 92, 93, 94]. Many works examine RL techniques like Reinforcement Learning from Human Feedback (RLHF) [58], Reinforcement Learning from AI Feedback (RLAIF) [95], and Direct Preference Optimization (DPO) [57], yet they overlook fine-tuning, scaling, and critical benchmarks essential for real-world applications. Furthermore, these studies have not explored the potential of RL even without human annotation supervised finetuning in various frameworks such as DeepSeek R1 with GRPO [59]. Other surveys explore LLMs in traditional RL tasks, such as multi-task learning and decision-making, but they primarily classify LLM functionalities rather than addressing test-time scaling and integrated post-training strategies [96, 97]. Similarly, studies on LLM reasoning [98, 99, 100, 55, 101, 102, 103, 104] discuss learning-to-reason techniques but lack structured guidance on combining fine-tuning, RL, and scaling. The absence of tutorials, along with reviews of software libraries and implementation tools, further limits their practicality. In contrast, this survey offers a comprehensive view of LLM post-training as shown in Figure 1 by systematically covering fine-tuning, RL, and scaling as interconnected optimization strategies. We offer practical resources—benchmarks, datasets, and tutorials—to aid LLM refinement for real-world applications.

1.2 Contributions

The key contributions of this survey are as follows:

- We provide a comprehensive and systematic review of post-training methodologies for LLMs, covering **fine-tuning**, **RL**, and **scaling** as integral components of model optimization.
- We offer a **structured taxonomy** of post-training techniques, clarifying their roles and interconnections, and present insights into open challenges and future research directions in optimizing LLMs for real-world deployment.
- Our survey provides practical guidance by introducing key **benchmarks, datasets, and evaluation metrics** essential for assessing post-training effectiveness, ensuring a structured framework for real-world applications.

2 Background

The LLMs have transformed reasoning by learning to predict the next token in a sequence based on vast amounts of text data [105, 4] using Maximum Likelihood Estimation (MLE) [106, 3, 107], which maximizes the probability of generating the correct sequence given an input. This is achieved by minimizing the negative log-likelihood:

$$\mathcal{L}_{\text{MLE}} = - \sum_{t=1}^T \log P_{\theta}(y_t | y_{<t}, X).$$

Here, X represents the input, such as a prompt or context. $Y = (y_1, y_2, \dots, y_T)$ is the corresponding target output sequence, and $P_{\theta}(y_t | y_{<t}, X)$ denotes the model’s predicted probability for token y_t , given preceding tokens.



Token-wise training can ensure fluency but may cause cascading errors due to uncorrected mistakes in inference.

As these models scale, they exhibit emergent reasoning abilities, particularly when trained on diverse data that include code and mathematical content [108, 8]. However, despite their impressive capabilities, LLMs struggle to maintain coherence and contextual relevance over long sequences. Addressing these limitations necessitates a structured approach to sequence generation, which naturally aligns with RL.

Since LLMs generate text autoregressively—where each token prediction depends on previously generated tokens—this process can be modeled as a sequential decision-making problem within a Markov Decision Process (MDP) [109]. In this setting, the state s_t represents the sequence of tokens generated so far, the action a_t is the next token, and a reward $R(s_t, a_t)$ evaluates the quality of the output. An LLM’s policy π_{θ} is optimized to maximize the expected return:

$$J(\pi_{\theta}) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right],$$

where γ is the discount factor that determines how strongly future rewards influence current decisions. A higher γ places greater importance on long-term rewards. The primary objective in RL is to learn a policy that maximizes the expected cumulative reward, often referred to as the return. This requires balancing exploration—trying new actions to discover their effects—and exploitation—leveraging known actions that yield high rewards. While LLMs optimize a likelihood function using static data, RL instead optimizes the expected return through dynamic interactions. To ensure that LLMs generate responses that are not only statistically likely but also aligned with human preferences, it is essential to go beyond static optimization methods. While likelihood-based training captures patterns from vast corpora, it lacks the adaptability needed for refining decision-making in interactive settings. By leveraging structured approaches to maximizing long-term objectives, models can dynamically adjust their strategies, balancing exploration and exploitation to improve reasoning, coherence, and alignment [110, 111, 49, 48].



LLMs exhibit emergent abilities due to scale, while RL refines and aligns them for better reasoning and interaction.

2.1 RL based Sequential Reasoning.

The chain-of-thought reasoning employed in modern LLMs is naturally framed as an RL problem. In this perspective, each intermediate reasoning step is treated as an action contributing to a final answer. The objective function $J(\pi_{\theta})$ represents the expected reward of the policy π_{θ} , capturing how well the model performs over multiple reasoning steps. The policy gradient update is given by:

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau} \left[\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(x_t | x_{1:t-1}) A(s_t, a_t) \right],$$

where the advantage function $A(s_t, a_t)$ distributes credit to individual steps, ensuring that the overall reasoning process is refined through both immediate and delayed rewards. Such formulations, including step-wise reward decomposition [112, 113], have been crucial for enhancing the interpretability and performance of LLMs on complex reasoning tasks. In traditional RL formulations, an agent has:

$$\text{Value function: } V(s) = \mathbb{E}[\text{future return} \mid s],$$

$$\text{Action-value (Q-) function: } Q(s, a) = \mathbb{E}[\text{future return} \mid s, a],$$

$$\text{Advantage function: } A(s, a) = Q(s, a) - V(s).$$


In words, $A(s, a)$ measures *how much better or worse* it is to take a specific action a in state s compared to what the agent would *normally* expect (its baseline $V(s)$).

2.2 Early RL Methods for Language Modeling.

Here, we briefly overview pioneering methods that laid the groundwork for applying RL to language generation tasks. These initial efforts train a decision-making model (policy (p_θ)) by directly adjusting its parameters to maximize rewards. Some policy gradient approaches are explained below: **Policy Gradient (REINFORCE)**. The REINFORCE algorithm [114, 115] is a method used to improve decision-making by adjusting the model’s strategy (policy) based on rewards received from its actions. Instead of directly learning the best action for every situation, the algorithm refines how likely different actions are to be chosen, gradually improving outcomes over time. At each step, the model updates its parameters (θ) based on how well its past decisions performed:

$$\theta \leftarrow \theta + \alpha \left(G - b \right) \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t \mid s_t).$$

Here: G represents the total reward the model accumulates over an episode, b is a baseline value that helps reduce variance, making learning more stable, $\nabla_{\theta} \log \pi_{\theta}(a_t \mid s_t)$ measures how much a small change in θ affects the probability of choosing action a_t given state s_t , α is the learning rate, controlling how much the policy updates at each step.

 **Optimizing actions based on long-term rewards, which account for the cumulative benefits of a sequence of reasoning steps rather than just immediate outcomes, is fundamental in recent LLMs. This approach allows models to explore multiple reasoning paths more effectively.**

Curriculum Learning with MIXER. Ranzato et al. [116] introduces a gradual transition from maximum likelihood estimation (MLE) to RL. The overall loss is a weighted combination:

$$\mathcal{L} = \lambda(t) \mathcal{L}_{\text{MLE}} + (1 - \lambda(t)) \mathcal{L}_{\text{RL}},$$

where $\lambda(t)$ decreases with training time. This curriculum helps the model ease into the RL objective and mitigate the mismatch between training and inference.

Self-Critical Sequence Training (SCST). SCST [117] refines the policy gradient method by comparing the model’s sampled outputs against its own best (greedy) predictions. Instead of using an arbitrary baseline, SCST uses the model’s

own highest-scoring output, ensuring that updates directly improve performance relative to what the model currently considers its best response. The gradient update follows:

$$\nabla_{\theta} J(\pi_{\theta}) \approx \left(r(y^s) - r(\hat{y}) \right) \nabla_{\theta} \log \pi_{\theta}(y^s),$$

where y^s is a sampled sequence, \hat{y} is the greedy output, and $r(y)$ represents an evaluation metric such as BLEU [118] for translation or CIDEr [119] for image captioning. Since the learning signal is based on the difference $r(y^s) - r(\hat{y})$, the model is explicitly trained to generate outputs that score higher than its own baseline under the evaluation metric. If the sampled output outperforms the greedy output, the model reinforces it; otherwise, it discourages that sequence. This direct feedback loop ensures that training aligns with the desired evaluation criteria rather than just maximizing likelihood. By leveraging the model’s own best predictions as a baseline, SCST effectively reduces variance and stabilizes training while optimizing real-world performance metrics.

Minimum Risk Training (MRT). MRT [151] directly minimizes the expected risk over the output distribution. Given a task-specific loss $\Delta(y, y^*)$ comparing the generated output y with the reference y^* , the MRT objective is defined as:

$$\mathcal{L}_{\text{MRT}}(\theta) = \sum_{y \in \mathcal{Y}} p_{\theta}(y \mid x) \Delta(y, y^*).$$

This formulation incorporates evaluation metrics (e.g., 1 – BLEU) directly into training, enabling fine-grained adjustments of the policy.

Advantage Actor-Critic (A2C/A3C). RL methods like REINFORCE [114] rely solely on policy gradients, which suffer from high variance, leading to unstable and inefficient learning. Since the reward signal fluctuates across different trajectories, updates may be noisy, causing slow or erratic convergence. To mitigate this, Actor-Critic methods [152, 153, 154, 155] combine two components as follows: an actor and a critic. The actor is a policy $\pi_{\theta}(a_t \mid s_t)$ that selects actions a_t at state s_t , while the critic is a value function $V_{\phi}(s_t)$ that evaluates the expected return of a state. The critic provides a more stable learning signal, reducing variance in policy updates and enabling efficient learning in continuous action spaces. Actor updates are guided by the policy gradient theorem, where the advantage function $A(s_t, a_t)$ defined in Sec. 2.1, determines how much better an action a_t is compared to the expected value of state s_t . The policy with the learning rate α is updated as:

$$\theta \leftarrow \theta + \alpha A(s_t, a_t) \nabla_{\theta} \log \pi_{\theta}(a_t \mid s_t).$$

Meanwhile, the critic is updated using temporal difference learning, minimizing the squared error between its estimate and the actual return:

$$\phi \leftarrow \phi - \beta \nabla_{\phi} \left(V_{\phi}(s_t) - G_t \right)^2.$$

where β is a learning rate for critic. To enhance stability and efficiency, several improvements have been proposed. Eligibility traces allow learning from recent states, enabling faster convergence. Function approximation with neural networks ensures effective handling of high-dimensional inputs. Advanced variants such as Natural Gradient methods [156] adjust updates using the Fisher Information Matrix, improving convergence speed.

RL Enhanced LLMs	Developer	Source	# Params	RL Methods	Fine-Tuning	Architecture Type	Model	TTS
DeepSeek-V2 [16]	Deepseek	Link	236B-A21B	GRPO	DPO + GRPO	MoE	Open	✓
GPT 4.5 [120]	OpenAI	Link	-	RLHF, PPO, RBRM	SFT + RLHF	MoE	Closed	✓
Gemini [15]	Google	Link	-	RLHF	SFT + RLHF	Single Model	Closed	✗
Claude 3.7 [121]	Anthropic	Link	-	RLAIF	SFT + RLAIF	Single Model	Closed	✗
Reka [122]	Reka	Link	7B, 21B	RLHF, PPO	SFT + RLHF	Single Model	Closed	✗
DeepSeekR1 [40]	Deepseek	Link	240B-A22B	GRPO	DPO + GRPO	MoE	Open	✓
Nemotron-4 340B [123]	NVIDIA	Link	340B	DPO, RPO	DPO + RPO	Single Model	Closed	✗
Falcon [124]	TII	Link	40B	-	SFT	Single Model	Open	✗
GPT-4 [39]	OpenAI	Link	-	RLHF, PPO, RBRM	SFT + RLHF	MoE	Closed	✓
Llama 3 [13]	Meta	Link	8B, 70B, 405B	DPO	SFT + DPO	Single Model	Open	✗
Qwen2 [125]	Alibaba	Link	(0.5-72)B, 57B-A14B	DPO	SFT + DPO	Single Model	Open	✓
Gemma2 [14]	Google	Link	2B, 9B, 27B	RLHF	SFT + RLHF	Single Model	Open	✗
Starling-7B [26]	Berkeley	Link	7B	RLAIF, PPO	SFT + RLAIF	Single Model	Open	✗
Moshi [126]	Kyutai	Link	7B	-	-	Multi-modal	Open	✓
Athene-70B [127]	Nexusflow	Link	70B	RLHF	SFT + RLHF	Single Model	Open	✗
GPT-3.5 [39]	OpenAI	Link	3.5B, 175B	RLHF, PPO	SFT + RLHF	MoE	Closed	✓
Hermes 3 [128]	Nous	Link	8B, 70B, 405B	DPO	SFT + DPO	Single Model	Open	✗
Zed [129]	Zed AI	Link	500B	RLHF	RLHF	Multi-modal	Open	✓
PaLM 2 [130]	Google	Link	-	RLHF	-	Single Model	Closed	✓
InternLM2 [131]	SAIL	Link	1.8B, 7B, 20B	RLHF, PPO	SFT + RLHF	Single Model	Closed	✗
Supernova [132]	Nova AI	Link	220B	RLHF	RLHF	Multi-modal	Open	✓
Grok3 [133]	Grok-3	Link	175B	-	DPO	Dense	Open	✓
Pixtral [134]	Mistral AI	Link	12B, 123B	-	PEFT	Multimodal	Open	✓
Minimaxtext [135]	MiniMax	Link	456B	-	SFT	Single Model	Closed	✗
Amazonnova [136]	Amazon	Link	-	DPO, RLHF, RLAIF	SFT	Single Model	Closed	✗
Fugakullm [137]	Fujitsu	Link	13B	-	-	Single Model	Closed	✗
Nova [138]	Rubik's AI	Link	-	-	SFT	Proprietary	Closed	✗
03 [139]	OpenAI	Link	-	RL through CoT	RL through CoT	Single Model	Closed	✓
Dbrx [140]	Databricks	Link	136B	-	SFT	Single Model	Open	✗
Instruct-GPT [58]	OpenAI	Link	1.3B, 6B, 175B	RLHF, PPO	SFT + RLHF	Single Model	Closed	✗
Openassistant [141]	LAION	Link	17B	-	SFT	Single Model	Open	✗
ChatGLM [142]	Zhipu AI	Link	6B, 9B	ChatGLM-RLHF	SFT + RLHF	Single Model	Open	✗
Zephyr [143]	Argilla	Link	141B-A39B	ORPO	DPO + ORPO	MoE	Open	✓
phi-3 [17]	Microsoft	Link	3.8B, 7B, 14B	DPO	SFT + DPO	Single Model	Closed	✗
Jurassic [144]	A121 Labs	Link	-	-	SFT	Proprietary	Closed	✗
Kimi K1.5 [145]	Moonshot AI	Link	150B	-	RLHF	Multi-modal	Open	✓
Phi-4 [146]	Microsoft	Link	28B, 70B, 140B	DPO	SFT + DPO	Single Model	Closed	✗
Chameleon [147]	Meta AI	Link	34B	-	SFT	Single Model	Open	✗
Cerebrasgpt [148]	Cerebras	Link	13B	-	SFT	Single Model	Open	✗
Bloombergpt [149]	Bloomberg L.P.	Link	50B	-	SFT	Single Model	Closed	✗
Chinchilla [150]	DeepMind	Link	70B	RLHF, PPO	SFT	Single Model	Closed	✗

TABLE 1: An overview of reinforcement learning-enhanced LLMs, where '141B-A39B' denotes a Mixture of Experts (MoE) model with 141 billion total parameters, of which 39 billion are utilized during inference. TTS stands for Test-Time Scaling.

A notable early example is Barto’s Actor-Critic model [157], where the critic uses a linear function $V_\phi(s_t)$ and the actor follows a linear policy. Modern methods like A2C (Advantage Actor-Critic) [154] and A3C (Asynchronous Advantage Actor-Critic) [155] extend this approach by parallelizing training across multiple environments, leading to faster and more stable learning. By leveraging the critic’s value estimation, actor-critic methods stabilize learning, improve sample efficiency, and accelerate convergence, making them more effective for complex decision-making tasks.

Connection with Modern Methods. The aforementioned early RL methods—REINFORCE [114], MIXER [116], SeqGAN [158], SCST [117], MRT [151], and actor-critic algorithms established the mathematical foundations for sequential reasoning in LLMs. These methods provided initial solutions to challenges such as exposure bias and high variance. Modern techniques such as large-scale RL from Human Feedback (RLHF) using PPO [73] and advanced reward models, e.g., Group Relative Policy Optimization (GRPO) [159] build directly upon these ideas. By integrating sophisticated reward signals and leveraging efficient policy updates, contemporary LLMs achieve improved reasoning, safety, and alignment with human values and pave the way for robust multi-step reasoning and improved quality of generated text. Table 1 provides an overview of recent models, including their parameters, architecture types, and the distilled RL methods employed, along with links for easy access.

3 Reinforced LLMs

From a methodological perspective, the integration of RL into LLM reasoning typically follows three core steps:

- 1) **Supervised Fine-Tuning (SFT):** Commences with a pretrained language model that is subsequently refined on a supervised dataset of high-quality, human-crafted examples. This phase ensures the model acquires a baseline compliance with format and style guidelines.
- 2) **Reward Model (RM) Training:** Generated outputs from the fine-tuned model are collected and subjected to human preference labeling. The reward model is then trained to replicate these label-based scores or rankings, effectively learning a continuous reward function that maps response text to a scalar value.
- 3) **RL Fine-Tuning:** Finally, the main language model is optimized via a policy gradient algorithm most e.g PPO to maximize the reward model’s output. By iterating this loop, the LLM learns to produce responses that humans find preferable along key dimensions such as accuracy, helpfulness, and stylistic coherence.
- 4) **Reward Modeling and Alignment:** Sophisticated reward functions are developed—drawing from human preferences, adversarial feedback, or automated metrics—to guide the model toward outputs that are coherent, safe, and contextually appropriate. These rewards are critical for effective credit assignment across multi-step reasoning processes.

Early approaches to aligning LLMs with human preferences leveraged classical RL algorithms, such as PPO [73] and Trust

Region Policy Optimization (TRPO) [160], which optimize a policy by maximizing the expected cumulative reward while enforcing constraints on policy updates via a surrogate objective function and KL-divergence regularization [161]. Improved alternatives to these methods for scalable preference-based optimization have emerged, such as Direct Preference Optimization (DPO) [57, 162] and Group Relative Policy Optimization (GRPO) [159, 59, 16], which reformulate the alignment objective as a ranking-based contrastive loss function [163] over human-labeled preference data. Unlike PPO and TRPO [160], which rely on explicit reward models and critic networks, DPO and GRPO directly optimize the policy by leveraging log-likelihood ratios and group-wise reward comparisons, respectively, eliminating the need for explicit value function approximation while preserving preference-consistent learning dynamics. This transition from classical RL-based alignment to preference-based direct optimization introduces novel formulations such as contrastive ranking loss, policy likelihood ratio regularization, and grouped advantage estimation, which are explained in subsequent sections.

3.1 Reward modeling

Let \mathcal{X} be the space of possible *queries* (e.g., user prompts). For each query $x \in \mathcal{X}$, we collect one or more *candidate responses* $\{y_j\}_{j=1}^{m_x}$ where m_x is the number of candidate responses for query x . Typically, these responses are generated by a language model or policy under different sampling or prompting conditions. Human annotators provide preference judgments for these responses. These can take various forms:

- **Pairwise preference:** For two responses y_j and y_k to the same query x , an annotator indicates whether y_j is preferred to y_k .
- **Rankings:** A partial or total ordering of the candidate responses, e.g. $y_{j_1} \succ y_{j_2} \succ \dots \succ y_{j_{m_x}}$.

We denote such human preference data by $\{r_j\}$ for each response or pair, where r_j might be a label, a rank, or an index indicating preference level. The overall dataset \mathcal{D} then consists of N annotated examples:

$$\mathcal{D} = \left\{ (x^i, \{y_j^i\}_{j=1}^{m_i}, \{\text{preferences}^i\}) \right\}_{i=1}^N.$$

In practice, a large number of queries x are sampled from real or simulated user requests. Candidate responses $\{y_j\}_{j=1}^{m_x}$ are generated by either sampling from a base language model or using beam search or other decoding strategies. Human annotators then provide pairwise or ranking feedback on which responses are better (or worse) according to predefined criteria (e.g., quality, correctness, helpfulness, etc). We train a parametric model Reward Model ($R_\theta(x, y)$), referred to as the *reward model*, to map each (query, response) pair (x, y) to a scalar score. The goal is for R_θ to reflect the *alignment* or *preference* level, such that:

$$R_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}.$$

Here \mathcal{Y} is the space of all possible responses.

To train R_θ , we use the human preference labels in \mathcal{D} to define a suitable *ranking-based* loss, as explained below.

I. Bradley–Terry Model (Pairwise). For pairwise preferences, Bradley–Terry model [164] is often used. Suppose the dataset indicates that, for a given query x , human annotators

prefer y_j to y_k , we denote it as $y_j \succ y_k$. Under Bradley–Terry, the probability of y_j being preferred over y_k is given by:

$$P(y_j \succ y_k \mid x; \theta) = \frac{\exp(R_\theta(x, y_j))}{\exp(R_\theta(x, y_j)) + \exp(R_\theta(x, y_k))}.$$

We train R_θ by *maximizing* the likelihood of observed preferences (or equivalently *minimizing* the negative log-likelihood):

$$\mathcal{L}_{\text{BT}}(\theta) = - \sum_{(x, y_j \succ y_k) \in \mathcal{D}} \log P(y_j \succ y_k \mid x; \theta).$$

II. Plackett–Luce Model¹ (Rankings). When full or partial *rankings* of m responses are available, i.e.,

$$y_{j_1} \succ y_{j_2} \succ \dots \succ y_{j_m},$$

the Plackett–Luce model [165] factorizes the probability of this ranking as:

$$P(y_{j_1}, \dots, y_{j_m} \mid x; \theta) = \prod_{\ell=1}^m \frac{\exp(R_\theta(x, y_{j_\ell}))}{\sum_{k=\ell}^m \exp(R_\theta(x, y_{j_k}))}.$$


Its negative log-likelihood is:

$$\mathcal{L}_{\text{PL}}(\theta) = - \sum_{(x, \text{rank}) \in \mathcal{D}} \sum_{\ell=1}^m \log \left(\frac{\exp(R_\theta(x, y_{j_\ell}))}{\sum_{k=\ell}^m \exp(R_\theta(x, y_{j_k}))} \right).$$

In practice, one minimizes the sum (or average) of the chosen ranking-based loss over all preference data:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x, \{y_j\}, \text{prefs}) \in \mathcal{D}} \mathcal{L}_{\text{ranking}}(\theta; x, \{y_j\}, \text{prefs}),$$

where $\mathcal{L}_{\text{ranking}}$ could be either \mathcal{L}_{BT} or \mathcal{L}_{PL} . While the reward model $R_\theta(x, y)$ provides a *scalar reward signal* reflecting human preferences, this connects to common RL concepts, especially the *advantage function*.

 **Reward modeling uses ranking-based losses to learn a function from human preferences for policy optimization.**

Reward modeling Types. Rewards can be categorized into explicit and implicit approaches.

3.1.1 Explicit Reward Modeling

Explicit reward modeling defines reward functions directly based on predefined rules, heuristics, or human annotations. This reward structure involves direct, numeric signals from humans or from specialized AI modules trained to approximate human judgments (e.g., ranking or pairwise comparison). This method can produce precise reward estimates but may be time-consuming or costly at scale. Illustrative use cases include ‘red-teaming’ exercises where experts rate the severity of toxic outputs, or domain-specialist tasks in which correctness must be validated by a subject matter expert.

1. <https://hturner.github.io/PlackettLuce/>

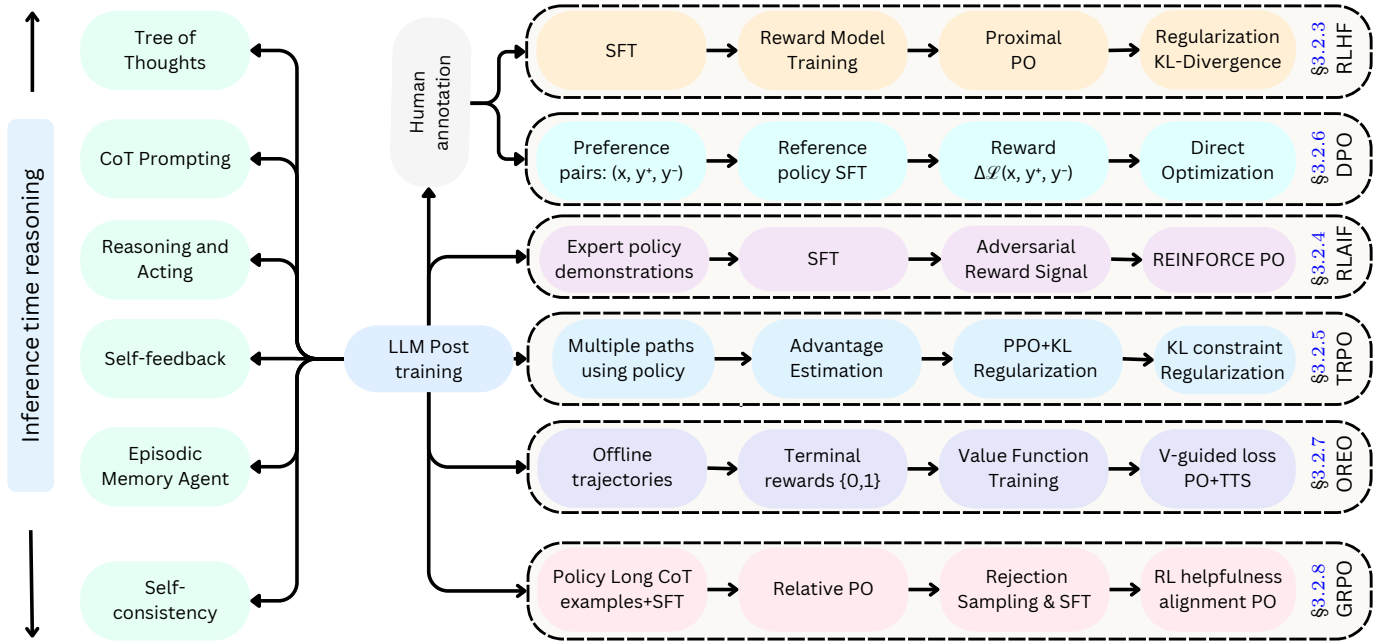


Fig. 2: Overview of Large Language Models (LLMs) reasoning methods, showcasing pathways for enhancing reasoning capabilities through approaches like Chain-of-Thought (CoT) prompting, self-feedback, and episodic memory. The diagram highlights multiple reinforcement learning-based optimization techniques, including GRPO, RLHF, DPO, and RLAIIF, for fine-tuning reasoning models with reward mechanisms and preference-based learning.

3.1.2 Implicit Reward Modeling

Implicit reward modeling infers rewards indirectly from observed behaviors, interactions, or preference signals, often leveraging machine learning techniques to uncover latent reward structures. It derives its signals from user interaction metrics such as upvotes, acceptance rates, click-through patterns, or session engagement times. While it can accumulate vast datasets with minimal overhead, this approach risks fostering behaviors that exploit engagement heuristics at the expense of content quality or veracity.

Reward Function. Defining a reward function for text generation tasks is an ill-posed problem [166, 167]. The existing RL methods in LLMs either focus on the generation process outcome (Outcome Reward Modeling) or the (Process Reward Modeling), to shape LLM behaviors. We explain these two reward modeling paradigms below.

3.1.3 Outcome Reward Modeling

Measures the end result (e.g., whether the final answer is factually correct or solves the user’s query). This model is straightforward to implement but may offer limited insight into how the conclusion was reached. It is prevalent in short-response tasks, where the user’s primary concern is the correctness or succinctness of the final statement. For long-response tasks, outcome based reward can lead to *credit assignment problem*, i.e., which specific actions or states lead to a particular reward outcome.

3.1.4 Process Reward Modeling

Assigns feedback at intermediate reasoning steps, incentivizing coherent, logically consistent, and well-structured chains of thought. This approach is particularly valuable for tasks involving mathematical derivations, legal arguments, or code

debugging, in which the path to the answer is as significant as the final statement. In such problems, the reward assigned in individual steps encourages transparency and robust step-by-step reasoning. However, it requires a more complex annotation process, e.g., requires “gold” reasoning steps or partial credit scoring. Process rewards can be combined with outcome rewards for a strong multi-phase training signal.

💡 **Policy Reward Modeling (PRM) with last-step aggregation outperforms Outcome Reward Modeling (ORM) by leveraging final-step evaluations to optimize policy updates more effectively.**

3.1.5 Iterative RL with Adaptive Reward Models

Adaptive Reward Models is a training methodology designed to continuously improve the performance of LLMs by iteratively refining the reward models and the policy model. This approach addresses the challenges of reward hacking and reward model drift, which can occur when the reward model becomes misaligned with the desired objectives during large-scale RL training. The RL process is divided into multiple iterations, where the model is trained in cycles. After each iteration, the reward model is updated based on the latest model behavior and human feedback. The reward model is not static but evolves over time to better align with human preferences and task requirements. This adaptation ensures that the reward signals remain accurate and relevant as the model improves. Repeat the iterative process until the model’s performance plateaus or meets the desired benchmarks. The reward model and policy model co-evolve, with each iteration bringing them closer to optimal alignment.

3.2 Policy Optimization

Once we have a trained reward model $R_\theta(x, y)$ that captures human preferences, we can integrate it into a RL framework to *optimize* a policy π_ϕ . In essence, we replace (or augment) the environment’s native reward signal with $R_\theta(x, y)$ so that the agent focuses on producing responses y that humans prefer for a given query x .

In typical RL notation:

- Each *state* s here can be interpreted as the partial dialogue or partial generation process for the next token (in language modeling).
- Each *action* a is the next token (or next chunk of text) to be generated.
- The *policy* $\pi_\phi(a | s)$ is a conditional distribution over the next token, parameterized by ϕ .

We seek to find ϕ that *maximizes* the expected reward under R_θ . Concretely, let x be a user query, and let $y \sim \pi_\phi(\cdot | x)$ be the generated response. We aim to solve:

$$\max_{\phi} \mathbb{E}_{x \sim \mathcal{X}} \left[\mathbb{E}_{y \sim \pi_\phi(\cdot | x)} [R_\theta(x, y)] \right].$$

This means that *on average*, over user queries x and responses y drawn from the policy π_ϕ , we want the reward model’s score $R_\theta(x, y)$ to be as high as possible.


Policy Gradient and Advantage. The modern algorithms (e.g., PPO [73], GRPO [59], TRPO [160]) rely on *policy gradients*. Figure 5 presents a structured comparison of these main RL frameworks. Each framework builds upon different principles for policy learning, reference modeling, and reward computation. Recall that the *advantage function* $A(s, a)$ quantifies how much better an action a is than the baseline expected return $V(s)$. At a high level, we update the policy π_ϕ in the direction that increases $\pi_\phi(a | s)$ for actions a with *positive advantage* and decreases it for negative-advantage actions. Formally, the advantage A_t at time t can be written as:

$$A_t = Q(s_t, a_t) - V(s_t),$$

where $Q(s_t, a_t)$ is the expected future return (sum of future rewards, including R_θ) starting from s_t when taking action a_t .

When using the reward model R_θ :

- 1) We interpret $R_\theta(x, y)$ as the *immediate* or *terminal* reward for the generated response y .
- 2) The policy’s *future returns* thus factor in how likely subsequent tokens are to be positively scored by R_θ .
- 3) The *advantage* function still captures how much better a particular generation step is compared to the baseline performance $V(s_t)$.

 **The reward model learns relative preferences rather than absolute scores. This avoids the need for calibrated human ratings and focuses on pairwise comparisons.**

3.2.1 Odds Ratio Preference Optimization (ORPO)

The simplest method is ORPO [168] which *directly* optimizing a policy from pairwise human preferences. Instead of first learning a separate reward model and then running standard RL, ORPO updates the policy to increase the likelihood of

preferred responses (according to human labels) relative to *dispreferred* ones. The key idea is to look at the *odds ratio*:

$$\frac{\pi_\phi(y_j | x)}{\pi_\phi(y_k | x)},$$

where y_j is the *preferred* response and y_k is the *less-preferred* response for a given query x .

Pairwise Preference Probability. In many direct preference approaches (e.g., Bradley–Terry style), one writes

$$P_\phi(y_j \succ y_k | x) = \sigma\left(\ln \frac{\pi_\phi(y_j | x)}{\pi_\phi(y_k | x)}\right) = \frac{1}{1 + \exp\left(\ln \frac{\pi_\phi(y_k | x)}{\pi_\phi(y_j | x)}\right)},$$

where $\sigma(\cdot)$ is the logistic (sigmoid) function. Intuitively, if the policy π_ϕ assigns higher probability to y_j than to y_k , the *odds* $\frac{\pi_\phi(y_j | x)}{\pi_\phi(y_k | x)}$ exceed 1, making y_j more likely to be the *preferred* outcome under the model.

In ORPO, one typically defines a *negative log-likelihood* loss for all pairs $\{(x, y_j \succ y_k)\}$ in the dataset:

$$\mathcal{L}_{\text{ORPO}}(\phi) = - \sum_{(x, y_j \succ y_k) \in \mathcal{D}} \log\left(P_\phi(y_j \succ y_k | x)\right).$$

Substituting the logistic form gives:

$$\mathcal{L}_{\text{ORPO}}(\phi) = - \sum_{(x, y_j \succ y_k) \in \mathcal{D}} \log\left(\frac{\pi_\phi(y_j | x)}{\pi_\phi(y_j | x) + \pi_\phi(y_k | x)}\right),$$

which can also be interpreted as maximizing the *log odds ratio* for the correct (preferred) label in each pairwise comparison.

Interpretation via Odds Ratios. By treating each preference label ($y_j \succ y_k$) as a constraint on the *odds* $\frac{\pi_\phi(y_j | x)}{\pi_\phi(y_k | x)}$, ORPO pushes the policy to increase its probability mass on y_j while decreasing it on y_k . When viewed in logarithmic space:

$$\ln\left(\frac{\pi_\phi(y_j | x)}{\pi_\phi(y_k | x)}\right),$$

a higher value corresponds to a greater likelihood of selecting y_j over y_k . Hence, minimizing $\mathcal{L}_{\text{ORPO}}(\phi)$ aligns π_ϕ with the human-labeled preferences.

⚠ Odds Ratio Preference Optimization (ORPO) is potentially less flexible for combining multiple reward signals.

3.2.2 Proximal Policy Optimization (PPO) in LLMs

A popular method for policy optimization is PPO [73], a strategy adapted to align LLMs with human feedback. Given a policy π_θ parameterized by θ and a reward function R , PPO updates the policy by optimizing a clipped objective that balances exploration and stability. Specifically, if $r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{\text{ref}}}(a_t | s_t)}$ denotes the probability ratio for an action a_t in state s_t , the clipped PPO objective is:

$$\mathcal{L}^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min(r_t(\theta) A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A_t) \right],$$

where A_t is an estimator of the advantage function and ϵ is a hyperparameter controlling the allowable deviation from the previous policy. A_t is computed using Generalized Advantage Estimation (GAE) [169] based on rewards and a learned value function. The clipping objective of PPO restricts how drastically the updated policy distribution can diverge from the

original policy. This moderation averts catastrophic shifts in language generation and preserves training stability.

Policy Optimization with KL Penalty. During RL fine-tuning with PPO, the policy π is optimized to maximize reward while staying close to the base model ρ . The modified reward function includes a KL divergence penalty:

$$J(\pi) = \mathbb{E}_{(x,y) \sim \mathcal{D}} [r(x,y) - \beta \text{KL}(\pi(\cdot|x) \parallel \rho(\cdot|x))],$$

where β controls the penalty strength. The KL term $\text{KL}(\pi \parallel \rho)$ prevents over-optimization to the proxy reward $r(x,y)$ (i.e., reward hacking).



The KL penalty is a regularization, which ensure policy retains the base model’s linguistic coherence and avoids degenerate outputs.

3.2.3 Reinforcement Learning from Human Feedback (RLHF)

RLHF [58] refines LLMs through direct human preference signals, making them more aligned with human expectations. The process involves three main steps. First, SFT is performed on a pretrained model using high-quality labeled data to establish strong linguistic and factual capabilities. Second, a reward function R is trained using human-annotated rankings of generated responses, allowing it to predict preferences and provide a scalar reward signal. Third, PPO is employed in the RLHF [58] pipeline by using human-provided preference scores (or rankings) to shape R and thereby guide the policy updates. This ensures that the model prioritizes outputs aligned with human-preferred behavior. The robust performance under conditions of noisy or partial reward signals makes PPO well-suited for text generation tasks, where large action spaces and nuanced reward definitions are common.

3.2.4 Reinforcement Learning from AI Feedback (RLAIF)

RLAIF [95] is an alternative to RLHF that replaces human annotation with AI-generated feedback. Instead of relying on human-labeled preferences, RLAIF employs a secondary, highly capable language model to generate preference labels, which are then used to train a reward model. This reward model guides reinforcement learning-based fine-tuning of the target model. RLAIF reduces the cost and time required for data collection by eliminating the need for human annotators. It enables large-scale model alignment without requiring extensive human intervention while maintaining high performance and alignment. Empirical studies indicate that RLAIF [95, 170] is a scalable and efficient alternative to RLHF, making it a promising direction for reinforcement learning-driven language model optimization.



The clipping mechanism constrains policy updates to remain within a safe trust region, which is crucial when dealing with complex, high-dimensional action spaces.

3.2.5 Trust Region Policy Optimization (TRPO)

TRPO [160] is another widely used policy optimization method, preceding PPO and sharing its fundamental goal: improving stability in reinforcement learning updates. TRPO optimizes

policy updates while ensuring they remain within a constrained trust region, measured by KL divergence.

Instead of using a clipped objective like PPO, TRPO enforces a hard constraint on policy updates by solving the following optimization problem:

$$\max_{\theta} \mathbb{E}_t \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} A_t \right]$$

subject to the constraint:

$$\mathbb{E}_t [D_{KL}(\pi_{\theta_{\text{old}}}(\cdot | s_t) \parallel \pi_{\theta}(\cdot | s_t))] \leq \delta.$$

where δ is a hyperparameter that controls how much the new policy can diverge from the old one.

Unlike PPO, which approximates this constraint using clipping, TRPO directly solves a constrained optimization problem, ensuring each update does not move too far in policy space. However, solving this constrained problem requires computationally expensive second-order optimization techniques like conjugate gradient methods, making TRPO less efficient for large-scale models like LLMs. In practice, PPO is preferred over TRPO due to its simplicity, ease of implementation, and comparable performance in large-scale applications like RLHF. However, TRPO remains an important theoretical foundation for stable policy optimization in deep reinforcement learning.

3.2.6 Direct Preference Optimization (DPO)

DPO [162] is a recently proposed method for training LLMs from human preference data without resorting to the traditional RL loop (as in RLHF with PPO). Instead of learning a separate reward function and then running policy-gradient updates, DPO directly integrates human preference signals into the model’s training objective. So instead of the above PPO objective, DPO instead constructs an objective that directly pushes up the probability of a chosen (preferred) response (y^+) while pushing down the probability of a less-preferred response (y^-), all within a single log-likelihood framework. Rather than bounding policy changes with clip, the DPO loss uses the difference between log probabilities of ‘winning’ vs. ‘losing’ responses. This explicitly encodes the user’s preference in the updated parameters.

Here, π_{θ} is the learnable policy, π_{ref} is a reference policy (often the SFT-trained model), $\sigma(\cdot)$ is the sigmoid function, β is a scaling parameter, and $\mathcal{D}_{\text{train}}$ is a dataset of triplets (x, y^+, y^-) where y^+ is the preferred output over y^- .

$$\mathcal{L}^{\text{DPO}}(\theta) = \mathbb{E}_{((x,y^+),y^-) \sim \mathcal{D}_{\text{train}}} \left[\sigma \left(\beta \log \frac{\pi_{\theta}(y^+ | x)}{\pi_{\text{ref}}(y^+ | x)} - \beta \log \frac{\pi_{\theta}(y^- | x)}{\pi_{\text{ref}}(y^- | x)} \right) \right].$$

The key insight is that an LLM can be treated as a “hidden reward model”: we can reparameterize preference data so that the model’s own log probabilities reflect how preferable one response is over another. By directly adjusting the log-likelihood of more-preferred responses relative to less-preferred ones, DPO sidesteps many complexities of RL-based methods (e.g., advantage functions or explicit clipping).

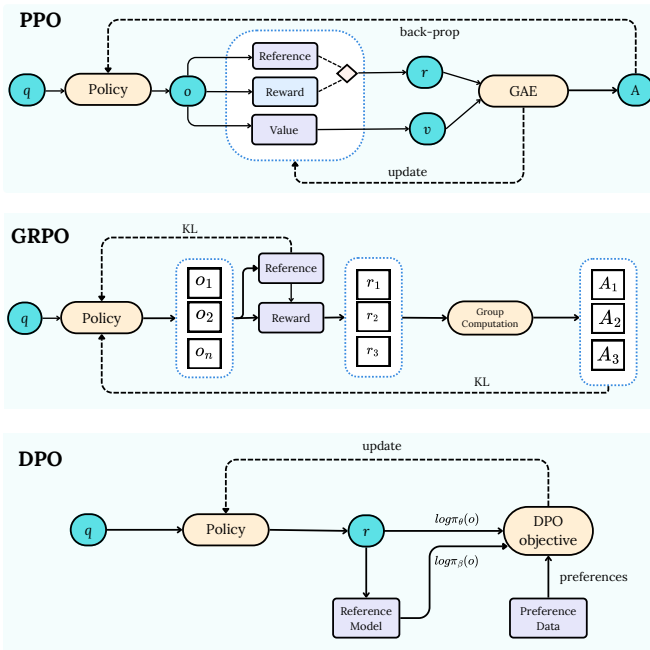


Fig. 3: Comparison of PPO [73], GRPO [59], and DPO [162]. We highlight policy models, reference models, rewards and optimization flows with corresponding loss functions.

💡 **The advantage function** $A_\phi = V_\phi(\mathbf{s}_{t+1}) - V_\phi(\mathbf{s}_t)$ quantifies per-step contributions, critical for identifying key reasoning errors. This granularity is lost in DPO, which treats entire trajectories uniformly.

Perplexity Filtering for Out-of-Distribution Data. To ensure DPO training data is on-distribution (aligned with ρ), responses are filtered using perplexity. The perplexity of a response $y = (y_1, y_2, \dots, y_T)$ is defined as:

$$PP(y) = \exp\left(-\frac{1}{T} \sum_{i=1}^T \log P_\rho(y_i | y_{<i})\right),$$

where y_i is the i -th token. Only responses with perplexity below a threshold (e.g., the 95th percentile of ρ -generated responses) are retained.

💡 **The advantage function remains a core concept to determine which actions (token choices) are better than the baseline at each step.**

3.2.7 Offline Reasoning Optimization (OREO)

OREO [171] is an offline reinforcement learning method designed to enhance LLMs' multi-step reasoning by optimizing the soft Bellman equation [109]. Unlike DPO, which relies on paired preference data, OREO uses sparse rewards based on final outcomes (e.g., correctness of reasoning chains) and jointly trains a policy model π_θ and a value function V_ϕ for

fine-grained credit assignment. The core objective minimizes the inconsistency in the soft Bellman equation:

$$V_\phi(\mathbf{s}_t) - V_\phi(\mathbf{s}_{t+1}) = r(\mathbf{s}_t, \mathbf{a}_t) - \beta \log \frac{\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)}{\pi_{\text{ref}}(\mathbf{a}_t | \mathbf{s}_t)},$$

where $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$ is the next state, r is the sparse reward, and β controls KL regularization. The policy and value losses are:

$$\mathcal{L}_V(\phi) = \frac{1}{T} \sum_{t=0}^{T-1} \left(V_\phi(\mathbf{s}_t) - R_t + \beta \sum_{i \geq t} \log \frac{\pi_\theta(\mathbf{a}_i | \mathbf{s}_i)}{\pi_{\text{ref}}(\mathbf{a}_i | \mathbf{s}_i)} \right)^2,$$

$$\mathcal{L}_\pi(\theta) = \frac{1}{T} \sum_{t=0}^{T-1} \left(V_\phi(\mathbf{s}_t) - R_t + \beta \log \frac{\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)}{\pi_{\text{ref}}(\mathbf{a}_t | \mathbf{s}_t)} \right)^2 + \alpha \mathcal{L}_{\text{reg}},$$

where \mathcal{L}_{reg} penalizes deviations from π_{ref} , and α balances regularization.

💡 **OREO's explicit value function enables test-time beam search (e.g., selecting high-value reasoning steps) and iterative training, where failed trajectories refine the policy. This contrasts with DPO implicit value function, which lacks stepwise credit assignment.**

⚠️ **OREO's computational cost scales with trajectory length and value-model training. While effective for math/agent tasks, its generalization to broader domains (e.g., coding) requires validation. Iterative training also demands careful data curation to avoid overfitting to failure modes.**

3.2.8 Group Relative Policy Optimization (GRPO)

GRPO [59] simplifies the PPO framework by eliminating the need for a separate value function. Instead, GRPO estimates the baseline from the average reward of multiple sampled outputs for the same question. The primary contribution in GRPO is that it removes the need for a separate value model (critic model) and instead estimates the baseline reward from a group of sampled LLM outputs. This significantly reduces memory usage and stabilizes policy learning. The approach also aligns well with how reward models are trained, i.e., by comparing different LLM-generated outputs rather than predicting an absolute value.

For each question q , GRPO samples a group of outputs $\{o_1, o_2, \dots, o_G\}$ from the old policy π_θ^{old} . A reward model is used to score each output in the group, yielding rewards $\{r_1, r_2, \dots, r_G\}$. The rewards are normalized by subtracting the group average and dividing by the standard deviation:

$$\bar{r}_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)}.$$

The advantage $\hat{A}_{i,t}$ for each token in the output is set as the normalized reward \bar{r}_i .

GRPO first samples a question $q \sim P(Q)$ and then samples G outputs $\{o_i\}_{i=1}^G$ from $\pi_\theta^{old}(O | q)$. Define the per-output objective as

$$J(o_i, \theta, q) = \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left(\min \left\{ r_{\text{ratio},i,t} \hat{A}_{i,t}, \right. \right. \\ \left. \left. \text{clip}(r_{\text{ratio},i,t}, 1 - \epsilon, 1 + \epsilon) \hat{A}_{i,t} \right\} \right. \\ \left. - \beta D_{KL}[\pi_\theta \| \pi_{\text{ref}}] \right).$$

Then, the GRPO objective becomes

$$J_{GRPO}(\theta) = \mathbb{E}_{q \sim P(Q)} \left[\frac{1}{G} \sum_{i=1}^G J(o_i, \theta, q) \right],$$

where the probability ratio is defined as

$$r_{\text{ratio},i,t} \triangleq \frac{\pi_\theta(o_{i,t} | q, o_{i,<t})}{\pi_\theta^{old}(o_{i,t} | q, o_{i,<t})}.$$

where ϵ is a clipping hyperparameter akin to PPO, and β adjusts the KL-divergence penalty encouraging the new policy π_θ not to deviate excessively from a reference policy π_{ref} , which is typically the initial supervised fine-tuned (SFT) model [172, 173]. GRPO can be applied in two modes: outcome supervision and process supervision.

Outcome Supervision: Provides a reward only at the end of each output. The advantage $\hat{A}_{i,t}$ for all tokens in the output is set as the normalized reward \bar{r}_i .

$$\bar{r}_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)}.$$

Process Supervision: Provides a reward at the end of each reasoning step. The advantage $\hat{A}_{i,t}$ for each token is calculated as the sum of the normalized rewards from the following steps:

$$\hat{A}_{i,t} = \sum_{\text{index}(j) \geq t} \bar{r}_{i,\text{index}(j)},$$

where $\text{index}(j)$ is the end token index of the j -th step. Overall, GRPO serves as an efficient alternative to classic actor-critic frameworks in **DeepSeekR1** [40] by leveraging group-level advantages, thereby reducing training costs without sacrificing the capacity to distinguish fine-grained differences among candidate responses.



Fine-grained per-step rewards enable the model to effectively identify and reinforce high-quality responses, boosting overall performance in complex, multi-step reasoning tasks.

3.2.9 Multi-Sample Comparison Optimization

Instead of relying solely on single-pair comparisons, multi-sample comparison optimization [174] approach compares multiple responses simultaneously to promote diversity and mitigate bias. Specifically, given a set of responses $\{y_1, y_2, \dots, y_n\}$ for a query x , the probability of observing the ranking $y_1 > y_2 > \dots > y_n$ is determined by the product

$$P(y_1 > y_2 > \dots > y_n) = \prod_i \frac{e^{R(x, y_i)}}{\sum_j e^{R(x, y_j)}}.$$

In this formulation, each response y_i is jointly evaluated in the context of all other responses, ensuring that comparisons are not isolated pairwise events but rather part of a broader ranking framework that helps capture more nuanced preferences and reduces potential biases.

3.3 Pure RL Based LLM Refinement

The work from Guo et al. (2025) [40] introduces two main models: *DeepSeek-R1-Zero* and *DeepSeek-R1*.

- **DeepSeek-R1-Zero** operates with a purely Reinforcement Learning approach, excluding any SFT.
- **DeepSeek-R1** incorporates *cold-start* data and applies a multi-stage training pipeline.

The methodology encompasses several steps (See Figure 2 in GRPO for main steps): collecting cold-start data, performing RL training, carrying out SFT, using distillation to transfer knowledge to smaller models, and addressing specific challenges such as language mixing and readability. This multi-stage pipeline ensures robustness and alignment with human preferences, while distillation enables efficient deployment of smaller models without significant performance loss.

3.3.1 Cold-Start RL Phase

The process begins with a *cold-start* RL phase, where a small amount of curated data is gathered to fine-tune an initial, or *base*, model. Following this preliminary fine-tuning, RL is conducted—often via algorithms like GRPO until convergence. The cold-start phase is critical for stabilizing the model before full RL training, preventing instability that can arise from purely RL-driven updates. The *cold-start data preparation* focuses on capturing human-readable reasoning patterns to prevent instability from purely RL-driven updates. This step generates CoT-style examples with consistent `< reasoning_process >` and `< summary >` fields, usually involving thousands of carefully curated samples. Structured CoT formats and consistent fields ensure clarity and robustness in the model’s reasoning outputs, reducing errors and improving interpretability [8, 175, 176, 177].



Providing CoT reasoning traces before RL training establishes a stronger foundation for reasoning tasks, enhancing both robustness and interpretability of outputs.

3.3.2 Rejection Sampling and Fine-tuning

This concept is also used in WebGPT [81]. Once RL stabilizes, a *rejection sampling* mechanism is employed to generate high-quality responses that are subsequently filtered for correctness, clarity, and other quality metrics. These filtered responses are then blended with additional datasets to produce a new, larger corpus for Supervised Fine-Tuning. Rejection sampling ensures that only high-quality outputs are used for further training, enhancing the model’s overall performance and reliability. After RL converges for high-stakes reasoning tasks, *rejection sampling* is used to filter a large number of generated outputs, expanding the training set. These newly generated reasoning examples (potentially up to hundreds of thousands in quantity) are mixed with existing SFT data to create a combined dataset of substantial size (often around

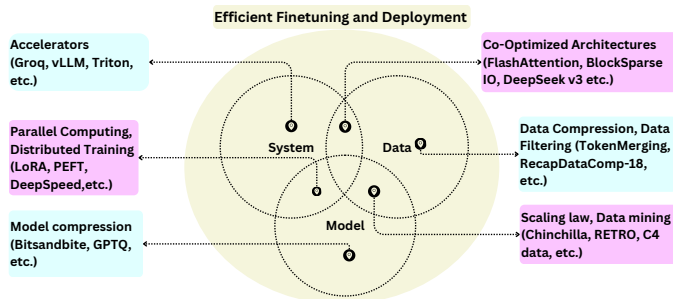


Fig. 4: This Venn diagram illustrates the interplay between System, Data, and Model for efficient finetuning and deployment. It covers strategies like accelerators (Groq, vLLM), adaptation (LoRA, PEFT), co-optimized architectures (FlashAttention), data compression (TokenMerging), scaling laws (Chinchilla), and model compression (GPTQ) to boost performance and scalability.

800k samples). Rejection sampling and dataset expansion significantly enhance the model’s coverage of general tasks while preserving its reasoning proficiency.

3.3.3 Reasoning-Oriented RL

The *reasoning-oriented RL* leverages GRPO [59], which samples a group of outputs from the current policy and computes rewards and advantages for each output. Rewards may be computed via rule-based checks, e.g., ensuring correct solutions in math or code tasks, enforcing structured CoT tags, and penalizing undesired language mixing. GRPO group-based sampling and reward computation ensure that the model prioritizes high-quality, structured outputs, enhancing its reasoning capabilities.

3.3.4 Second RL Stage for Human Alignment

A *second RL stage* further aligns the model with broader human preferences (helpfulness, harmlessness, creativity, etc.) by introducing additional reward signals and prompt distributions. The second RL stage ensures the model aligns with human values, making it more versatile and contextually aware. After re-training the base model on this combined dataset, a second round of RL can be conducted to align the model more closely with human preferences (e.g., for helpfulness and harmlessness). This RL stage fine-tunes the model to better align with human values, ensuring outputs are not only accurate but also contextually appropriate.

3.3.5 Distillation for Smaller Models

Finally, *distillation* techniques are used to transfer the refined capabilities of the main model to smaller architectures, enabling more efficient deployments without sacrificing much performance. It allows smaller models to inherit advanced reasoning capabilities, making them competitive on challenging benchmarks without the computational costs of full-scale RL training. Finally, *distillation* plays a pivotal role: the top-performing model, *DeepSeek-R1* [40], serves as a teacher to smaller architectures (e.g., Qwen or Llama families, ranging from 1.5B to 70B parameters). This transfer allows the smaller models to inherit advanced reasoning capabilities, making them competitive on challenging benchmarks without incurring the computational costs of full-scale RL training.

💡 **Distillation democratizes advanced reasoning capabilities, enabling smaller models to achieve competitive performance with reduced computational overhead.**

4 Supervised Finetuning in LLMs

As shown in Figure 2, finetuning forms a basic component of LLM post-training recipes. In this section, we summarize the different types of LLM fine-tuning mechanisms.

4.1 Instruction finetuning

In instruction finetuning, a model is trained on curated pairs of instruction (prompt) and response (completion). The main goal is to guide the LLM to follow a user-provided instruction accurately and helpfully, regardless of the task domain. This usually involves compiling large, diverse instruction-response datasets covering many task types (e.g., summarization, QA, classification, creative writing). Models such as T0 [178], FLAN [179], Alpaca [180], Vicuna [181] and Dolly [182] demonstrate how instruction-finetuned LLMs can outperform base models on zero-shot or few-shot tasks by virtue of their enhanced instruction-following abilities.

4.2 Dialogue (Multi-turn) Finetuning

Some LLMs undergo dialogue-style finetuning to better handle multi-turn conversations. Different from instruction tuning described above, here the data takes the form of a continuous dialogue (multi-turn conversations) instead of a single prompt-response pair. In this approach, training data consists of chat transcripts with multiple user queries and system responses, ensuring the model learns to maintain context across turns and produce coherent replies. Models like LaMDA [183] and ChatGPT [39] highlight how dialogue-tuned LLMs can feel more interactive and context-aware. While dialogue finetuning can overlap with instruction finetuning (because many instructions come in a chat format), specialized conversation data often yields more natural, multi-turn user experiences.

4.3 CoT Reasoning finetuning

Chain-of-Thought (CoT) reasoning finetuning teaches models to produce step-by-step reasoning traces instead of just final answers. By exposing intermediate rationales or *thoughts*, CoT finetuning can improve both interpretability and accuracy on complex tasks (e.g., math word problems, multi-hop QA). In practice, CoT finetuning uses supervised reasoning annotations (often handcrafted by experts) to show how a solution unfolds. Notable early work includes Chain-of-Thought Prompting [8] and Self-Consistency [184], which initially applied the idea to prompting; subsequent efforts (e.g., Chain-of-Thought Distillation [185]) adapt it to a full finetuning or student-teacher paradigm. These efforts have also been extended to the multimodal domain, e.g., LLaVA-CoT [186] and LlamaV-o1 [187] where image, QA and CoT reasoning steps are used in LLM finetuning.

Model	Category	Source	Description
1. Parameter-Efficient Fine-Tuning & Model Compression			
LoRA [60]	Low-Rank Adaptation	Link	Injects trainable low-rank adapters for efficient fine-tuning.
QLoRA [188]	Quantized Adaptation	Link	Combines 4-bit quantization with LoRA to enable fine-tuning on consumer GPUs
GPTQ [189]	Post-Training Quantization	Link	Optimal 4-bit quantization method for GPT-style models with minimal loss
SparseGPT [190]	Pruning	Link	One-shot pruning that preserves model quality with compensation.
PEFT (HF) [191]	Unified Fine-Tuning	Link	Library integrating LoRA, prefix tuning, and other parameter-efficient methods
BitsAndBytes [192]	Low-Precision Training	Link	Enables 8-bit optimizers and 4-bit quantization for memory-efficient training
AdaLoRA [193]	Adaptive Adaptation	Link	Dynamically allocates parameter budget between layers during fine-tuning
P-Tuning v2 [194]	Prompt Optimization	Link	Learns continuous prompt embeddings through deep prompt tuning
2. Data Management & Preprocessing			
HF Datasets [195]	Data Processing	Link	Unified API for 30k+ datasets with streaming, versioning, and preprocessing
WebDataset [196]	Data Streaming	Link	Efficient tar-based sharding format for petascale distributed training
DVC [197]	Data Versioning	Link	Git-like version control for datasets and machine learning pipelines
Apache Arrow [198]	Memory Format	Link	Language-agnostic columnar memory format for zero-copy data access
Zstandard [199]	Compression	Link	High-speed compression algorithm for training data storage/transfer
Cleanlab [200]	Data Quality	Link	Automatic detection of label errors and outliers in training datasets
3. Distributed Training & Optimization			
DeepSpeed [201]	Training Optimization	Link	ZeRO parallelism, 3D parallelism, and memory optimizations for giant models
Megatron-LM [202]	Model Parallelism	Link	NVIDIA’s optimized framework for large transformer model training
Colossal-AI [203]	Heterogeneous Training	Link	Unified system supporting multiple parallelization strategies
Horovod [204]	Distributed Training	Link	MPI-inspired framework for multi-GPU/multi-node synchronization
Ray [205]	Distributed Computing	Link	Universal framework for distributed Python applications at scale
4. Efficient Inference & Deployment			
vLLM [206]	Serving Optimization	Link	Paged attention implementation for high-throughput LLM serving
TensorRT [207]	GPU Optimization	Link	NVIDIA’s inference optimizer with kernel fusion and quantization support
Triton [208]	Serving Framework	Link	Production-grade serving with concurrent model execution support
ONNX [209]	Cross-Platform	Link	Unified inference engine with hardware-specific optimizations
OpenVINO [210]	Intel Optimization	Link	Runtime for Intel CPUs/iGPUs with pruning/quantization support
XNNPACK [211]	Mobile Inference	Link	Highly optimized floating-point kernels for ARM CPUs
Groq [212]	AI Accelerator	Link	Deterministic low-latency inference via custom tensor streaming processor
5. Integrated Development Ecosystems			
HF Ecosystem [213]	Full Stack	Link	Transformers + Datasets + Accelerate + Inference Endpoints
DeepSpeed [201]	Training/Inference	Link	Microsoft’s end-to-end solution for billion-parameter models
PyTorch [214]	Unified Framework	Link	Native LLM support via torch.compile and scaled dot-product attention
LLM Reasoners [215]	Advanced Reasoning	Link	Enhances LLM reasoning capabilities using advanced search algorithms.

TABLE 2: Comprehensive Overview of Methods and Frameworks employed in Modern LLMs

4.4 Domain-Specific (Specialized) Finetuning

When an LLM needs to excel in a specific domain (e.g., biomedicine, finance, or legal), domain-specific finetuning is used. Here, a curated corpus of domain-relevant text and labeled examples is employed to finetune the LLM. For instance, BioGPT [71] and BiMediX [216] specialize in biomedical literature, FinBERT [217] for financial texts, ClimatGPT [218, 219] for climate and sustainability and CodeT5 [220] for code understanding. Supervised finetuning in these domains often includes classification, retrieval, or QA tasks with domain-specific data, ensuring the model’s parameters adapt to the specialized language and concepts of the field. Domain-specific finetuning is also extended to vision-language models such as, [221] finetuned on remote sensing imagery, [222] on medical imaging modalities, [223, 224, 225] on spatiotemporal video inputs, and [226] adapted for chart understanding.

4.5 Distillation-Based Finetuning

Large ‘teacher’ models are sometimes used to produce labeled data or rationales, which a smaller ‘student’ model finetunes on, this is generally called knowledge distillation [227, 228]. In the context of LLMs, CoT Distillation [185] is one example where a powerful teacher LLM generates intermediate reasoning steps, and the student LLM is finetuned to reproduce both the final answer and the reasoning chain. Step-by-step distillation [229] generates descriptive rationales alongside final answers to train smaller models through distillation

with smaller datasets. This approach can yield lighter, faster models that retain much of the teacher’s performance, even in zero-shot or few-shot tasks [230].

4.6 Preference and Alignment SFT

While RLHF is not purely supervised, it starts with a supervised *preference* or *alignment* finetuning stage. This stage uses human-labeled or human-ranked examples to teach the model about desirable vs. undesirable outputs (e.g., safe vs. toxic). By training on these explicit preferences, the model becomes more aligned with user values, reducing harmful or off-topic completions. Works like InstructGPT [58] illustrate how supervised preference data is critical before reward model training and RL updates begin.

4.7 Efficient Finetuning

Fully finetuning a LLM can be computationally and memory-intensive, particularly as model sizes grow into the tens or hundreds of billions of parameters. To address these challenges, parameter-efficient finetuning (PEFT) techniques introduce a small set of trainable parameters or learnable prompts while leaving most of the model weights frozen. Approaches such as LoRA [60], Prefix Tuning [231], and Adapters [232] exemplify this strategy by injecting lightweight modules (or prompts) in specific layers, thus significantly reducing the memory footprint.

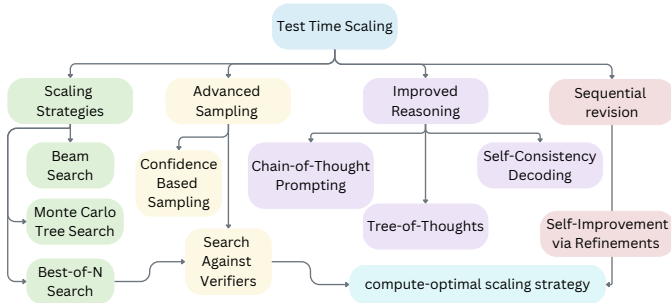


Fig. 5: An overview of Test-time Scaling methods: parallel scaling, sequential scaling, and search-based methods. It also shows how they integrate into a compute-optimal strategy.

Figure 4 illustrates how these techniques fit into a broader ecosystem that involves system-level optimizations, data management, and evaluation strategies for LLMs. In particular, PEFT approaches can be combined with quantization and pruning methods [190, 188] to further minimize memory usage and compute overhead, enabling finetuning on smaller GPUs or even consumer-grade hardware. For instance, QLoRA unifies 4-bit quantization with low-rank adaptation, while BitsAndBytes provides 8-bit optimizers to make LLM training more practical in constrained environments (Table 2).

Moreover, these PEFT methods still require supervised data to guide the adaptation process, but the reduction in the number of trainable parameters makes it more feasible to use in-domain or task-specific datasets. This is especially valuable for specialized domains (e.g., medical or software development), where data might be limited or expensive to annotate. As shown in Table 2, PEFT (HF) integrates several of these approaches (LoRA, prefix tuning, and more) into a single library, streamlining deployment in both research and production settings.

Combining efficient tuning designs like LoRA and QLoRA with system and data optimizations (Figure 4) enables cost-effective LLM adaptation for tasks like domain-specific text generation, without expensive full fine-tuning.

5 Test-time Scaling Methods

While RL fine-tunes the model’s policy, test-time scaling (TTS) enhances reasoning during inference typically without model updates. Figure 5 presents a taxonomy of TTS methods, categorizing them based on their underlying techniques.

5.1 Beam Search

Beam search was first introduced in the context of speech recognition [233]. It gained prominence as a decoding strategy for sequence models and was later adopted in neural machine translation and speech systems [234]. With the popularity of LLMs, this algorithm has been used for approximate search in many text generation tasks.

The concept of Beam search is similar to pruned *breadth-first search*, where top N highest-probability partial sequences (the ‘beam’) are kept at each step, discarding lower-

probability paths. By limiting the beam width (N), it manages the exponential search space while aiming to find a near-optimal sequence. These beams are expanded at each decoding step to find multiple probable paths. In reasoning LLMs, such paths allow us to systematically explore multiple reasoning chains in parallel, focusing on the most promising ones. This ensures that high-likelihood reasoning steps are considered, which can improve the chances of finding a correct and coherent solution compared to greedy decoding. It has traditionally been used in tasks such as translation, summarization, and code generation, where the goal is a highly probable correct sequence [93].

While modern LLMs often favor stochastic sampling (e.g., temperature sampling) to promote diversity in generated text, beam search is still a valuable technique for structured reasoning problems. For example, the Tree-of-Thoughts framework [84] allows plugging in different search algorithms to explore a tree of possible ‘thoughts’ or reasoning steps; usually a beam search (with beam width b) is used to maintain the b most promising states at each reasoning step. Here, beam search is used to systematically explore solution steps for tasks like mathematical puzzles and planning problems, pruning less promising reasoning branches and thus improving the model’s problem-solving accuracy. Beam search remains a strong baseline for test-time reasoning when one wants the model to output the single most likely reasoning path or answer under the model’s learned distribution.


5.2 Best-of-N Search (Rejection Sampling)

Best-of-N (BoN) [235] search generates N candidate outputs (usually via sampling) and then picks the best one according to a chosen criterion (e.g., a reward model or the model’s own likelihood) [236, 237, 238]. Conceptually, this is an application of rejection sampling: one draws multiple samples and rejects all but the top-rated result. Unlike Beam Search [233, 234], which incrementally expands and prunes partial hypotheses, BoN simply samples full solutions independently, allowing for greater diversity but at a higher computational cost. Beam Search systematically aims for the most probable sequence, while BoN may capture high-quality but lower-probability solutions through brute-force sampling.

Beam search (effective for harder questions) outperforms best-of-N sampling at low compute budgets, while best-of-N scales better for easier tasks.


During LLM inference, BoN is used to enhance correctness or alignment without retraining the model. By sampling multiple answers and selecting the top candidate (e.g., via a reward model or a checker), BoN effectively boosts accuracy on tasks like QA or code generation. BoN is easy to understand and implement and is almost hyper-parameter-free, with N being the only parameter that can be adjusted at inference. In reinforcement learning contexts, BoN sampling can serve as a baseline exploration mechanism i.e., to generate many roll-outs, pick the best outcome according to the learned reward, and proceed, although at increased computational overhead. *OpenAI’s* WebGPT used BoN to pick the best response via a reward model, yielding strong QA performance [81]. BoN

is also used as a simple alignment method that is highly competitive with other post-training techniques e.g., RLHF [58] and DPO [78]. Studies have shown BoN can approach or match RLHF results when guided by a sufficiently robust reward model [82, 239]. Alternatives such as speculative rejection [240] build on this idea and utilize a better reward model to improve efficiency. The studies also highlight issues of *reward hacking* if the (proxy) reward function used for BoN is imperfect [241] or instability issues if the N parameter gets very large.

 **Choice of either process reward models with beam search vs best-of-N depends on the difficulty and compute budget.**

5.3 Compute-Optimal Scaling


The Compute-Optimal Scaling Strategy (COS) [83] is a dynamic method designed to allocate computational resources efficiently during inference in LLMs, optimizing accuracy without unnecessary expense. Instead of applying a uniform sampling strategy across all inputs, this approach categorizes prompts into five difficulty levels—ranging from easy to hard—either by leveraging oracle difficulty (ground-truth success rates) or model-predicted difficulty (e.g., verifier scores from Preference Ranking Models). Once categorized, the strategy adapts compute allocation: easier prompts undergo sequential refinement, where the model iteratively refines its output to improve correctness, while harder prompts trigger parallel sampling or beam search, which explores multiple response variations to increase the likelihood of finding a correct solution. This dual approach balances exploration (for challenging inputs) and refinement (for near-correct responses), ensuring optimal performance per unit of computational effort. Remarkably, this method achieves four times lower compute usage than traditional best-of-N sampling while maintaining equivalent performance. The key insight is that by matching computational strategy to problem difficulty, it avoids wasted resources on trivial cases while ensuring sufficient sampling diversity for complex tasks. In essence, it functions as a “smart thermostat” for LLM inference, dynamically adjusting computational effort in response to input complexity, leading to a more efficient and cost-effective deployment of large-scale language models.

 **COS achieves 4× efficiency gains over best-of-N baselines by optimally balancing sequential/parallel compute. Beam search + revisions outperform larger models on easy/intermediate questions.**

5.4 Chain-of-thought prompting

CoT prompting induces LLMs to produce intermediate reasoning steps rather than jumping directly to the final answer. By breaking down problems into logical sub-steps, CoT taps into a model’s latent ability to perform multi-step inferences, significantly improving performance on tasks like math word problems, logical puzzles, and multi-hop QA.


Wei et al. [8] demonstrated CoT’s effectiveness on arithmetic and logic tasks, showing large gains over direct prompting. Kojima et al. [242] introduced Zero-Shot CoT, revealing that even adding a simple phrase like “Let’s think step by step” can trigger coherent reasoning in sufficiently large models. Subsequent works (e.g., Wang et al., 2022 [184]) combined CoT with sampling-based strategies (Self-Consistency) for even higher accuracy. As described in Sec. 5.4, CoT format data have also been used for SFT and are shown to help reshape the model responses to be more step-by-step.

 **Fine-tuning models to revise answers sequentially allows them to build on previous attempts, improving accuracy over time. This approach is particularly effective for easier questions, while parallel sampling (exploration) proves more beneficial for harder ones.**

5.5 Self-Consistency Decoding

Self-Consistency is a decoding strategy introduced by Wang et al. [243]. It was proposed as an alternative to simple greedy decoding for chain-of-thought prompts. It built upon the idea of sampling multiple distinct reasoning paths for a question and was the first to show that marginalizing over those paths can significantly improve accuracy on arithmetic and reasoning problems. In other words, it allows the model to think in many ways and then trust the consensus, which improves correctness in many reasoning scenarios.

The self-consistency method works by sampling a diverse set of reasoning chains from the model (via prompt engineering to encourage different CoTs, and using temperature sampling) and then letting the model output a final answer for each chain. Instead of trusting a single chain, the method selects the answer that is most consistent across these multiple reasoning paths, effectively a *majority vote* or *highest probability answer* after marginalizing out the latent reasoning. The intuition is that if a complex problem has a unique correct answer, different valid reasoning paths should converge to that same answer. By pooling the outcomes of many chains, the model can “decide” which answer is most supported. In application, one might sample, e.g., 20 CoTs for a math problem and see what final answer appears most frequently; that answer is then taken as the model’s output. This approach turns the one-shot CoT process into an ensemble where the model cross-verifies its answers. It is especially useful for arithmetic and commonsense reasoning tasks where reasoning diversity helps.

 **Smaller models with test-time compute can outperform much larger models in certain scenarios.**

Self-consistency is often combined with other methods: e.g., sampling multiple chains and then applying a verifier to the most common answer. Its strength lies in requiring no new training, only extra sampling, making it a popular test-time scaling strategy to obtain more reliable answers from LLMs. It has also inspired other variants, e.g., Universal Self-Consistency [244] extend the original idea (which worked only

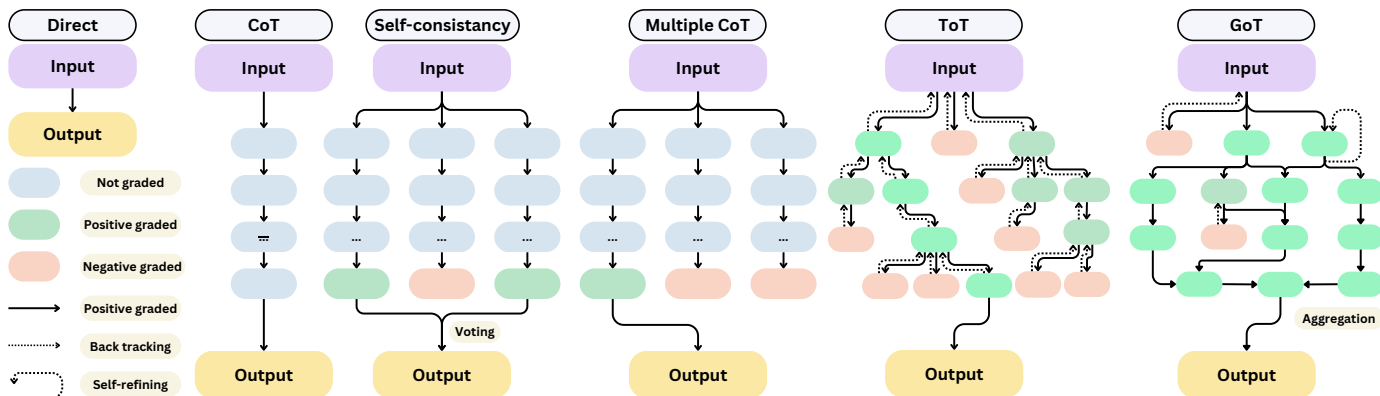


Fig. 6: This figure compares reasoning strategies in LLMs, evolving from Direct Prompting, which maps input to output without reasoning, to more structured approaches. Chain-of-Thought (CoT) introduces step-by-step reasoning, while Self-Consistency (CoT-SC) generates multiple CoT paths and selects the most frequent answer. Multiple CoTs explores diverse reasoning paths independently. Tree-of-Thoughts (ToT) structures reasoning as a tree, enabling backtracking and refinement, whereas Graph-of-Thoughts (GoT) generalizes this by dynamically aggregating and connecting thoughts. The legend deciphers key mechanisms like grading, backtracking, and self-refinement, crucial for optimizing reasoning efficiency.

with majority vote on single final answer) to more general generation tasks such as summarization and open-ended QA.

5.6 Tree-of-thoughts

ToT framework [84] generalizes the chain-of-thought approach by allowing the model to branch out into multiple possible thought sequences instead of following a single linear chain. It thus formulates the problem of language-model reasoning as a tree search, drawing on classic AI search methods inspired by human problem-solving [245, 37]. Tree of Thoughts treats intermediate reasoning steps as “nodes” in a search tree and uses the language model to expand possible next steps (thoughts) from a given state. Rather than sampling one long reasoning path, the model explores a tree of branching thoughts and can perform lookahead and backtracking. At each step, the LLM might generate several candidate next thoughts, and a heuristic or value function evaluates each partial solution state. Then a search algorithm (e.g., depth-first, breadth-first, beam search) navigates this tree, deciding which branches to explore further. This approach allows systematic exploration of different reasoning strategies: if one path leads to a dead-end, the model can return to an earlier state and try a different branch (unlike standard CoT which commits to one line of reasoning). In effect, ToT is an iterative prompting procedure where the model generates thoughts, evaluates them, and refines its approach, mimicking how a human might mentally map out various ways to solve a problem.

ToT is especially useful for complex problems like puzzles, planning tasks, or games where multiple steps and strategic exploration are needed and outperforms simpler CoT methods by systematically searching through the solution space. It provides a flexible framework – one can plug in various generation strategies (e.g. sampling vs. prompting) and search algorithms (BFS, DFS, A*, MCTS) depending on the task. Although more computationally heavy, ToT shows that allocating extra “thinking time” (compute) to explore alternatives can yield significantly better reasoning and planning performance. It has spawned follow-up research aiming to improve or utilize it for better reasoning e.g., multi-agent systems have been

combined with ToT: different LLM “agents” generate thoughts in parallel and a validator agent prunes incorrect branches, leading to improved accuracy over the single-agent ToT [246].

💡 Inference-time computation for LLMs can outperform scaling model parameters, especially for challenging reasoning tasks like math problems.


5.7 Graph of Thoughts

The Graph of Thoughts (GoT) [247] framework extends the ToT by allowing more flexible and efficient reasoning processes through graph-based structures rather than strict hierarchical trees. Thought representation differs between the two approaches: in ToT, each step in reasoning is structured as a node in a tree with fixed parent-child relationships, whereas GoT represents thoughts as nodes in a graph, enabling more adaptable dependencies and interconnections.

In terms of thought expansion strategies, ToT follows a traditional approach where multiple thought candidates are generated at each step, explored using tree-based search strategies, and pruned based on heuristics before selecting the most optimal path. In contrast, GoT incorporates graph-based thought expansion, allowing thoughts to interconnect dynamically. This enables three key transformations: aggregation (merging multiple solutions into a unified answer), refinement (iteratively improving thoughts over time), and generation (producing diverse candidates). Instead of navigating through a rigid hierarchy, GoT prioritizes thoughts using a volume metric and explores paths optimally, reducing unnecessary computations.

A critical limitation of ToT is its restricted backtracking—once a branch is discarded, it is not reconsidered. GoT overcomes this by allowing iterative refinement, where previous thoughts can be revisited, modified, and improved upon. This iterative nature is particularly useful in complex reasoning tasks where initial thoughts may require adjustments. Moreover, computational efficiency in GoT is significantly

improved by reducing redundant calculations through the merging of partial solutions.

 **GoT enhances problem-solving efficiency and adaptability, making it superior to ToT for tasks requiring complex reasoning.**

5.8 Confidence-based Sampling

In confidence-based sampling, the language model generates multiple candidate solutions or reasoning paths and then prioritizes or selects among them based on the model’s own confidence in each outcome [248]. This can happen in two ways: (a) **Selection**: Generate N outputs and pick the one with the highest log probability (i.e., the model’s most confident output). This is essentially best-of- N by probability – the model chooses the answer it thinks is most likely correct. (b) **Guided exploration**: When exploring a reasoning tree or multi-step solution, use the model’s token probabilities to decide which branch to expand (higher confidence branches are explored first). In other words, the model’s probability estimates act as a heuristic guiding the search through solution space [249]. Compared to pure random sampling, confidence-based methods bias the process toward what the model believes is right, potentially reducing wasted exploration on low-likelihood (and often incorrect) paths.

Confidence-based strategies have been incorporated at inference time e.g., a tree-based search for LLM generation [248] assigns each possible completion (leaf) a confidence score. The algorithm samples leaves in proportion to these confidence scores to decide which paths to extend [272]. Similarly, some reasoning approaches use the model’s estimated likelihood of an answer to decide when to halt or whether to ask a follow-up question – essentially if the model’s confidence is low, it might trigger further reasoning (a form of self-reflection). Confidence-based selection is also used in ensemble settings: e.g., an LLM may generate multiple answers and a secondary model evaluates the confidence of each answer being correct, picking the answer with the highest confidence. This was explored in tasks like medical Q&A, where an LLM gave an answer and a confidence score, and only high confidence answers were trusted or returned [250].

5.9 Search Against Verifiers


This verification approach [251] in LLMs enhances answer quality by generating multiple candidate responses and selecting the best one using automated verification systems. This approach shifts focus from increasing pre-training compute to optimizing test-time compute, allowing models to “think longer” during inference through structured reasoning steps or iterative refinement. The method involves two main steps: **Generation**: The model (or “proposer” produces multiple answers or reasoning paths, often using methods like high-temperature sampling or diverse decoding.

Verification: A verifier (e.g., a reward model) evaluates these candidates based on predefined criteria, such as correctness, coherence, or alignment with desired processes. Verifiers are categorized based on their evaluation focus:

- 1) **Outcome Reward Models (ORM)**: Judge only the final answer (e.g., correctness of a math solution).

- 2) **Process Reward Models (PRM)**: Evaluate the reasoning steps (e.g., logical coherence in a thought chain), providing granular feedback to prune invalid paths.

Several techniques fall under this paradigm, enhancing verification-based optimization. Best-of- N Sampling involves generating multiple answers and ranking them via a verifier (ORM/PRM), selecting the highest-scoring one, making it a simple yet effective approach for improving answer correctness. Beam Search with PRM tracks top-scoring reasoning paths (beams) and prunes low-quality steps early, similar to Tree of Thought approaches, balancing breadth and depth in reasoning path exploration. Monte Carlo Tree Search balances exploration and exploitation by expanding promising reasoning branches, simulating rollouts, and backpropagating scores, providing an optimal trade-off between search depth and verification confidence. Majority Voting (Self-Consistency) aggregates answers from multiple samples and selects the most frequent one, avoiding explicit verifiers, which works well in settings where consistency across multiple responses indicates correctness.


 **ORM is suitable for tasks where correctness is binary (right/wrong) and can be easily assessed.**

 **PRM is useful in multi-step reasoning, ensuring intermediate steps follows logical progression.**

5.10 Self-Improvement via Refinements

This approach refers to the ability of LLMs to enhance their outputs through self-evaluation and revision iteratively. This process enables models to refine their responses dynamically during inference rather than relying solely on pre-trained weights. One notable method is **Self-Refinement** [252], where an LLM generates an initial response, critiques it, and then refines the output based on its self-generated feedback. This iterative process continues until the model achieves a satisfactory result. Such techniques have been shown to improve performance on various tasks, including mathematical reasoning and code generation. This process follows these key steps: a) **Initial Generation**: The model produces an answer or reasoning path. b) **Self-Critique**: The model reviews its own response and identifies errors, inconsistencies, or areas for improvement. c) **Refinement**: The model adjusts its response based on the critique and generates an improved version. d) **Iteration**: The process repeats until the output meets a predefined quality threshold or stops improving.


Another approach is called **Self-Polish** [253], where the model progressively refines given problems to make them more comprehensible and solvable. By rephrasing or restructuring problems, the model enhances its understanding and provides more accurate solutions. Self-Polish involves progressive refinement of problem statements to make them more comprehensible and solvable. The model first rephrases or restructures the problem for better clarity, then breaks down complex queries into simpler sub-problems and refines ambiguous inputs to ensure precise understanding. By restructuring problems before solving them, the model improves its comprehension and generates more accurate solutions.

 **Self-improvement methodologies represent a paradigm shift in LLM optimization, emphasizing active reasoning and internal feedback over static pre-training. By iterating on their own responses, models achieve greater consistency and accuracy across a wide range of applications.**

5.11 Monte Carlo Tree Search


MCTS [254] is based on the application of Monte Carlo simulations to game-tree search. It rose to prominence with successes in games, notably, it powered AlphaGo [255] in 2016 by searching possible moves guided by policy and value networks. This, as well as the application to other board and video games, demonstrates the power of MCTS for sequential decision-making under uncertainty.

MCTS is a stochastic search algorithm that builds a decision tree by performing many random simulations. It is best known for finding good moves in game states, but it can be applied to any problem where we can simulate outcomes. The algorithm iteratively: (a) Selects a path from the root according to a heuristic (like UCT [256], which picks nodes with a high upper-confidence bound), (b) Expands a new node (a previously unvisited state) from the end of that path, (c) Simulates a random rollout from that new state to get an outcome (e.g., win or loss in a game, or some reward), and (d) Backpropagates the result up the tree to update the values of nodes and inform future selections. Repeating these simulations thousands of times concentrates the search on the most promising branches of the tree. In essence, MCTS uses random sampling to evaluate the potential of different action sequences, gradually biasing the search towards those with better average outcomes. In LLM reasoning, we can treat the generation of text as a decision process and use to explore different continuations. For example, at a given question (root), each possible next reasoning step or answer is an action; a simulation could mean letting the LLM continue to a final answer (perhaps with some randomness), and a reward could be whether the answer is correct. By doing this repeatedly, MCTS can identify which chain of thoughts or answers has the highest empirical success rate. The appeal of MCTS for reasoning is that it can handle large search spaces by sampling intelligently rather than exhaustively, and it naturally incorporates uncertainty and exploration.

 **Train verifiers to score intermediate steps (via Monte Carlo rollouts) instead of just final answers.**

Recent efforts have integrated MCTS with LLMs to tackle complex reasoning and decision-making tasks. One example is using MCTS for query planning: Monte Carlo Thought Search [257], where an LLM is guided to ask a series of sub-questions to find an answer. Jay et al. [257] used an MCTS-based algorithm called ‘Monte Carlo Reasoner’ that treats the LLM as an environment: each node is a prompt (state) and each edge is an action (e.g., a particular question to ask or step to take), and random rollouts are used to evaluate outcomes. This approach allowed the system to efficiently explore a space

of possible reasoning paths and pick a high-reward answer path, outperforming naive sampling in a scientific Q&A task. Similarly, MCTS has been applied to code generation with LLMs [258] – the algorithm explores different code paths (using the model to propose code completions and test them) to find a correct solution. Another line of work ensembles multiple LLMs with MCTS, treating each model’s output as a branch and using a reward model to simulate outcomes [259]. Early results show that MCTS-based reasoning can solve problems that single-pass or greedy methods often miss, although with more compute [74]. The downside is that MCTS can be significantly slower than straightforward sampling or beam search, which recent research is addressing by improving efficiency (e.g., by state merging [87]). In general, MCTS brings the strength of planning algorithms to LLM inference and enables an LLM to ‘look ahead’ through simulated rollouts and make more informed reasoning choices, much like it has done for AI in gameplay.

 **Test-time compute is not a 1-to-1 replacement for pretraining but, offers a viable alternative in many cases.**

5.12 Chain-of-Action-Thought reasoning

LLMs excel in reasoning tasks but rely heavily on external guidance (e.g., verifiers) or extensive sampling at inference time. Existing methods like CoT [8] lack mechanisms for self-correction and adaptive exploration, limiting their autonomy and generalization. Satori [260] introduced a two-stage training paradigm, which works by initially tuning the model’s output format and then enhancing its reasoning capabilities through self-improvement. In Stage 1 (Format Tuning), the model is exposed to a large set of 10K synthetic trajectories generated by a multi-agent framework comprising a generator, a critic, and a reward model. This supervised fine-tuning helps the model to produce outputs in specific reasoning format using meta-action tokens, although it may still have difficulty generalizing beyond these examples. In Stage 2 (Self-Improvement via RL), the model employs PPO with a Restart and Explore strategy [260], which allows it to restart from intermediate steps, whether they were correct or not, to refine its reasoning process. The model receives rewards based on a combination of rule-based correctness, reflection bonuses, and preference-based Outcome Reward Model feedback explained in § 5.9, thereby incentivizing the allocation of more computational resources to tougher problems and enabling extended reasoning during testing for complex tasks.

Multi-agent frameworks and advanced fine-tuning strategies are increasingly being explored to enhance reasoning in LLMs. Multi-Agent LLM Training (MALT) [261] introduces a structured approach where generation, verification, and refinement steps are distributed across specialized agents, allowing for iterative self-correction and improved reasoning chains. Similarly, optimizing preference alignment remains a crucial challenge in ensuring both safety and helpfulness in LLMs [262]. Approaches like Bi-Factorial Preference Optimization (BFPO) [263] reframe RLHF objectives into a single supervised learning task, reducing human intervention while maintaining robust alignment. Beyond text-based reasoning, multimodal approaches like Multimodal Visualization-of-

Thought (MVoT) [264] extend CoT prompting by incorporating visual representations, significantly enhancing performance in spatial reasoning tasks. These advancements highlight the growing need for structured multi-agent collaboration, safety-aware optimization, and multimodal reasoning to address fundamental limitations in LLM reasoning [265, 266, 267].

5.13 Pretraining vs. Test-Time Scaling

Pretraining and TTS are two distinct strategies for improving LLM performance, each with different tradeoffs in computational cost and effectiveness. Pretraining involves scaling model parameters or increasing training data to enhance capabilities, requiring substantial upfront computational investment [3]. In contrast, TTS optimizes inference-time compute (such as iterative refinements, search-based decoding, or adaptive sampling), allowing performance improvements without modifying the base model.

From a performance vs. cost perspective, TTS achieves results comparable to a model 14× larger on easy to intermediate tasks (e.g., MATH benchmarks), while reducing inference costs by 4× fewer FLOPs in compute-intensive scenarios [268]. However, pretraining remains superior for the hardest tasks or when inference compute constraints are high, as larger pretrained models inherently encode deeper reasoning capabilities.



A smaller model with test-time compute can outperform a 14× larger model on easy/intermediate questions, when inference tokens (Y) are limited (e.g., self-improvement settings).

In terms of use cases, TTS is useful for scenarios with flexible inference budget or when base models already exhibit reasonable competence in the task. Conversely, pretraining is essential for tasks requiring fundamentally new capabilities (e.g., reasoning on novel domains) where inference-time optimizations alone may not suffice.

There are notable tradeoffs between the two approaches. TTS reduces upfront training costs, making it attractive for flexible, on-the-go optimization, but requires dynamic compute allocation at inference. Pretraining, on the other hand, incurs high initial costs but guarantees consistent performance without additional runtime overhead, making it ideal for large-scale API deployments or latency-sensitive applications. Overall, TTS and pretraining are complementary in nature. Future LLM systems may adopt a hybrid approach, where smaller base models are pretrained with essential knowledge, while TTS dynamically enhances responses through adaptive, on-demand computation. This synergy enables more cost-effective and efficient large-scale model deployment.



Choose pretraining for foundational capabilities and test-time scaling for accurate context-aware refinement.

6 Benchmarks for LLM Post-training Evaluation

To evaluate the success of LLM post-training phases, a diverse set of benchmarks have been proposed covering multiple domains: reasoning tasks, alignment, multilinguality,

TABLE 3: Comprehensive Overview of Reasoning, RL Alignment, and Multilingual Datasets. Here, pointwise and pairwise refer to different methods of evaluating model performance across various tasks.

Datasets	Domain	Type	#Samples	Evaluation Criteria
Reasoning Benchmarks				
MATH [269]	Math Reasoning	Pointwise	7,500	Step-by-step solutions
GSM8K [270]	Math Reasoning	Pointwise	8.5K	Multi-step reasoning
MetaMathQA [271]	Math Reasoning	Pointwise	40K+	Self-verification, FOBAR
WorldTree V2 [272]	Science QA	Pointwise	1,680	Multi-hop explanations
PangeaBench [273]	Multimodal Reasoning	Pairwise	47 Langs.	Cultural understanding
MMMU [274]	Science/Math	Pointwise	College-Level	Physics, Chemistry, Bilingual
TruthfulQA [275]	QA/Reasoning	Pointwise	N/A	Truthfulness
MathInstruct [276]	Math Reasoning	Pointwise	262K	Correctness
MMLU [277, 278]	Multitask Reasoning	Pointwise	57 Tasks	Broad knowledge evaluation
MMLU-Fairness [277]	Fairness/Reasoning	Pointwise	N/A	Bias/Equity Analysis
DR0P [279]	Reading/Reasoning	Pointwise	96K	Discrete reasoning over paragraphs
BBH [175]	Hard Reasoning	Pairwise	N/A	Complex logical problem-solving
VRG-Bench [187]	Multimodal Reasoning	Pairwise	N/A	Visual Reasoning and Classification
RL Alignment Benchmarks				
HelpSteer [280]	RL Alignment	Pairwise	37K+	Multi-attribute scoring
Anthropic HH-RLHF [121]	RL Alignment	Pairwise	42.5K	Harmlessness alignment
UltraFeedback [281]	RL Alignment	Pairwise	64K	Instruction-following, Truthfulness
D4RL [282]	RL/Control	Pointwise	N/A	Offline RL across domains
Meta-World [283]	RL/Control	Pointwise	N/A	Multi-task robotic RL
MineRL [284]	RL/Games	Pairwise	N/A	Imitation learning, rewards
Multilingual Evaluation				
CulturaX [285]	Multilingual	Pointwise	6.3T	Deduplication, Quality
Pangeas [286]	Multilingual	Pointwise	6M	Multilingual instructions
TyDiQA [287]	Multilingual	Pointwise	N/A	Cross-lingual QA
XGLUE [288]	Multilingual	Pointwise	N/A	Cross-lingual language tasks
MM-Eval [289]	Multilingual	Pairwise	4,981	Task-oriented multilingual QA
ALM-Bench [289]	Multilingual QA	Pointwise	N/A	Multilingual Evaluation
Dialogue and Search Benchmarks				
BigBench [290]	General Comprehension	Pointwise	200+ Tasks	Broad multi-domain evaluation
Chatbot Arena [291]	Comprehension	Pairwise	33K	User preference
MTBench [291]	Comprehension	Pairwise	3K	Multi-turn conversations
RewardBench [167]	Comprehension	Pairwise	2,998	User preference
General Comprehension Benchmarks				
ConvAI2 [292]	Dialogue	Pointwise	N/A	Engagingness, Consistency
MultiWOZ [293]	Dialogue	Pointwise	N/A	Task success, Coherence
Trec DL21&22 [294, 295]	Search	Pointwise	1,549/2,673	Relevance scoring
BEIR [296]	Search	Pointwise	18 Datasets	Information retrieval
Story & Recommendation Benchmarks				
HANNA [297]	Story	Pointwise	1,056	Relevance, Coherence, Complexity
StoryER [298]	Story	Pairwise	100K	User preference-based ranking
PKU-SafRLHF [299]	Values	Pairwise	83.4K	Helpfulness, Harmlessness
CValue [300]	Values	Pairwise	145K	Safety, Responsibility
NaturalInst. [301, 302]	Instruction Tuning	Pointwise	1,600+	Instruction-following evaluation

general comprehension, and dialogue and search tasks. A well-structured evaluation framework ensures a comprehensive understanding of an LLM strengths, and limitations across various tasks. These benchmarks play a crucial role in LLM post-processing stages, where models undergo fine-tuning, calibration, alignment, and optimization to improve response accuracy, robustness, and ethical compliance. Next, we explain the main benchmark groups. Table 3 provides an overview of key datasets categorized under these benchmark groups.

Reasoning Benchmarks. These benchmarks assess LLMs on their ability to perform logical, mathematical, and scientific reasoning. Mathematical reasoning datasets like MATH [269], GSM8K [270], and MetaMathQA [271] test models on problem-solving, multi-step arithmetic, and theorem-based problem formulations. Scientific and multimodal reasoning benchmarks such as WorldTree V2 [272] and MMMU [274] evaluate knowledge in physics, chemistry, and multimodal understanding, which are crucial for fact-checking and verification processes in LLM-generated responses. Additionally, datasets like PangeaBench [273] extend reasoning tasks into multilingual and cultural domains, enabling models to refine cross-lingual reasoning. These benchmarks help determine how well models can process structured knowledge and apply logical deductions.

RL Alignment Benchmarks. RL alignment benchmarks are central to LLM alignment and post-training optimization. They refine response generation, ethical constraints, and user-aligned outputs through RLHF. Datasets such as HelpSteer [280] and UltraFeedback [281] evaluate models based on multi-attribute scoring and alignment with user instructions. Anthropic’s HH-RLHF [121] explores how well mod-

els learn human preference optimization through reinforcement learning with human feedback. D4RL [282] and Meta-World [283] focus on robotic control and offline RL, which have implications for autonomous model decision-making. MineRL [284] extends RL testing into complex environments such as Minecraft-based interactions, useful for training LLMs in adaptive decision-making settings.

Multilingual Evaluation. Multilingual benchmarks are essential for LLM post-processing in cross-lingual generalization, translation adaptation, and fine-tuning for low-resource languages. CulturaX [285] and PangeaIns [286] evaluate tokenization, translation, and instruction-following in over 150 languages, ensuring fairness and diversity in model outputs. TydiQA [287] and MM-Eval [289] target bilingual and task-oriented multilingual evaluation, enabling improvements in LLM fine-tuning. These datasets ensure that LLMs are not just English-centric but optimized for multilingual adaptability.

General Comprehension Benchmarks. General comprehension benchmarks contribute to model fine-tuning, response coherence, and preference optimization. Datasets such as Chatbot Arena [291], MTBench [291], and RewardBench [167] test user preference modeling and conversational fluency, crucial for LLM response ranking and re-ranking methods. BigBench [290] evaluates broad multi-domain comprehension, while MMLU [277, 278] measures correctness and informativeness. These datasets help in refining LLM fluency, factual correctness, and open-ended response generation.

Dialogue and Search Benchmarks. Dialogue and search benchmarks play a key role in optimizing LLM retrieval-based responses, multi-turn coherence, and information retrieval accuracy. Datasets such as ConvAI2 [292] and MultiWOZ [293] evaluate multi-turn conversational models, essential for dialogue history tracking and adaptive response fine-tuning. For search relevance assessment, BEIR [296] provides large-scale human-annotated judgments for retrieval fine-tuning, ensuring LLMs generate and rank responses effectively. TREC DL21/22 [294, 295] contributes to document relevance ranking and fact retrieval.

7 Future Directions

We gathered all papers related to post-training methods in LLMs and analyzed their trends, as shown in Figure 7. Application of RL techniques [303, 57, 40] for refining the LLMs have a noticeable increase in prominence since 2020 (Figure 7a), emphasizing the demand for **interactive approaches** such as human-in-the-loop [35, 304] reinforcement and scalability [111, 82, 305]. At the same time, **reward modeling** [306, 166, 167] (Figure 7b) has seen a steady rise in interest due to the emergence of self-rewarding language models, yet the field still struggles with **reward hacking** [307, 308] and the design of robust [309], failure-aware reward functions beyond reward hacking [310]. **Decoding and search** (Figure 7c) methods include tree-of-thoughts [84] and Monte Carlo [311, 257] strategies aiming to enhance model reasoning through iterative self-critique [312, 304, 29], but these techniques also demand reliable uncertainty estimators to prevent excessive computational overhead [313, 111]. Safety [299, 314, 315], robustness [316], and interpretability [317, 318, 319] have likewise become central concerns (Figure 7d), motivating the development of bias-aware [320, 321]

and uncertainty-aware [322] RL methods beyond correlation with human uncertainty [323] that safeguard user trust and prevent adversarial attacks. Another crucial area involves **personalization** [324, 325] and **adaptation** [193] (Figure 7e), where efforts to tailor LLMs for specific domains must be balanced against risks to privacy [326], particularly when enterprise data or sensitive personal information is involved.

In parallel, **process** [327, 328] vs. **outcome reward optimization** [329] (Figure 7f) remains an open question: while process-based rewards help guide incremental improvements, outcome-focused metrics are simpler but may not capture crucial intermediate decision-making steps. Beyond reward structure, **fine-tuning LLMs** on new tasks still encounter issues like **catastrophic forgetting** [330] and potential data leakage [331, 332], underscoring the need for parameter-efficient methods [60] and privacy-preserving strategies such as differential privacy [333] and federated learning [334]. Human feedback, while central to alignment, is inherently costly and limited in scope; methods like **Constitutional AI** [53] and RLAIIF [95] seek to automate parts of this oversight, though they introduce fresh concerns about bias calibration [335] and model self-consistency [184]. Finally, test-time scaling [111] and **dynamic reasoning** [336] frameworks pose further challenges: models must learn when to allocate more computation for complex queries, how to adapt verification modules [337] efficiently, and how to maintain robust performance even when facing adversarial inputs. These converging research directions—spanning reward modeling, decoding strategies, interpretability, personalization, and safe fine-tuning—highlight the multifaceted role of RL in LLMs and collectively shape the future trajectory of large-scale language model development. Below, we delve into some of these directions in greater detail.

Fine-tuning challenges. Fine-tuning remains one of the most direct post-training methods to adapt LLMs to specific tasks or domains, yet it faces several open challenges. One fundamental issue is catastrophic forgetting – when updating an LLM on new data causes it to lose or degrade previously learned capabilities. Even advanced PEFT methods like LoRA [60], which greatly reduce the number of trainable weights, do not fully solve this problem [330]. Future work can explore better continual learning strategies and regularization techniques so that models can acquire new skills without erasing old ones. For example, new fine-tuning algorithms (e.g. CURLoRA [330]) explicitly aim to stabilize training and preserve prior knowledge while adding new tasks. Promising research directions include curriculum-based fine-tuning [338] (introducing new facts gradually or in context with known facts) and hybrid training that combines retrieval or external knowledge bases. For instance, rather than solely adjusting the model’s weights, one could fine-tune LLMs to consult a knowledge repository or perform tool use (such as database queries or computations) when faced with queries outside their original training distribution [339, 340]. This retrieval-augmented fine-tuning [341] could let models incorporate fresh information at inference time, reducing the need to overwrite their internal weights with new facts. Another approach is training models to explicitly represent uncertainty about new knowledge, thereby enabling them to say ‘I don’t know’ or defer to an external source if a query concerns content not seen in pre-training. By blending weight updates with external knowledge integration, future fine-tuned LLMs will

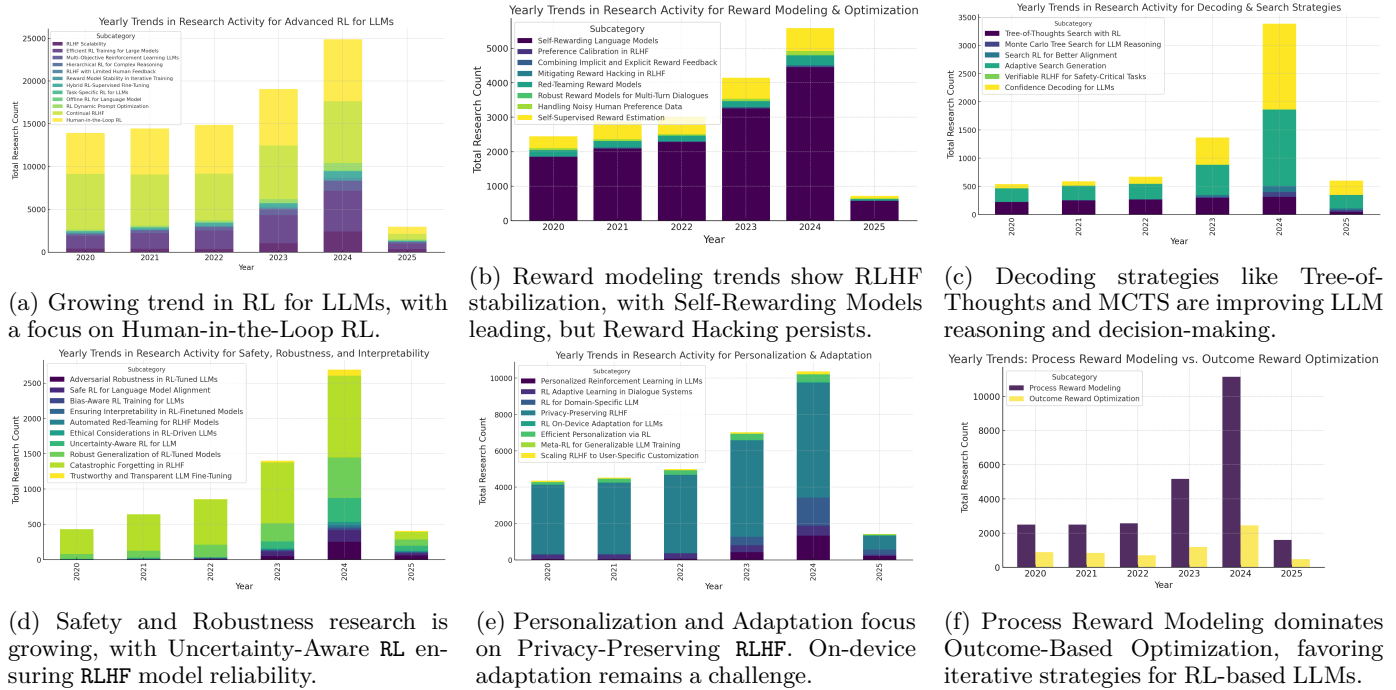


Fig. 7: Yearly Trends in RL specific post-training methods for LLMs and emerging research directions.

maintain higher factual accuracy and lower hallucination rates on emerging information.

Safe Fine-tuning. From an ethical and safety perspective, fine-tuning raises important open research questions. Fine-tuning data often contains sensitive or proprietary information [326], which can lead to privacy risks if the model memorizes and later regurgitates that data. A recent comprehensive survey [342] highlights vulnerabilities in the fine-tuning stage, such as membership inference attacks (detecting if a specific record was in the fine-tuning set) and data extraction (recovering parts of the fine-tuning data from the model’s outputs). Mitigating these risks is an open problem: methods like differential privacy fine-tuning [333] (adding noise to the weight updates) and federated fine-tuning (where data never leaves user devices and only aggregated updates are sent to the model) are being actively explored. However, these methods often come at the cost of model utility or require careful calibration to avoid degrading performance.

Limitations of Human Feedback. Human feedback is costly and subjective. One promising avenue to address the limitations of human feedback is using AI feedback and automation to assist or replace human evaluators. Constitutional AI [53], introduced by Anthropic, is a notable example: instead of relying on extensive human feedback for every harmful or helpful behavior, the model is guided by a set of written principles (a ‘constitution’) and is trained to critique and refine its own responses using another AI model as the judge [343]. Emerging directions here include RLAIIF [95] and other semi-automated feedback techniques [344]: using strong models to evaluate or guide weaker models, or even having multiple AI agents debate a question and using their agreement as a reward signal [345, 346]. Such AI-aided feedback could vastly scale the tuning process and help overcome the bottleneck of limited human expert time. However, it raises new theoretical questions: how do we ensure the AI judge is

itself aligned and correct? There is a risk of feedback loops or an echo chamber of biases if the automated preferences are flawed. An open gap is the creation of robust AI feedback systems that are calibrated to human values (perhaps periodically ‘grounded’ by human oversight or by a diverse set of constitutional principles). The blending of human and AI feedback in a hierarchical scheme could provide a scalable yet reliable RL paradigm for LLMs.

Test-time scaling challenges. Open challenges in TTS revolve around how to orchestrate the inference-time processes efficiently and reliably. A key question is how much computing is enough for a given query, and how to determine this on the fly? Using less resources can result in mistakes, but using too much is inefficient and could introduce inconsistencies. Recent research by Snell et al. [83] tackled it by proposing a unified framework with a ‘Proposer’ and a ‘Verifier’ to systematically explore and evaluate answers. In their framework, the Proposer (usually the base LLM) generates multiple candidate solutions, and the Verifier (another model or a heuristic) judges and selects the best. The optimal strategy can vary by problem difficulty: for easier queries, generating many answers in parallel and picking the top might be sufficient, whereas for harder problems, sequential, step-by-step reasoning with verification at each step works better. An important future direction is building adaptive systems where the LLM dynamically allocates computation based on an estimate of the question’s complexity. This idea connects to meta-cognition in AI [314], enabling models to have a sense of what they don’t know or what deserves more thought. Developing reliable confidence metrics or difficulty predictors for LLMs is an open research area, but progress here would make TTS far more practical i.e., the model would only ‘slow down and think’ when necessary, much like a human spending extra time on a hard problem. Additionally, By reframing

inference-time scaling as a probabilistic inference problem and employing particle-based Monte Carlo methods [?], the small models achieved o1 level accuracy in only 32 rollouts, a 4–16x improvement in scaling efficiency across various mathematical reasoning tasks. Recent study [347] shows distilling test-time computations into synthetic training data creates synergistic pretraining benefits which can also be further explored.

Reward Modeling and Credit Assignment. Current RL approaches suffer from reward misgeneralization, where models over-optimize superficial proxy metrics rather than genuine reasoning quality. The sparse nature of terminal rewards in multi-step tasks increases credit assignment challenges, particularly in long-horizon reasoning scenarios. Traditional methods like DPO require inefficient pairwise preference data and fail to utilize failure trajectories effectively. Hybrid reward models can be investigated by integrating process supervision with outcome-based rewards using contrastive stepwise evaluation [348]. This approach enables a more granular assessment of intermediate decision-making steps while aligning with long-term objectives. Recent work [171] suggests step-level policy optimization could improve value function accuracy while maintaining safety constraints. Dynamic credit assignment mechanisms can be explored through temporal difference learning adapted for transformers [349, 350]. Such adaptations may enhance the model’s ability to capture long-range dependencies and optimize reward propagation over extended sequences. Failure-aware training strategies can be developed by incorporating negative examples into the RL loop via adversarial data augmentation [351]. This can improve model robustness by systematically exposing it to challenging scenarios and encouraging more resilient policy learning.

Efficient RL Training and Distillation. Current RL methods for LLMs require prohibitive computational resources [352] while often underperforming knowledge distillation techniques [93]. This inefficiency limits scalability and practical deployment, as distilled models frequently surpass RL-trained counterparts despite requiring less training overhead. Additionally, pure RL approaches struggle to balance language quality with reasoning improvement [97, 93], creating a performance ceiling.

The development of hybrid frameworks that initialize RL policies with distilled knowledge from large models, combining the exploratory benefits of RL with the stability of supervised learning is an interesting direction. Similarly, curriculum sampling strategies that progressively increase task complexity while using distillation to preserve linguistic coherence can also help. PEFT methods [60] can be leveraged during RL updates to maintain base capabilities while enhancing reasoning.



Integration: Combining PRM-guided tree search with online distillation achieves 4× efficiency gains over baseline methods, while maintaining 94% solution accuracy on MATH dataset.

Privacy-Preserving Personalization. Customizing models for enterprise and individual use cases raises the risk of exposing private training data through memorization, making privacy-preserving [326] adaptation essential. Promising solutions include homomorphic instruction tuning [353], which processes encrypted user queries while maintaining end-to-end

encryption during inference; differential privacy via reward noising [354], which introduces mathematically bounded noise into RLHF preference rankings during alignment; and federated distillation, which aggregates knowledge from decentralized user-specific models without sharing raw data.

Collaborative Multi-Model Systems. As single-model [355, 356, 357] scaling approaches physical limits, alternative paradigms such as multi-agent LLM collaboration [358, 359, 178] become necessary. Researchers are investigating emergent communication protocols that train models to develop lossy compression “languages” for inter-model knowledge transfer such as GenAINet [360], robust ensembles where stress-test induced specialization drives automatic division of problem spaces based on failure analysis [361], and gradient-free synergy learning through evolutionary strategies designed to discover complementary model combinations without relying on backpropagation [362].

Multimodal RL Integration. Multimodal reinforcement learning [363, 49, 364] faces the obstacle of a combinatorial state explosion, especially in contexts exceeding 128k tokens. Pioneering methods to overcome this include hierarchical attention frameworks that employ modality-specific policies with cross-attention gating [365], adaptive truncation strategies that compress context while preserving critical reasoning segments [366], and flash curriculum approaches that leverage self-supervised [367, 368, 369] complexity prediction to facilitate progressive multimodal integration.

Efficient RL Training. Efficient RL training paradigms continue to be a critical research frontier as current methods exhibit significant sample inefficiency and computational overhead. Addressing issues like the overthinking [370, 371] phenomenon, where excessive reasoning chains waste valuable computation [145], requires approaches such as partial rollout strategies [372], adaptive length penalty mechanisms employing learned compression transformers, and hybrid architectures that combine MCTS with advanced RL optimizers. These innovations are essential for scaling RL to long-context tasks while minimizing wasted computational resources.

⚠ Overthinking Phenomenon: Analysis reveals 22% wasted computation is in reasoning chains exceeding optimal reasoning length.

RL methods exhibit sample inefficiency and computational overhead, particularly when scaling to contexts exceeding 128k tokens. The ‘overthinking’ phenomenon, where models generate excessively long reasoning chains, further reduces token efficiency and increases deployment costs [373]. Investigate partial rollout strategies with flash attention mechanisms for long-context processing. Develop length penalty mechanisms using learned compression transformers for iterative long2short distillation. Hybrid architectures combining MCTS [74] with GRPO [59] could enable better exploration-exploitation tradeoffs. Parallel work by Xie et. al. [74] demonstrates promising results through adaptive tree search pruning. Several open challenges persist in the field. Uncertainty propagation remains problematic as current confidence estimators add approximately 18% latency overhead, while catastrophic forgetting results in a degradation of 29% of base capabilities during RL fine-tuning [374]. Moreover, benchmark

saturation is an issue, with MMLU scores correlating poorly ($r = 0.34$) with real-world performance [375].

▲ Adversarial Vulnerabilities: Stress tests reveal a high success rate on gradient-based prompt injections.

8 Conclusion

This survey and tutorial provides a systematic review of post-training methodologies for LLMs, focusing on fine-tuning, reinforcement learning, and scaling. We analyze key techniques, along with strategies for improving efficiency and alignment with human preferences. Additionally, we explore the role of RL in enhancing LLMs through reasoning, planning, and multi-task generalization, categorizing their functionalities within the agent-environment paradigm. Recent advancements in reinforcement learning and test-time scaling have significantly improved LLMs reasoning capabilities, enabling them to tackle increasingly complex tasks. By consolidating the latest research and identifying open challenges, we aim to guide future efforts in optimizing LLMs for real-world applications.

References

- [1] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017. 1
- [2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020. 1
- [3] Z. Yang, "Xlnet: Generalized autoregressive pretraining for language understanding," *arXiv preprint arXiv:1906.08237*, 2019. 1, 3, 19
- [4] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. 1, 2, 3
- [5] Z. Lan, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019. 1
- [6] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020. 1
- [7] P. Verga, S. Hofstatter, S. Althammer, Y. Su, A. Piktus, A. Arkhangorodsky, M. Xu, N. White, and P. Lewis, "Replacing judges with juries: Evaluating llm generations with a panel of diverse models," *arXiv preprint arXiv:2404.18796*, 2024. 1
- [8] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al., "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24824–24837, 2022. 1, 2, 3, 11, 12, 15, 18
- [9] C. Wang, Y. Deng, Z. Lyu, L. Zeng, J. He, S. Yan, and B. An, "Q*: Improving multi-step reasoning for llms with deliberative planning," *arXiv preprint arXiv:2406.14283*, 2024. 1
- [10] Y. Wu, X. Han, W. Song, M. Cheng, and F. Li, "Mindmap: Constructing evidence chains for multi-step reasoning in large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 19270–19278, 2024. 1
- [11] Y. Ding, Z. Wang, W. Ahmad, H. Ding, M. Tan, N. Jain, M. K. Ramanathan, R. Nallapati, P. Bhatia, D. Roth, et al., "Cross-codeeval: A diverse and multilingual benchmark for cross-file code completion," *Advances in Neural Information Processing Systems*, vol. 36, 2024. 1
- [12] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altschmidt, S. Altman, S. Anadkat, et al., "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023. 1
- [13] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al., "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024. 1, 5
- [14] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, et al., "Gemma 2: Improving open language models at a practical size," *arXiv preprint arXiv:2408.00118*, 2024. 1, 5
- [15] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, et al., "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023. 1, 5
- [16] A. Liu, B. Feng, B. Wang, B. Wang, B. Liu, C. Zhao, C. Deng, C. Ruan, D. Dai, D. Guo, et al., "Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model," *arXiv preprint arXiv:2405.04434*, 2024. 1, 2, 5, 6
- [17] M. Abdin, J. Anreja, H. Awadalla, A. Awadallah, A. A. Awan, N. Bach, A. Bahree, A. Bakhtiari, J. Bao, H. Behl, et al., "Phi-3 technical report: A highly capable language model locally on your phone," *arXiv preprint arXiv:2404.14219*, 2024. 1, 5
- [18] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," *arXiv preprint arXiv:1805.04833*, 2018. 1
- [19] C. Chhun, P. Colombo, C. Clavel, and F. M. Suchanek, "Of human criteria and automatic metrics: A benchmark of the evaluation of story generation," *arXiv preprint arXiv:2208.11646*, 2022. 1
- [20] S. Arif, S. Farid, A. H. Azeemi, A. Athar, and A. A. Raza, "The fellowship of the llms: Multi-agent workflows for synthetic preference optimization dataset generation," *arXiv preprint arXiv:2408.08688*, 2024. 1
- [21] S. Ye, Y. Jo, D. Kim, S. Kim, H. Hwang, and M. Seo, "Selfee: Iterative self-revising llm empowered by self-feedback generation," *Blog post*, 2023. 1
- [22] M. Musolesi, "Creative beam search: Llm-as-a-judge for improving response generation," *ICCC*, 2024. 1
- [23] J. Ren, Y. Zhao, T. Vu, P. J. Liu, and B. Lakshminarayanan, "Self-evaluation improves selective generation in large language models," in *Proceedings on*, pp. 49–64, PMLR, 2023. 1
- [24] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020. 1, 2
- [25] Y. Lai, C. Li, Y. Wang, T. Zhang, R. Zhong, L. Zettlemoyer, W.-t. Yih, D. Fried, S. Wang, and T. Yu, "Ds-1000: A natural and reliable benchmark for data science code generation," in *International Conference on Machine Learning*, pp. 18319–18345, PMLR, 2023. 1
- [26] B. Zhu, E. Frick, T. Wu, H. Zhu, K. Ganesan, W.-L. Chiang, J. Zhang, and J. Jiao, "Starling-7b: Improving helpfulness and harmlessness with rlai," in *First Conference on Language Modeling*, 2024. 1, 5
- [27] D. Paul, M. Ismayilzada, M. Peyrard, B. Borges, A. Bosselut, R. West, and B. Faltings, "Refiner: Reasoning feedback on intermediate representations," *arXiv preprint arXiv:2304.01904*, 2023. 1
- [28] Y. Xie, K. Kawaguchi, Y. Zhao, J. X. Zhao, M.-Y. Kan, J. He, and M. Xie, "Self-evaluation guided beam search for reasoning," *Advances in Neural Information Processing Systems*, vol. 36, 2024. 1
- [29] H. Ye and H. T. Ng, "Self-judge: Selective instruction following with alignment self-evaluation," *arXiv preprint arXiv:2409.00935*, 2024. 1, 20
- [30] Z. Luo, H. Wu, D. Li, J. Ma, M. Kankanhalli, and J. Li, "Videoautoarena: An automated arena for evaluating large multimodal models in video analysis through user simulation," *arXiv preprint arXiv:2411.13281*, 2024. 1
- [31] S. Deng, W. Zhao, Y.-J. Li, K. Wan, D. Miranda, A. Kale, and Y. Tian, "Efficient self-improvement in multimodal large language models: A model-level judge-free approach," *arXiv preprint arXiv:2411.17760*, 2024. 1, 2
- [32] T. Xiong, X. Wang, D. Guo, Q. Ye, H. Fan, Q. Gu, H. Huang, and C. Li, "Llava-critic: Learning to evaluate multimodal models," *arXiv preprint arXiv:2410.02712*, 2024.
- [33] D. Chen, R. Chen, S. Zhang, Y. Liu, Y. Wang, H. Zhou, Q. Zhang, Y. Wan, P. Zhou, and L. Sun, "Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language bench-

- mark,” *arXiv preprint arXiv:2402.04788*, 2024. 1
- [34] T. Hagendorff, S. Fabi, and M. Kosinski, “Human-like intuitive behavior and reasoning biases emerged in large language models but disappeared in chatgpt,” *Nature Computational Science*, vol. 3, no. 10, pp. 833–838, 2023. 1
- [35] Q. Pan, Z. Ashktorab, M. Desmond, M. S. Cooper, J. Johnson, R. Nair, E. Daly, and W. Geyer, “Human-centered design recommendations for llm-as-a-judge,” *arXiv preprint arXiv:2407.03479*, 2024. 1, 20
- [36] G. H. Chen, S. Chen, Z. Liu, F. Jiang, and B. Wang, “Humans or llms as the judge? a study on judgement biases,” *arXiv preprint arXiv:2402.10669*, 2024. 1
- [37] A. Newell, “Human problem solving,” *Upper Saddle River/Prentice Hall*, 1972. 1, 16
- [38] X. Wang, H. Kim, S. Rahman, K. Mitra, and Z. Miao, “Human-llm collaborative annotation through effective verification of llm labels,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 1–21, 2024. 1
- [39] R. OpenAI, “Gpt-4 technical report. arxiv 2303.08774,” *View in Article*, vol. 2, no. 5, 2023. 1, 5, 12
- [40] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, *et al.*, “DeepSeek-R1: Incentivizing reasoning capability in llms via reinforcement learning,” *arXiv preprint arXiv:2501.12948*, 2025. 1, 5, 11, 12, 20
- [41] M. T. Hicks, J. Humphries, and J. Slater, “Chatgpt is bullshit,” *Ethics and Information Technology*, vol. 26, no. 2, pp. 1–10, 2024. 1
- [42] N. Maleki, B. Padmanabhan, and K. Dutta, “Ai hallucinations: A misnomer worth clarifying,” 2024. 1
- [43] A. Bruno, P. L. Mazzeo, A. Chetouani, M. Tliba, and M. A. Kerkouri, “Insights into classifying and mitigating llms’ hallucinations,” *arXiv preprint arXiv:2311.08117*, 2023. 1
- [44] S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal, “Detecting hallucinations in large language models using semantic entropy,” *Nature*, vol. 630, no. 8017, pp. 625–630, 2024. 1
- [45] F. Leiser, S. Eckhardt, V. Leuthe, M. Knaeble, A. Maedche, G. Schwabe, and A. Sunyaev, “Hill: A hallucination identifier for large language models,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2024. 1
- [46] A. Gunjal, J. Yin, and E. Bas, “Detecting and preventing hallucinations in large vision language models,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 18135–18143, 2024. 1
- [47] S. Hao, Y. Gu, H. Ma, J. J. Hong, Z. Wang, D. Z. Wang, and Z. Hu, “Reasoning with language model is planning with world model,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023* (H. Bouamor, J. Pino, and K. Bali, eds.), pp. 8154–8173, Association for Computational Linguistics, 2023. 1
- [48] Y. Ye, Z. Huang, Y. Xiao, E. Chern, S. Xia, and P. Liu, “Limo: Less is more for reasoning,” 2025. 1, 3
- [49] C. Li, W. Wu, H. Zhang, Y. Xia, S. Mao, L. Dong, I. Vulić, and F. Wei, “Imagine while reasoning in space: Multimodal visualization-of-thought,” 2025. 1, 3, 22
- [50] T. Xia, B. Yu, Y. Wu, Y. Chang, and C. Zhou, “Language models can evaluate themselves via probability discrepancy,” *arXiv preprint arXiv:2405.10516*, 2024. 1
- [51] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, “Survey of hallucination in natural language generation,” *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023. 1, 3
- [52] H. He and W. J. Su, “A law of next-token prediction in large language models,” 2024. 2
- [53] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, *et al.*, “Constitutional ai: Harmlessness from ai feedback,” *arXiv preprint arXiv:2212.08073*, 2022. 2, 20, 21
- [54] interconnects.ai, “blob reinforcement fine-tuning.” (Accessed: 2025-12-6). 2
- [55] E. Lobo, C. Agarwal, and H. Lakkaraju, “On the impact of fine-tuning on chain-of-thought reasoning,” *arXiv preprint arXiv:2411.15382*, 2024. 2, 3
- [56] L. Trung, X. Zhang, Z. Jie, P. Sun, X. Jin, and H. Li, “Reft: Reasoning with reinforced fine-tuning,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7601–7614, 2024. 2
- [57] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, “Direct preference optimization: Your language model is secretly a reward model,” *Advances in Neural Information Processing Systems*, vol. 36, 2024. 2, 3, 6, 20
- [58] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, *et al.*, “Training language models to follow instructions with human feedback,” *Advances in neural information processing systems*, vol. 35, pp. 27730–27744, 2022. 2, 3, 5, 9, 13, 15
- [59] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, *et al.*, “Deepseekmath: Pushing the limits of mathematical reasoning in open language models,” *arXiv preprint arXiv:2402.03300*, 2024. 2, 3, 6, 8, 10, 12, 22
- [60] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021. 2, 13, 20, 22
- [61] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, “Self-rag: Learning to retrieve, generate, and critique through self-reflection,” *arXiv preprint arXiv:2310.11511*, 2023. 2
- [62] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, “Retrieval-augmented generation for large language models: A survey,” *arXiv preprint arXiv:2312.10997*, 2023. 2
- [63] Z. Hu, L. Song, J. Zhang, Z. Xiao, J. Wang, Z. Chen, and H. Xiong, “Rethinking llm-based preference evaluation,” *arXiv preprint arXiv:2407.01085*, 2024. 2
- [64] S. Yue, W. Chen, S. Wang, B. Li, C. Shen, S. Liu, Y. Zhou, Y. Xiao, S. Yun, X. Huang, *et al.*, “Disc-lawllm: Fine-tuning large language models for intelligent legal services,” *arXiv preprint arXiv:2309.11325*, 2023. 2
- [65] Y. Luo, Z. Yang, F. Meng, Y. Li, J. Zhou, and Y. Zhang, “An empirical study of catastrophic forgetting in large language models during continual fine-tuning,” *arXiv preprint arXiv:2308.08747*, 2023. 2
- [66] H. Li, J. Chen, W. Su, Q. Ai, and Y. Liu, “Towards better web search performance: Pre-training, fine-tuning and learning to rank,” 2023. 2
- [67] OpenAI, “Reinforcement fine-tuning.” (Accessed: 2025-12-6). 2
- [68] W. Zhang, Y. Deng, B. Liu, S. J. Pan, and L. Bing, “Sentiment analysis in the era of large language models: A reality check,” *arXiv preprint arXiv:2305.15005*, 2023. 2
- [69] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011. 2
- [70] Y. Wang, S. Wang, Y. Li, and D. Dou, “Recognizing medical search query intent by few-shot learning,” in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 502–512, 2022. 2
- [71] R. Luo, L. Sun, Y. Xia, T. Qin, S. Zhang, H. Poon, and T.-Y. Liu, “Biogpt: generative pre-trained transformer for biomedical text generation and mining,” *Briefings in bioinformatics*, vol. 23, no. 6, p. bbac409, 2022. 2, 13
- [72] O. Wysocki, M. Wysocka, D. Carvalho, A. T. Bogatu, D. M. Gusicuma, M. Delmas, H. Unsworth, and A. Freitas, “An llm-based knowledge synthesis and scientific reasoning framework for biomedical discovery,” *arXiv preprint arXiv:2406.18626*, 2024. 2
- [73] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017. 2, 5, 8, 10
- [74] Y. Xie, A. Goyal, W. Zheng, M.-Y. Kan, T. P. Lillicrap, K. Kawaguchi, and M. Shieh, “Monte carlo tree search boosts reasoning via iterative preference learning,” *arXiv preprint arXiv:2405.00451*, 2024. 2, 18, 22
- [75] M. Shanahan, K. McDonnell, and L. Reynolds, “Role play with large language models,” *Nature*, vol. 623, no. 7987, pp. 493–498, 2023. 2
- [76] T. Xu, E. Helenowski, K. A. Sankararaman, D. Jin, K. Peng, E. Han, S. Nie, C. Zhu, H. Zhang, W. Zhou, *et al.*, “The perfect blend: Redefining rlhf with mixture of judges,” *arXiv preprint arXiv:2409.20370*, 2024. 2
- [77] M. Cao, L. Shu, L. Yu, Y. Zhu, N. Wichers, Y. Liu, and L. Meng, “Beyond sparse rewards: Enhancing reinforcement

- learning with language model critique in text generation,” 2024. [2](#)
- [78] Y. Dubois, C. X. Li, R. Taori, T. Zhang, I. Gulrajani, J. Ba, C. Guestrin, P. S. Liang, and T. B. Hashimoto, “AlpacaFarm: A simulation framework for methods that learn from human feedback,” *Advances in Neural Information Processing Systems*, vol. 36, 2024. [2](#), [15](#)
- [79] L. Shani, A. Rosenberg, A. Cassel, O. Lang, D. Calandriello, A. Zipori, H. Noga, O. Keller, B. Piot, I. Szpektor, *et al.*, “Multi-turn reinforcement learning from preference human feedback,” *arXiv preprint arXiv:2405.14655*, 2024. [2](#)
- [80] Z. Li, Y. He, L. He, J. Wang, T. Shi, B. Lei, Y. Li, and Q. Chen, “Falcon: Feedback-driven adaptive long/short-term memory reinforced coding optimization system,” *arXiv preprint arXiv:2410.21349*, 2024. [2](#)
- [81] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, *et al.*, “Webgpt: Browser-assisted question-answering with human feedback,” *arXiv preprint arXiv:2112.09332*, 2021. [2](#), [11](#), [14](#)
- [82] L. Gao, J. Schulman, and J. Hilton, “Scaling laws for reward model overoptimization,” in *International Conference on Machine Learning*, pp. 10835–10866, PMLR, 2023. [2](#), [15](#), [20](#)
- [83] C. Snell, J. Lee, K. Xu, and A. Kumar, “Scaling llm test-time compute optimally can be more effective than scaling model parameters,” *arXiv preprint arXiv:2408.03314*, 2024. [2](#), [15](#), [21](#)
- [84] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, “Tree of thoughts: Deliberate problem solving with large language models,” *Advances in Neural Information Processing Systems*, vol. 36, 2024. [2](#), [14](#), [16](#), [20](#)
- [85] J. Jiang, D. He, and J. Allan, “Searching, browsing, and clicking in a search session: Changes in user behavior by task and over time,” in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pp. 607–616, 2014. [2](#)
- [86] Y. Xie, K. Kawaguchi, Y. Zhao, J. X. Zhao, M.-Y. Kan, J. He, and M. Xie, “Self-evaluation guided beam search for reasoning,” *Advances in Neural Information Processing Systems*, vol. 36, 2024. [2](#)
- [87] Y. Tian, B. Peng, L. Song, L. Jin, D. Yu, H. Mi, and D. Yu, “Toward self-improvement of llms via imagination, searching, and criticizing,” *arXiv preprint arXiv:2404.12253*, 2024. [2](#), [18](#)
- [88] K. Gandhi, D. H. J. Lee, G. Grand, M. Liu, W. Cheng, A. Sharma, and N. Goodman, “Stream of search (SoS): Learning to search in language,” in *First Conference on Language Modeling*, 2024. [2](#)
- [89] K. Yang, A. M. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. Prenger, and A. Anandkumar, “Leandojo: Theorem proving with retrieval-augmented language models,” 2023. [2](#)
- [90] C. Sun, S. Huang, and D. Pompili, “Retrieval-augmented hierarchical in-context reinforcement learning and hindsight modular reflections for task planning with llms,” 2024. [2](#)
- [91] E. Davis, “Testing gpt-4-o-preview on math and science problems: A follow-up study,” *arXiv preprint arXiv:2410.22340*, 2024. [2](#)
- [92] J. Y. Wang, N. Sukiennik, T. Li, W. Su, Q. Hao, J. Xu, Z. Huang, F. Xu, and Y. Li, “A survey on human-centric llms,” *arXiv preprint arXiv:2411.14491*, 2024. [3](#)
- [93] J. Wu, S. Yang, R. Zhan, Y. Yuan, L. S. Chao, and D. F. Wong, “A survey on llm-generated text detection: Necessity, methods, and future directions,” *Computational Linguistics*, pp. 1–65, 2025. [3](#), [14](#), [22](#)
- [94] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, *et al.*, “A survey on evaluation of large language models,” *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 3, pp. 1–45, 2024. [3](#)
- [95] H. Lee, S. Phatale, H. Mansoor, K. R. Lu, T. Mesnard, J. Ferret, C. Bishop, E. Hall, V. Carbune, and A. Rastogi, “Rlaif: Scaling reinforcement learning from human feedback with ai feedback,” 2023. [3](#), [9](#), [20](#), [21](#)
- [96] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, and Z. Sui, “A survey on in-context learning,” *arXiv preprint arXiv:2301.00234*, 2022. [3](#)
- [97] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, *et al.*, “A survey of large language models,” *arXiv preprint arXiv:2303.18223*, 2023. [3](#), [22](#)
- [98] S. Han, H. Schoelkopf, Y. Zhao, Z. Qi, M. Riddell, W. Zhou, J. Coady, D. Peng, Y. Qiao, L. Benson, L. Sun, A. Wardle-Solano, H. Szabo, E. Zubova, M. Burtell, J. Fan, Y. Liu, B. Wong, M. Sailor, A. Ni, L. Nan, J. Kasai, T. Yu, R. Zhang, A. R. Fabbri, W. Kryscinski, S. Yavuz, Y. Liu, X. V. Lin, S. Joty, Y. Zhou, C. Xiong, R. Ying, A. Cohan, and D. Radev, “Folio: Natural language reasoning with first-order logic,” 2024. [3](#)
- [99] Z. Xi, S. Jin, Y. Zhou, R. Zheng, S. Gao, T. Gui, Q. Zhang, and X. Huang, “Self-polish: Enhance reasoning in large language models via problem refinement,” 2024. [3](#)
- [100] A. Saparov and H. He, “Language models are greedy reasoners: A systematic formal analysis of chain-of-thought,” 2023. [3](#)
- [101] J. Liu, C. Wang, C. Y. Liu, L. Zeng, R. Yan, Y. Sun, Y. Liu, and Y. Zhou, “Improving multi-step reasoning abilities of large language models with direct advantage policy optimization,” *arXiv preprint arXiv:2412.18279*, 2024. [3](#)
- [102] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, “Large language models are zero-shot reasoners,” *Advances in neural information processing systems*, vol. 35, pp. 22199–22213, 2022. [3](#)
- [103] X. Deng, Y. Su, A. Lees, Y. Wu, C. Yu, and H. Sun, “Reasonbert: Pre-trained to reason with distant supervision,” *arXiv preprint arXiv:2109.04912*, 2021. [3](#)
- [104] T. Sawada, D. Paleka, A. Havrilla, P. Tadepalli, P. Vidas, A. Kranias, J. J. Nay, K. Gupta, and A. Komatsuzaki, “Arb: Advanced reasoning benchmark for large language models,” 2023. [3](#)
- [105] A. Radford, “Improving language understanding by generative pre-training,” 2018. [3](#)
- [106] I. J. Myung, “Tutorial on maximum likelihood estimation,” *Journal of mathematical psychology*, vol. 47, no. 1, pp. 90–100, 2003. [3](#)
- [107] Y. Shao, J. Mao, Y. Liu, W. Ma, K. Satoh, M. Zhang, and S. Ma, “Bert-pli: Modeling paragraph-level interactions for legal case retrieval,” in *IJCAI*, pp. 3501–3507, 2020. [3](#)
- [108] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, *et al.*, “Emergent abilities of large language models,” *arXiv preprint arXiv:2206.07682*, 2022. [3](#)
- [109] R. Bellman, “A markovian decision process,” *Journal of mathematics and mechanics*, pp. 679–684, 1957. [3](#), [10](#)
- [110] S. Hao, Y. Gu, H. Luo, T. Liu, X. Shao, X. Wang, S. Xie, H. Ma, A. Samavedhi, Q. Gao, Z. Wang, and Z. Hu, “Llm reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models,” 2024. [3](#)
- [111] J. Geiping, S. McLeish, N. Jain, J. Kirchenbauer, S. Singh, B. R. Bartoldson, B. Kailkhura, A. Bhatele, and T. Goldstein, “Scaling up test-time compute with latent reasoning: A recurrent depth approach,” 2025. [3](#), [20](#)
- [112] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe, “Let’s verify step by step,” *arXiv preprint arXiv:2305.20050*, 2023. [4](#)
- [113] G. Chen, M. Liao, C. Li, and K. Fan, “Step-level value preference optimization for mathematical reasoning,” *arXiv preprint arXiv:2406.10858*, 2024. [4](#)
- [114] K. Nguyen, H. Daumé III, and J. Boyd-Graber, “Reinforcement learning for bandit neural machine translation with simulated human feedback,” *arXiv preprint arXiv:1707.07402*, 2017. [4](#), [5](#)
- [115] G. Williams, “Substantive and adjectival law,” in *Learning the Law*, pp. 19–23, Law Book Company, Limited, 1982. [4](#)
- [116] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, “Sequence level training with recurrent neural networks,” 2016. [4](#), [5](#)
- [117] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, “Self-critical sequence training for image captioning,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7008–7024, 2017. [4](#), [5](#)
- [118] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002. [4](#)
- [119] R. Vedantam, C. L. Zitnick, and D. Parikh, “Cider: Consensus-based image description evaluation,” 2015. [4](#)
- [120] OpenAI, “Openai gpt-4.5 system card,” 2025. Accessed: 2025-02-28. [5](#)
- [121] Anthropic, “Claude 3.7 sonnet,” 2025. Accessed: 2025-02-26. [5](#), [19](#)

- [122] R. Team, A. Ormazabal, C. Zheng, C. d. M. d’Autume, D. Yagatama, D. Fu, D. Ong, E. Chen, E. Lamprecht, H. Pham, *et al.*, “Reka core, flash, and edge: A series of powerful multimodal language models,” *arXiv preprint arXiv:2404.12387*, 2024. 5
- [123] B. Adler, N. Agarwal, A. Aithal, D. H. Anh, P. Bhattacharya, A. Brundyn, J. Casper, B. Catanzaro, S. Clay, J. Cohen, *et al.*, “Nemotron-4 340b technical report,” *arXiv preprint arXiv:2406.11704*, 2024. 5
- [124] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, Étienne Goffinet, D. Hesslow, J. Lounay, Q. Malartic, D. Mazzotta, B. Noune, B. Pannier, and G. Penedo, “The falcon series of open language models,” 2023. 5
- [125] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Lu, *et al.*, “Qwen2. 5-coder technical report,” *arXiv preprint arXiv:2409.12186*, 2024. 5
- [126] A. Défossez, L. Mazaré, M. Orsini, A. Royer, P. Pérez, H. Jégou, E. Grave, and N. Zeghidour, “Moshi: a speech-text foundation model for real-time dialogue,” tech. rep., 2024. 5
- [127] Nexusflow, “Athene: An rlhf-enhanced language model,” 2024. Accessed: 2025-02-26. 5
- [128] R. Teknium, J. Quesnelle, and C. Guang, “Hermes 3 technical report,” *arXiv preprint arXiv:2408.11857*, 2024. 5
- [129] Z. AI, “Zed,” 2025. 500B, Zed AI, RLHF, Multi-modal, Open. 5
- [130] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, *et al.*, “Palm 2 technical report,” *arXiv preprint arXiv:2305.10403*, 2023. 5
- [131] Z. Cai, M. Cao, H. Chen, K. Chen, K. Chen, X. Chen, X. Chen, Z. Chen, Z. Chen, P. Chu, *et al.*, “Internlm2 technical report,” *arXiv preprint arXiv:2403.17297*, 2024. 5
- [132] S. Labs, “Supernova,” 2025. 220B, Supernova Labs, RLHF, Multi-modal, Open. 5
- [133] xAI, “Grok-3: The next generation ai model by xai,” tech. rep., xAI, 2025. Accessed: 2025-02-24. 5
- [134] P. Agrawal, S. Antoniak, *et al.*, “Pixtral 12b,” 2024. 5
- [135] MiniMax, A. Li, *et al.*, “Minimax-01: Scaling foundation models with lightning attention,” 2025. 5
- [136] A. A. G. Intelligence, “The amazon nova family of models: Technical report and model card,” *Amazon Technical Reports*, 2024. 5
- [137] Fujitsu and T. I. of Technology, “Fugaku-llm: The largest cpu-only trained language model,” *Fujitsu Research Press Release*, May 2024. 5
- [138] R. AI, “Nova: A family of ai models by rubik’s ai,” *Rubik’s AI Research*, October 2024. 5
- [139] OpenAI, “Openai o3 system card,” technical report, OpenAI, 2025. 5
- [140] M. R. Team *et al.*, “Introducing dbrx: a new state-of-the-art open llm,” *Mosaic AI Research*, 2024. 5
- [141] A. Köpf, Y. Kilcher, D. von Rütte, S. Anagnostidis, Z.-R. Tam, K. Stevens, A. Barhoum, N. M. Duc, O. Stanley, R. Nagyfi, S. ES, S. Suri, D. Glushkov, A. Dantuluri, A. Maguire, C. Schuhmann, H. Nguyen, and A. Mattick, “Openassistant conversations – democratizing large language model alignment,” 2023. 5
- [142] T. GLM, A. Zeng, B. Xu, B. Wang, C. Zhang, D. Yin, D. Zhang, D. Rojas, G. Feng, H. Zhao, *et al.*, “Chatglm: A family of large language models from glm-130b to glm-4 all tools,” *arXiv preprint arXiv:2406.12793*, 2024. 5
- [143] A. Bartolome, J. Hong, N. Lee, K. Rasul, and L. Tunstall, “Zephyr 141b a39b,” 2024. 5
- [144] O. Lieber, O. Sharir, B. Lenz, and Y. Shoham, “Jurassic-1: Technical details and evaluation,” *White Paper. AI21 Labs*, vol. 1, no. 9, pp. 1–17, 2021. 5
- [145] K. Team, A. Du, B. Gao, B. Xing, C. Jiang, C. Chen, C. Li, C. Xiao, C. Du, C. Liao, *et al.*, “Kimi k1. 5: Scaling reinforcement learning with llms,” *arXiv preprint arXiv:2501.12599*, 2025. 5, 22
- [146] M. Abidin, J. Aneja, H. Behl, S. Bubeck, R. Eldan, S. Gunasekar, M. Harrison, R. J. Hewett, M. Javaheripi, P. Kauffmann, J. R. Lee, Y. T. Lee, Y. Li, W. Liu, C. C. T. Mendes, A. Nguyen, E. Price, G. de Rosa, O. Saarikivi, A. Salim, S. Shah, X. Wang, R. Ward, Y. Wu, D. Yu, C. Zhang, and Y. Zhang, “Phi-4 technical report,” 2024. 5
- [147] C. Team, “Chameleon: Mixed-modal early-fusion foundation models,” *arXiv preprint arXiv:2405.09818*, 2024. 5
- [148] N. Dey, G. Gosal, Zhiming, Chen, H. Khachane, W. Marshall, R. Pathria, M. Tom, and J. Hestness, “Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster,” 2023. 5
- [149] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann, “Bloomberggpt: A large language model for finance,” 2023. 5
- [150] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre, “Training compute-optimal large language models,” 2022. 5
- [151] S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu, “Minimum risk training for neural machine translation,” 2016. 4, 5
- [152] V. Konda and J. Tsitsiklis, “Actor-critic algorithms,” *Advances in neural information processing systems*, vol. 12, 1999. 4
- [153] S. Bhatnagar, M. Ghavamzadeh, M. Lee, and R. S. Sutton, “Incremental natural actor-critic algorithms,” *Advances in neural information processing systems*, vol. 20, 2007. 4
- [154] V. Mnih, “Asynchronous methods for deep reinforcement learning,” *arXiv preprint arXiv:1602.01783*, 2016. 4, 5
- [155] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, “Reinforcement learning through asynchronous advantage actor-critic on a gpu,” *arXiv preprint arXiv:1611.06256*, 2016. 4, 5
- [156] S. M. Kakade, “A natural policy gradient,” *Advances in neural information processing systems*, vol. 14, 2001. 4
- [157] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuronlike adaptive elements that can solve difficult learning control problems,” *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983. 5
- [158] L. Yu, W. Zhang, J. Wang, and Y. Yu, “Seqgan: Sequence generative adversarial nets with policy gradient,” 2017. 5
- [159] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, *et al.*, “Qwen2. 5 technical report,” *arXiv preprint arXiv:2412.15115*, 2024. 5, 6
- [160] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization,” 2017. 6, 8, 9
- [161] N. Vieillard, T. Kozuno, B. Scherrer, O. Pietquin, R. Munos, and M. Geist, “Leverage the average: an analysis of kl regularization in reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 12163–12174, 2020. 6
- [162] S. Guo, B. Zhang, T. Liu, T. Liu, M. Khalman, F. Llinares, A. Rame, T. Mesnard, Y. Zhao, B. Piot, *et al.*, “Direct language model alignment from online ai feedback,” *arXiv preprint arXiv:2402.04792*, 2024. 6, 9, 10
- [163] C. An, M. Zhong, Z. Wu, Q. Zhu, X. Huang, and X. Qiu, “Colo: A contrastive learning based re-ranking framework for one-stage summarization,” *arXiv preprint arXiv:2209.14569*, 2022. 6
- [164] R. A. Bradley and M. E. Terry, “Rank analysis of incomplete block designs: I. the method of paired comparisons,” *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952. 6
- [165] R. L. Plackett, “The analysis of permutations,” *Journal of the Royal Statistical Society C: Applied Statistics*, vol. 24, no. 2, pp. 193–202, 1975. 6
- [166] A. Setlur, C. Nagpal, A. Fisch, X. Geng, J. Eisenstein, R. Agarwal, A. Agarwal, J. Berant, and A. Kumar, “Rewarding progress: Scaling automated process verifiers for llm reasoning,” *arXiv preprint arXiv:2410.08146*, 2024. 7, 20
- [167] N. Lambert, V. Pyatkin, J. Morrison, L. Miranda, B. Y. Lin, K. Chandu, N. Dziri, S. Kumar, T. Zick, Y. Choi, *et al.*, “Rewardbench: Evaluating reward models for language modeling,” *arXiv preprint arXiv:2403.13787*, 2024. 7, 19, 20
- [168] J. Hong, N. Lee, and J. Thorne, “Orpo: Monolithic preference optimization without reference model,” 2024. 8
- [169] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” 2018. 8
- [170] R. Rafailov, Y. Chittepudi, R. Park, H. S. Sikchi, J. Hejna, B. Knox, C. Finn, and S. Niekum, “Scaling laws for reward model overoptimization in direct alignment algorithms,” *ArXiv*, vol. abs/2406.02900, 2024. 9
- [171] H. Wang, S. Hao, H. Dong, S. Zhang, Y. Bao, Z. Yang, and

- Y. Wu, "Offline reinforcement learning for llm multi-step reasoning," 2024. [10](#), [22](#)
- [172] G. Mukobi, P. Chatain, S. Fong, R. Windesheim, G. Kutyniok, K. Bhatia, and S. Alberti, "Superhf: Supervised iterative learning from human feedback," *arXiv preprint arXiv:2310.16763*, 2023. [11](#)
- [173] C. Li, H. Zhou, G. Glavaš, A. Korhonen, and I. Vulić, "On task performance and model calibration with supervised and self-ensembled in-context learning," *arXiv preprint arXiv:2312.13772*, 2023. [11](#)
- [174] C. Wang, Z. Zhao, C. Zhu, K. A. Sankararaman, M. Valko, X. Cao, Z. Chen, M. Khabsa, Y. Chen, H. Ma, and S. Wang, "Preference optimization with multi-sample comparisons," 2024. [11](#)
- [175] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, , and J. Wei, "Challenging big-bench tasks and whether chain-of-thought can solve them," *arXiv preprint arXiv:2210.09261*, 2022. [11](#), [19](#)
- [176] X. Zhang, C. Du, T. Pang, Q. Liu, W. Gao, and M. Lin, "Chain of preference optimization: Improving chain-of-thought reasoning in LLMs," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. [11](#)
- [177] Z. Zhang, A. Zhang, M. Li, and A. Smola, "Automatic chain of thought prompting in large language models," *arXiv preprint arXiv:2210.03493*, 2022. [11](#)
- [178] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, T. L. Scao, A. Raja, *et al.*, "Multitask prompted training enables zero-shot task generalization," *arXiv preprint arXiv:2110.08207*, 2021. [12](#), [22](#)
- [179] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," *arXiv preprint arXiv:2109.01652*, 2021. [12](#)
- [180] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," 2023. [12](#)
- [181] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," March 2023. [12](#)
- [182] M. Conover, M. Hayes, A. Mathur, J. Xie, J. Wan, S. Shah, A. Ghodsi, P. Wendell, M. Zaharia, and R. Xin, "Free dolly: Introducing the world's first truly open instruction-tuned llm," 2023. [12](#)
- [183] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, *et al.*, "Lamda: Language models for dialog applications," *arXiv preprint arXiv:2201.08239*, 2022. [12](#)
- [184] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," in *The Eleventh International Conference on Learning Representations*, 2023. [12](#), [15](#), [20](#)
- [185] L. C. Magister, J. Mallinson, J. Adamek, E. Malmi, and A. Severyn, "Teaching small language models to reason," *arXiv preprint arXiv:2212.08410*, 2022. [12](#), [13](#)
- [186] G. Xu, P. Jin, H. Li, Y. Song, L. Sun, and L. Yuan, "Llava-cot: Let vision language models reason step-by-step," 2024. [12](#)
- [187] O. Thawakar, D. Dissanayake, K. More, R. Thawkar, A. Heakl, N. Ahsan, Y. Li, M. Zumri, J. Lahoud, R. M. Anwer, H. Cholakkal, I. Laptev, M. Shah, F. S. Khan, and S. Khan, "Llamav-o1: Rethinking step-by-step visual reasoning in llms," 2025. [12](#), [19](#)
- [188] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *arXiv preprint arXiv:2305.14314*, 2023. [13](#), [14](#)
- [189] E. Frantar, S. Ashkboos, T. Hoeffler, and D. Alistarh, "GPTQ: Accurate post-training compression for generative pretrained transformers," *arXiv preprint arXiv:2210.17323*, 2022. [13](#)
- [190] E. Frantar and D. Alistarh, "SparseGPT: Massive language models can be accurately pruned in one-shot," *arXiv preprint arXiv:2301.00774*, 2023. [13](#), [14](#)
- [191] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, and B. Bossan, "Peft: State-of-the-art parameter-efficient finetuning methods," 2022. [13](#)
- [192] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm.int8(): 8-bit matrix multiplication for transformers at scale," *arXiv preprint arXiv:2208.07339*, 2022. [13](#)
- [193] Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, and T. Zhao, "Adaptive budget allocation for parameter-efficient fine-tuning," in *The Eleventh International Conference on Learning Representations*, 2023. [13](#), [20](#)
- [194] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," *CoRR*, vol. abs/2110.07602, 2021. [13](#)
- [195] Q. Lhoest, A. Villanova del Moral, Y. Jernite, A. Thakur, P. von Platen, S. Patil, J. Chaumond, M. Drame, J. Plu, L. Tunstall, J. Davison, M. Šaško, G. Chhablani, B. Malik, S. Brandeis, T. Le Scao, V. Sanh, C. Xu, N. Patry, A. McMillan-Major, P. Schmid, S. Gugger, C. Delangue, T. Matussière, L. Debut, S. Bekman, P. Cistac, T. Goehringer, V. Mustar, F. Lagunas, A. Rush, and T. Wolf, "Datasets: A community library for natural language processing," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (Online and Punta Cana, Dominican Republic), pp. 175–184, Association for Computational Linguistics, Nov. 2021. [13](#)
- [196] A. Torralba and Others, "Webdataset: A format for petascale deep learning." Efficient tar-based sharding format for petascale distributed training. [13](#)
- [197] I. Iterative, "Dvc: Data version control." Git-like version control for datasets and machine learning pipelines. [13](#)
- [198] N. Richardson, I. Cook, N. Crane, D. Dunnington, R. François, J. Keane, D. Moldovan-Grünfeld, J. Ooms, J. Wujciak-Jens, and Apache Arrow, *arrow: Integration to 'Apache' 'Arrow'*, 2025. R package version 19.0.0, <https://arrow.apache.org/docs/r/>. [13](#)
- [199] I. Facebook, "Zstandard: High-speed compression algorithm." High-speed compression algorithm for training data storage/-transfer. [13](#)
- [200] C. Team, "Cleanlab: The standard data-centric ai package for machine learning with noisy labels." Automatic detection of label errors and outliers in training datasets. [13](#)
- [201] R. Y. Aminabadi, S. Rajbhandari, M. Zhang, A. A. Awan, C. Li, D. Li, E. Zheng, J. Rasley, S. Smith, O. Ruwase, and Y. He, "Deepspeed inference: Enabling efficient inference of transformer models at unprecedented scale," 2022. [13](#)
- [202] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," 2020. [13](#)
- [203] S. Li, H. Liu, Z. Bian, J. Fang, H. Huang, Y. Liu, B. Wang, and Y. You, "Colossal-ai: A unified deep learning system for large-scale parallel training," 2023. [13](#)
- [204] A. Sergeev and M. D. Balso, "Horovod: fast and easy distributed deep learning in tensorflow," 2018. [13](#)
- [205] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica, "Ray: A distributed framework for emerging ai applications," 2018. [13](#)
- [206] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," 2023. [13](#)
- [207] Y. Zhou and K. Yang, "Exploring tensorrt to improve real-time inference for deep learning," in *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, pp. 2011–2018, 2022. [13](#)
- [208] P. Tillet, H.-T. Kung, and D. Cox, "Triton: an intermediate language and compiler for tiled neural network computations," in *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019. [13](#)
- [209] O. Community, "Onnx: Open neural network exchange." Unified inference engine with hardware-specific optimizations. [13](#)
- [210] I. Corporation, "Openvino: Intel optimization toolkit," 2025. Runtime for Intel CPUs/iGPUs with pruning/quantization support. [13](#)
- [211] M. Dukhan, "The indirect convolution algorithm," 2019. [13](#)
- [212] I. Groq, "Groq: Ai accelerator," 2025. Deterministic low-latency inference via custom tensor streaming processor. [13](#)

- [213] J. Castaño, S. Martínez-Fernández, X. Franch, and J. Bogner, “Analyzing the evolution and maintenance of ml models on hugging face,” 2024. [13](#)
- [214] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” in *NIPS-W*, 2017. [13](#)
- [215] S. Hao, Y. Gu, H. Luo, T. Liu, X. Shao, X. Wang, S. Xie, H. Ma, A. Samavedhi, Q. Gao, *et al.*, “Llm reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models,” *arXiv preprint arXiv:2404.05221*, 2024. [13](#)
- [216] S. Pieri, S. S. Mullappilly, F. S. Khan, R. M. Anwer, S. Khan, T. Baldwin, and H. Cholakkal, “Bimedix: Bilingual medical mixture of experts llm,” *arXiv preprint arXiv:2402.13253*, 2024. [13](#)
- [217] Y. Yang, M. C. S. Uy, and A. Huang, “Finbert: A pretrained language model for financial communications,” *arXiv preprint arXiv:2006.08097*, 2020. [13](#)
- [218] D. Thulke, Y. Gao, P. Pelsler, R. Brune, R. Jalota, F. Fok, M. Ramos, I. van Wyk, A. Nasir, H. Goldstein, *et al.*, “Climategpt: Towards ai synthesizing interdisciplinary research on climate change,” *arXiv preprint arXiv:2401.09646*, 2024. [13](#)
- [219] S. S. Mullappilly, A. Shaker, O. Thawakar, H. Cholakkal, R. M. Anwer, S. Khan, and F. S. Khan, “Arabic mini-climategpt: A climate change and sustainability tailored arabic llm,” *arXiv preprint arXiv:2312.09366*, 2023. [13](#)
- [220] Y. Wang, W. Wang, S. Joty, and S. C. Hoi, “Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation,” *arXiv preprint arXiv:2109.00859*, 2021. [13](#)
- [221] K. Kuckreja, M. S. Danish, M. Naseer, A. Das, S. Khan, and F. S. Khan, “Geochat: Grounded large vision-language model for remote sensing,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27831–27840, 2024. [13](#)
- [222] S. S. Mullappilly, M. I. Kurpath, S. Pieri, S. Y. Alseiri, S. Cholakkal, K. Aldahmani, F. Khan, R. Anwer, S. Khan, T. Baldwin, *et al.*, “Bimedix2: Bio-medical expert lmm for diverse medical modalities,” *arXiv preprint arXiv:2412.07769*, 2024. [13](#)
- [223] M. Maaz, H. Rasheed, S. Khan, and F. S. Khan, “Videochatgpt: Towards detailed video understanding via large vision and language models,” *arXiv preprint arXiv:2306.05424*, 2023. [13](#)
- [224] B. Lin, Y. Ye, B. Zhu, J. Cui, M. Ning, P. Jin, and L. Yuan, “Video-llava: Learning united visual representation by alignment before projection,” *arXiv preprint arXiv:2311.10122*, 2023. [13](#)
- [225] H. Zhang, X. Li, and L. Bing, “Video-llama: An instruction-tuned audio-visual language model for video understanding,” *arXiv preprint arXiv:2306.02858*, 2023. [13](#)
- [226] Y. Han, C. Zhang, X. Chen, X. Yang, Z. Wang, G. Yu, B. Fu, and H. Zhang, “Chartllama: A multimodal llm for chart understanding and generation,” *arXiv preprint arXiv:2311.16483*, 2023. [13](#)
- [227] X. Zhu, J. Li, Y. Liu, C. Ma, and W. Wang, “A survey on model compression for large language models,” *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 1556–1577, 2024. [13](#)
- [228] Z. Wan, X. Wang, C. Liu, S. Alam, Y. Zheng, J. Liu, Z. Qu, S. Yan, Y. Zhu, Q. Zhang, *et al.*, “Efficient large language models: A survey,” *arXiv preprint arXiv:2312.03863*, 2023. [13](#)
- [229] C.-Y. Hsieh, C.-L. Li, C.-K. Yeh, H. Nakhost, Y. Fujii, A. Ratner, R. Krishna, C.-Y. Lee, and T. Pfister, “Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes,” *arXiv preprint arXiv:2305.02301*, 2023. [13](#)
- [230] Y. Gu, L. Dong, F. Wei, and M. Huang, “Minillm: Knowledge distillation of large language models,” *arXiv preprint arXiv:2306.08543*, 2023. [13](#)
- [231] X. L. Li and P. Liang, “Prefix-tuning: Optimizing continuous prompts for generation,” *arXiv preprint arXiv:2101.00190*, 2021. [13](#)
- [232] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, “Parameter-efficient transfer learning for nlp,” 2019. [13](#)
- [233] B. P. Lowerre and B. R. Reddy, “Harpy, a connected speech recognition system,” *The Journal of the Acoustical Society of America*, vol. 59, no. S1, pp. S97–S97, 1976. [14](#)
- [234] A. Graves, “Sequence transduction with recurrent neural networks,” *arXiv preprint arXiv:1211.3711*, 2012. [14](#)
- [235] H. Sun, M. Haider, R. Zhang, H. Yang, J. Qiu, M. Yin, M. Wang, P. Bartlett, and A. Zanette, “Fast best-of-n decoding via speculative rejection,” 2024. [14](#)
- [236] A. Askell, Y. Bai, A. Chen, D. Drain, D. Ganguli, T. Henighan, A. Jones, N. Joseph, B. Mann, N. DasSarma, *et al.*, “A general language assistant as a laboratory for alignment,” *arXiv preprint arXiv:2112.00861*, 2021. [14](#)
- [237] A. Glaese, N. McAleese, M. Trębacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. Chadwick, P. Thacker, *et al.*, “Improving alignment of dialogue agents via targeted human judgements,” *arXiv preprint arXiv:2209.14375*, 2022. [14](#)
- [238] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano, “Learning to summarize with human feedback,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 3008–3021, 2020. [14](#)
- [239] J. Q. Yang, S. Salamatian, Z. Sun, A. T. Suresh, and A. Beirami, “Asymptotics of language model alignment,” *arXiv preprint arXiv:2404.01730*, 2024. [15](#)
- [240] H. Sun, M. Haider, R. Zhang, H. Yang, J. Qiu, M. Yin, M. Wang, P. Bartlett, and A. Zanette, “Fast best-of-n decoding via speculative rejection,” *arXiv preprint arXiv:2410.20290*, 2024. [15](#)
- [241] J. Hilton and L. Gao, “Measuring goodhart’s law,” *OpenAI Research Blog*, 2022. [15](#)
- [242] L. Wang, W. Xu, Y. Lan, Z. Hu, Y. Lan, R. K.-W. Lee, and E.-P. Lim, “Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models,” *arXiv preprint arXiv:2305.04091*, 2023. [15](#)
- [243] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, “Self-instruct: Aligning language models with self-generated instructions,” *arXiv preprint arXiv:2212.10560*, 2022. [15](#)
- [244] X. Chen, R. Aksitov, U. Alon, J. Ren, K. Xiao, P. Yin, S. Prakash, C. Sutton, X. Wang, and D. Zhou, “Universal self-consistency for large language models,” in *ICML 2024 Workshop on In-Context Learning*. [15](#)
- [245] A. Newell, “On the analysis of human problem solving protocols,” 1966. [16](#)
- [246] F. Haji, M. Bethany, M. Tabar, J. Chiang, A. Rios, and P. Najafirad, “Improving llm reasoning with multi-agent tree-of-thought validator agent,” *arXiv preprint arXiv:2409.11527*, 2024. [16](#)
- [247] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk, and T. Hoefler, “Graph of thoughts: Solving elaborate problems with large language models,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, p. 17682–17690, Mar. 2024. [16](#)
- [248] D. Wilson, “Llm tree search,” *arXiv preprint arXiv:2410.19117*, 2024. [17](#)
- [249] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” *arXiv preprint arXiv:1610.02136*, 2016. [17](#)
- [250] G. Portillo Wightman, A. DeLucia, and M. Dredze, “Strength in numbers: Estimating confidence of large language models by prompt agreement,” in *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pp. 326–362, 2023. [17](#)
- [251] J. Qi, H. Tang, and Z. Zhu, “Verifierq: Enhancing llm test time compute with q-learning-based verifiers,” *arXiv preprint arXiv:2410.08048*, 2024. [17](#)
- [252] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegrefe, U. Alon, N. Dziri, S. Prabhunoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark, “Self-refine: Iterative refinement with self-feedback,” 2023. [17](#)
- [253] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, *et al.*, “The rise and potential of large language model based agents: A survey,” *arXiv preprint arXiv:2309.07864*, 2023. [17](#)
- [254] R. Coulom, “Efficient selectivity and backup operators in monte-carlo tree search,” in *International conference on computers and games*, pp. 72–83, Springer, 2006. [18](#)

- [255] J. X. Chen, “The evolution of computing: AlphaGo,” *Computing in Science & Engineering*, vol. 18, no. 4, pp. 4–7, 2016. [18](#)
- [256] L. Kocsis and C. Szepesvári, “Bandit based monte-carlo planning,” in *European conference on machine learning*, pp. 282–293, Springer, 2006. [18](#)
- [257] H. W. Spruille, C. Edwards, M. V. Olarte, U. Sanyal, H. Ji, and S. Choudhury, “Monte carlo thought search: Large language model querying for complex scientific reasoning in catalyst design,” *arXiv preprint arXiv:2310.14420*, 2023. [18](#), [20](#)
- [258] M. DeLorenzo, A. B. Chowdhury, V. Gohil, S. Thakur, R. Karri, S. Garg, and J. Rajendran, “Make every move count: Llm-based high-quality rtl code generation using mcts,” *arXiv preprint arXiv:2402.03289*, 2024. [18](#)
- [259] S. Park, X. Liu, Y. Gong, and E. Choi, “Ensembling large language models with process reward-guided tree search for better complex reasoning,” *arXiv preprint arXiv:2412.15797*, 2024. [18](#)
- [260] M. Shen, G. Zeng, Z. Qi, Z.-W. Hong, Z. Chen, W. Lu, G. Wornell, S. Das, D. Cox, and C. Gan, “Satori: Reinforcement learning with chain-of-action-thought enhances llm reasoning via autoregressive search,” *arXiv preprint arXiv:2502.02508*, 2025. [18](#)
- [261] S. R. Motwani, C. Smith, R. J. Das, R. Rafailov, I. Laptev, P. H. S. Torr, F. Pizzati, R. Clark, and C. S. de Witt, “Malt: Improving reasoning with multi-agent llm training,” 2025. [18](#)
- [262] U. Anwar, A. Saparov, J. Rando, D. Paleka, M. Turpin, P. Hase, E. S. Lubana, E. Jenner, S. Casper, O. Sourbut, *et al.*, “Foundational challenges in assuring alignment and safety of large language models,” *arXiv preprint arXiv:2404.09932*, 2024. [18](#)
- [263] W. Zhang, P. H. Torr, M. Elhoseiny, and A. Bibi, “Bi-factorial preference optimization: Balancing safety-helpfulness in language models,” *arXiv preprint arXiv:2408.15313*, 2024. [18](#)
- [264] C. Li, W. Wu, H. Zhang, Y. Xia, S. Mao, L. Dong, I. Vulić, and F. Wei, “Imagine while reasoning in space: Multimodal visualization-of-thought,” *arXiv preprint arXiv:2501.07542*, 2025. [19](#)
- [265] F. Nowak, A. Svete, A. Butoi, and R. Cotterell, “On the representational capacity of neural language models with chain-of-thought reasoning,” *arXiv preprint arXiv:2406.14197*, 2024. [19](#)
- [266] Z. Li, H. Liu, D. Zhou, and T. Ma, “Chain of thought empowers transformers to solve inherently serial problems,” *arXiv preprint arXiv:2402.12875*, vol. 1, 2024. [19](#)
- [267] W. Merrill and A. Sabharwal, “The expressive power of transformers with chain of thought,” *arXiv preprint arXiv:2310.07923*, 2023. [19](#)
- [268] C. V. Snell, J. Lee, K. Xu, and A. Kumar, “Scaling test-time compute optimally can be more effective than scaling llm parameters,” in *The Thirteenth International Conference on Learning Representations*. [19](#)
- [269] Saxton *et al.*, “Analysing mathematical reasoning abilities of neural models,” *arXiv:1904.01557*, 2019. [19](#)
- [270] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, “Training verifiers to solve math word problems,” *arXiv preprint arXiv:2110.14168*, 2021. [19](#)
- [271] L. Yu, W. Jiang, H. Shi, J. Yu, Z. Liu, Y. Zhang, J. T. Kwok, Z. Li, A. Weller, and W. Liu, “Metamath: Bootstrap your own mathematical questions for large language models,” *arXiv preprint arXiv:2309.12284*, 2023. [19](#)
- [272] Z. Xie, S. Thiem, J. Martin, E. Wainwright, S. Marmorstein, and P. Jansen, “WorldTree v2: A corpus of science-domain structured explanations and inference patterns supporting multi-hop inference,” in *Proceedings of the Twelfth Language Resources and Evaluation Conference* (N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odiijk, and S. Piperidis, eds.), (Marseille, France), pp. 5456–5473, European Language Resources Association, May 2020. [17](#), [19](#)
- [273] F. Liu, E. Bugliarello, E. M. Ponti, S. Reddy, N. Collier, and D. Elliott, “Visually grounded reasoning across languages and cultures,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, (Online and Punta Cana, Dominican Republic), pp. 10467–10485, Association for Computational Linguistics, Nov. 2021. [19](#)
- [274] X. Yue, Y. Ni, K. Zhang, T. Zheng, R. Liu, G. Zhang, S. Stevens, D. Jiang, W. Ren, Y. Sun, C. Wei, B. Yu, R. Yuan, R. Sun, M. Yin, B. Zheng, Z. Yang, Y. Liu, W. Huang, H. Sun, Y. Su, and W. Chen, “Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi,” in *Proceedings of CVPR*, 2024. [19](#)
- [275] S. Lin, J. Hilton, and O. Evans, “Truthfulqa: Measuring how models mimic human falsehoods,” *arXiv preprint arXiv:2109.07958*, 2021. [19](#)
- [276] X. Yue *et al.*, “Mammoth: Building math generalist models through hybrid instruction tuning,” *arXiv preprint arXiv:2309.05653*, 2023. [19](#)
- [277] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, “Measuring massive multitask language understanding,” *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. [19](#), [20](#)
- [278] D. Hendrycks, C. Burns, S. Basart, A. Critch, J. Li, D. Song, and J. Steinhardt, “Aligning ai with shared human values,” *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. [19](#), [20](#)
- [279] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner, “DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs,” in *Proc. of NAACL*, 2019. [19](#)
- [280] Z. Wang, Y. Dong, J. Zeng, V. Adams, M. N. Sreedhar, D. Egert, O. Delalleau, J. P. Scowcroft, N. Kant, A. Swope, *et al.*, “Helpsteer: Multi-attribute helpfulness dataset for steerlm,” *arXiv preprint arXiv:2311.09528*, 2023. [19](#)
- [281] G. Cui, L. Yuan, N. Ding, G. Yao, W. Zhu, Y. Ni, G. Xie, Z. Liu, and M. Sun, “Ultrafeedback: Boosting language models with high-quality feedback,” 2023. [19](#)
- [282] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, “D4rl: Datasets for deep data-driven reinforcement learning,” 2020. [19](#), [20](#)
- [283] T. Schmied, M. Hofmarcher, F. Paischer, R. Pascanu, and S. Hochreiter, “Learning to modulate pre-trained models in rl,” *Advances in Neural Information Processing Systems*, vol. 36, 2024. [19](#), [20](#)
- [284] W. H. Guss, B. Houghton, N. Topin, P. Wang, C. Codel, M. Veloso, and R. Salakhutdinov, “Minerl: A large-scale dataset of minecraft demonstrations,” 2019. [19](#), [20](#)
- [285] T. Nguyen, C. V. Nguyen, V. D. Lai, H. Man, N. T. Ngo, F. Dernoncourt, R. A. Rossi, and T. H. Nguyen, “CulturaX: A cleaned, enormous, and multilingual dataset for large language models in 167 languages,” in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)* (N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, and N. Xue, eds.), (Torino, Italia), pp. 4226–4237, ELRA and ICCL, May 2024. [19](#), [20](#)
- [286] X. Yue, Y. Song, A. Asai, S. Kim, J. de Dieu Nyandwi, S. Khanuja, A. Kantharuban, L. Sutawika, S. Ramamoorthy, and G. Neubig, “Pangea: A fully open multilingual multimodal llm for 39 languages,” *arXiv preprint arXiv:2410.16153*, 2024. [19](#), [20](#)
- [287] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” *CoRR*, vol. abs/1810.04805, 2018. [19](#), [20](#)
- [288] Y. Liang, N. Duan, Y. Gong, N. Wu, F. Guo, W. Qi, M. Gong, L. Shou, D. Jiang, G. Cao, X. Fan, R. Zhang, R. Agrawal, E. Cui, S. Wei, T. Bharti, Y. Qiao, J.-H. Chen, W. Wu, S. Liu, F. Yang, D. Campos, R. Majumder, and M. Zhou, “Xglue: A new benchmark dataset for cross-lingual pre-training, understanding and generation,” *arXiv*, vol. abs/2004.01401, 2020. [19](#)
- [289] G. Son, D. Yoon, J. Suk, J. Aula-Blasco, M. Aslan, V. T. Kim, S. B. Islam, J. Prats-Cristià, L. Tormo-Bañuelos, and S. Kim, “Mm-eval: A multilingual meta-evaluation benchmark for llms-as-a-judge and reward models,” 2024. [19](#), [20](#)
- [290] S. *et al.*, “Beyond the imitation game: Quantifying and extrapolating the capabilities of language models,” 2022. [19](#), [20](#)
- [291] L. Zhang, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, *et al.*, “Judging llm-as-a-judge with mt-bench and chatbot arena,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 46595–46623, 2023. [19](#), [20](#)
- [292] E. Dinan, V. Logacheva, V. Malykh, A. H. Miller, K. Shuster, J. Urbanek, D. Kiela, A. Szlam, I. Serban, R. Lowe, S. Prabhunoye, A. W. Black, A. I. Rudnicky, J. Williams, J. Pineau, M. S. Burtsev, and J. Weston, “The second conversational

- intelligence challenge (convai2),” *CoRR*, vol. abs/1902.00098, 2019. [19](#), [20](#)
- [293] M. Eric, R. Goel, S. Paul, A. Sethi, S. Agarwal, S. Gao, and D. Hakkani-Tur, “MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines,” in *Proceedings of the 12th Language Resources and Evaluation Conference*, (Marseille, France), pp. 422–428, European Language Resources Association, May 2020. [19](#), [20](#)
- [294] X. Li and D. Roth, “Learning question classifiers,” in *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. [19](#), [20](#)
- [295] E. Hovy, L. Gerber, U. Hermjakob, C.-Y. Lin, and D. Ravichandran, “Toward semantics-based answer pinpointing,” in *Proceedings of the First International Conference on Human Language Technology Research*, 2001. [19](#), [20](#)
- [296] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, “BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models,” in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [19](#), [20](#)
- [297] C. Chhun, F. M. Suchanek, and C. Clavel, “Do language models enjoy their own stories? Prompting large language models for automatic story evaluation,” *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 1122–1142, 2024. [19](#)
- [298] H. Chen, D. M. Vo, H. Takamura, Y. Miyao, and H. Nakayama, “Storyer: Automatic story evaluation via ranking, rating and reasoning,” 2022. [19](#)
- [299] J. Ji, M. Liu, J. Dai, X. Pan, C. Zhang, C. Bian, B. Chen, R. Sun, Y. Wang, and Y. Yang, “Beavertails: Towards improved safety alignment of llm via a human-preference dataset,” *Advances in Neural Information Processing Systems*, vol. 36, 2024. [19](#), [20](#)
- [300] G. Xu, J. Liu, M. Yan, H. Xu, J. Si, Z. Zhou, P. Yi, X. Gao, J. Sang, R. Zhang, *et al.*, “Cvalues: Measuring the values of chinese large language models from safety to responsibility,” *arXiv preprint arXiv:2307.09705*, 2023. [19](#)
- [301] S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi, “Cross-task generalization via natural language crowdsourcing instructions,” in *ACL*, 2022. [19](#)
- [302] Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Arunkumar, A. Ashok, A. S. Dhanasekaran, A. Naik, D. Stap, *et al.*, “Super-naturalinstructions: generalization via declarative instructions on 1600+ tasks,” in *EMNLP*, 2022. [19](#)
- [303] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, “Training language models to follow instructions with human feedback,” in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), 2022. [20](#)
- [304] W. Saunders, C. Yeh, J. Wu, S. Bills, L. Ouyang, J. Ward, and J. Leike, “Self-critiquing models for assisting human evaluators,” *arXiv preprint arXiv:2206.05802*, 2022. [20](#)
- [305] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, “Scaling laws for neural language models,” *arXiv preprint arXiv:2001.08361*, 2020. [20](#)
- [306] S. Roy and D. Roth, “Solving general arithmetic word problems,” 2016. [20](#)
- [307] Y. Jinai, T. Morimura, K. Ariu, and K. Abe, “Regularized best-of-n sampling to mitigate reward hacking for language model alignment,” *arXiv preprint arXiv:2404.01054*, 2024. [20](#)
- [308] L. Chen, C. Zhu, D. Soselia, J. Chen, T. Zhou, T. Goldstein, H. Huang, M. Shoenybi, and B. Catanzaro, “Odin: Disentangled reward mitigates hacking in rlhf,” *ArXiv*, vol. abs/2402.07319, 2024. [20](#)
- [309] T. Liu, W. Xiong, J. Ren, L. Chen, J. Wu, R. Joshi, Y. Gao, J. Shen, Z. Qin, T. Yu, D. Sohn, A. Makarova, J. Liu, Y. Liu, B. Piot, A. Ittycheriah, A. Kumar, and M. Saleh, “Rrm: Robust reward model training mitigates reward hacking,” *ArXiv*, vol. abs/2409.13156, 2024. [20](#)
- [310] C. Wang, Z. Zhao, Y. Jiang, Z. Chen, C. Zhu, Y. Chen, J. Liu, L. Zhang, X. Fan, H. Ma, and S.-Y. Wang, “Beyond reward hacking: Causal rewards for large language model alignment,” 2025. [20](#)
- [311] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, “A survey of monte carlo tree search methods,” *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012. [20](#)
- [312] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegrefe, U. Alon, N. Dziri, S. Prabhume, Y. Yang, *et al.*, “Self-refine: Iterative refinement with self-feedback,” *Advances in Neural Information Processing Systems*, vol. 36, 2024. [20](#)
- [313] Y. Fu, H. Peng, A. Sabharwal, P. Clark, and T. Khot, “Complexity-based prompting for multi-step reasoning,” in *The Eleventh International Conference on Learning Representations*, 2022. [20](#)
- [314] B. Johnson, “Metacognition for artificial intelligence system safety—an approach to safe and desired behavior,” *Safety Science*, vol. 151, p. 105743, 2022. [20](#), [21](#)
- [315] OpenAI, “Early access for safety testing,” 2024. [20](#)
- [316] Y. Yan, X. Lou, J. Li, Y. Zhang, J. Xie, C. Yu, Y. Wang, D. Yan, and Y. Shen, “Reward-robust rlhf in llms,” *ArXiv*, vol. abs/2409.15360, 2024. [20](#)
- [317] W. Yu, Z. Sun, J. Xu, Z. Dong, X. Chen, H. Xu, and J.-R. Wen, “Explainable legal case matching via inverse optimal transport-based rationale extraction,” in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 657–668, 2022. [20](#)
- [318] A. Amini, S. Gabriel, P. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi, “Mathqa: Towards interpretable math word problem solving with operation-based formalisms,” 2019. [20](#)
- [319] L. Chen, O. Sinavski, J. Hünermann, A. Karnsund, A. J. Willmott, D. Birch, D. Maund, and J. Shotton, “Driving with llms: Fusing object-level vector modality for explainable autonomous driving,” *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14093–14100, 2023. [20](#)
- [320] R. Poulain, H. Fayyaz, and R. Beheshti, “Bias patterns in the application of llms for clinical decision support: A comprehensive study,” *arXiv preprint arXiv:2404.15149*, 2024. [20](#)
- [321] Z. Fan, R. Chen, R. Xu, and Z. Liu, “Biasalert: A plug-and-play tool for social bias detection in llms,” *arXiv preprint arXiv:2407.10241*, 2024. [20](#)
- [322] M. Li, T. Shi, C. Ziem, M.-Y. Kan, N. F. Chen, Z. Liu, and D. Yang, “Coannotating: Uncertainty-guided work allocation between human and large language models for data annotation,” *arXiv preprint arXiv:2310.15638*, 2023. [20](#)
- [323] A. Elangovan, J. Ko, L. Xu, M. Elyasi, L. Liu, S. Bodapati, and D. Roth, “Beyond correlation: The impact of human uncertainty in measuring the effectiveness of automatic evaluation and llm-as-a-judge,” *arXiv preprint arXiv:2410.03775*, 2024. [20](#)
- [324] Y. R. Dong, T. Hu, and N. Collier, “Can llm be a personalized judge?,” *arXiv preprint arXiv:2406.11657*, 2024. [20](#)
- [325] D. Wang, K. Yang, H. Zhu, X. Yang, A. Cohen, L. Li, and Y. Tian, “Learning personalized story evaluation,” *arXiv preprint arXiv:2310.03304*, 2023. [20](#)
- [326] H. Du, S. Liu, L. Zheng, Y. Cao, A. Nakamura, and L. Chen, “Privacy in fine-tuning large language models: Attacks, defenses, and future directions,” 2024. [20](#), [21](#), [22](#)
- [327] L. Yuan, W. Li, H. Chen, G. Cui, N. Ding, K. Zhang, B. Zhou, Z. Liu, and H. Peng, “Free process rewards without process labels,” *arXiv preprint arXiv:2412.01981*, 2024. [20](#)
- [328] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, “Eureka: Human-level reward design via coding large language models,” *ArXiv*, vol. abs/2310.12931, 2023. [20](#)
- [329] A. Havrilla, S. C. Rapparth, C. Nalmpantis, J. Dwivedi-Yu, M. Zhuravinskyi, E. Hambro, and R. Railneau, “Glore: When, where, and how to improve llm reasoning via global and local refinements,” *ArXiv*, vol. abs/2402.10963, 2024. [20](#)
- [330] M. Fawi, “Curlora: Stable llm continual fine-tuning and catastrophic forgetting mitigation,” 2024. [20](#)
- [331] C. Fu, P. Chen, Y. Shen, Y. Qin, M. Zhang, X. Lin, Z. Qiu, W. Lin, J. Yang, X. Zheng, K. Li, X. Sun, and R. Ji, “Mme: A comprehensive evaluation benchmark for multimodal large language models,” *ArXiv*, vol. abs/2306.13394, 2023. [20](#)
- [332] Y. Wang, Z. Yu, Z. Zeng, L. Yang, C. Wang, H. Chen, C. Jiang, R. Xie, J. Wang, X. Xie, W. Ye, S.-B. Zhang, and Y. Zhang, “Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization,” *ArXiv*, vol. abs/2306.05087, 2023. [20](#)

- [333] Y. Sun, Z. Li, Y. Li, and B. Ding, “Improving lora in privacy-preserving federated learning,” *ArXiv*, vol. abs/2403.12313, 2024. [20](#), [21](#)
- [334] Y. He, Y. Kang, L. Fan, and Q. Yang, “Fedeval-llm: Federated evaluation of large language models on downstream tasks with collective wisdom,” *arXiv preprint arXiv:2404.12273*, 2024. [20](#)
- [335] J. Park, S. Jwa, M. Ren, D. Kim, and S. Choi, “Offsetbias: Leveraging debiased data for tuning evaluators,” *arXiv preprint arXiv:2407.06551*, 2024. [20](#)
- [336] P. Lu, L. Qiu, K.-W. Chang, Y. N. Wu, S.-C. Zhu, T. Rajpurohit, P. Clark, and A. Kalyan, “Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning,” 2023. [20](#)
- [337] L. Zhang, A. Hosseini, H. Bansal, M. Kazemi, A. Kumar, and R. Agarwal, “Generative verifiers: Reward modeling as next-token prediction,” in *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS’24*, 2024. [20](#)
- [338] S. Yang and D. Song, “FPC: Fine-tuning with prompt curriculum for relation extraction,” in *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (Y. He, H. Ji, S. Li, Y. Liu, and C.-H. Chang, eds.), (Online only), pp. 1065–1077, Association for Computational Linguistics, Nov. 2022. [20](#)
- [339] Y. Yu, W. Ping, Z. Liu, B. Wang, J. You, C. Zhang, M. Shoenybi, and B. Catanzaro, “Rankrag: Unifying context ranking with retrieval-augmented generation in llms,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 121156–121184, 2025. [20](#)
- [340] X. V. Lin, X. Chen, M. Chen, W. Shi, M. Lomeli, R. James, P. Rodriguez, J. Kahn, G. Szilvasy, M. Lewis, *et al.*, “Ra-dit: Retrieval-augmented dual instruction tuning,” in *The Twelfth International Conference on Learning Representations*, 2023. [20](#)
- [341] T. Zhang, S. G. Patil, N. Jain, S. Shen, M. Zaharia, I. Stoica, and J. E. Gonzalez, “Raft: Adapting language model to domain specific rag,” in *First Conference on Language Modeling*, 2024. [20](#)
- [342] T. Huang, S. Hu, F. Ilhan, S. F. Tekin, and L. Liu, “Harmful fine-tuning attacks and defenses for large language models: A survey,” *arXiv preprint arXiv:2409.18169*, 2024. [21](#)
- [343] S. R. Bowman, J. Hyun, E. Perez, E. Chen, C. Pettit, S. Heiner, K. Lukošiuūtė, A. Askell, A. Jones, A. Chen, *et al.*, “Measuring progress on scalable oversight for large language models,” *arXiv preprint arXiv:2211.03540*, 2022. [21](#)
- [344] N. Hollmann, S. Müller, and F. Hutter, “Llms for semi-automated data science: Introducing caafe for context-aware automated feature engineering,” *CoRR*, 2023. [21](#)
- [345] S. R. Motwani, C. Smith, R. J. Das, M. Rybchuk, P. H. Torr, I. Laptsev, F. Pizzati, R. Clark, and C. S. de Witt, “Malt: Improving reasoning with multi-agent llm training,” *arXiv preprint arXiv:2412.01928*, 2024. [21](#)
- [346] A. Estornell, J.-F. Ton, Y. Yao, and Y. Liu, “Acc-debate: An actor-critic approach to multi-agent debate,” *arXiv preprint arXiv:2411.00053*, 2024. [21](#)
- [347] L. Luo, Y. Liu, R. Liu, S. Phatale, H. Lara, Y. Li, L. Shu, Y. Zhu, L. Meng, J. Sun, *et al.*, “Improve mathematical reasoning in language models by automated process supervision,” *arXiv preprint arXiv:2406.06592*, 2024. [22](#)
- [348] W. Shen, X. Zhang, Y. Yao, R. Zheng, H. Guo, and Y. Liu, “Improving reinforcement learning from human feedback using contrastive rewards,” *arXiv preprint arXiv:2403.07708*, 2024. [22](#)
- [349] M. Ma, P. D’Oro, Y. Bengio, and P.-L. Bacon, “Long-term credit assignment via model-based temporal shortcuts,” in *Deep RL Workshop NeurIPS 2021*, 2021. [22](#)
- [350] E. Pignatelli, J. Ferret, M. Geist, T. Mesnard, H. van Hasselt, O. Pietquin, and L. Toni, “A survey of temporal credit assignment in deep reinforcement learning,” *arXiv preprint arXiv:2312.01072*, 2023. [22](#)
- [351] H. Zhang and Y. Guo, “Generalization of reinforcement learning with policy-aware adversarial data augmentation,” 2021. [22](#)
- [352] A. Ahmadian, C. Cremer, M. Gallé, M. Fadaee, J. Kreutzer, O. Pietquin, A. Üstün, and S. Hooker, “Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms,” *arXiv preprint arXiv:2402.14740*, 2024. [22](#)
- [353] S. Lee, G. Lee, J. W. Kim, J. Shin, and M.-K. Lee, “Hetal: Efficient privacy-preserving transfer learning with homomorphic encryption,” 2024. [22](#)
- [354] Y. Wei, J. Jia, Y. Wu, C. Hu, C. Dong, Z. Liu, X. Chen, Y. Peng, and S. Wang, “Distributed differential privacy via shuffling versus aggregation: A curious study,” *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 2501–2516, 2024. [22](#)
- [355] W. Zhang, K. Tang, H. Wu, M. Wang, Y. Shen, G. Hou, Z. Tan, P. Li, Y. Zhuang, and W. Lu, “Agent-pro: Learning to evolve via policy-level reflection and optimization,” *arXiv preprint arXiv:2402.17574*, 2024. [22](#)
- [356] C. Ma, J. Zhang, Z. Zhu, C. Yang, Y. Yang, Y. Jin, Z. Lan, L. Kong, and J. He, “Agentboard: An analytical evaluation board of multi-turn llm agents,” *arXiv preprint arXiv:2401.13178*, 2024. [22](#)
- [357] J. Zhang, J. Xiang, Z. Yu, F. Teng, X. Chen, J. Chen, M. Zhuge, X. Cheng, S. Hong, J. Wang, *et al.*, “Aflow: Automating agentic workflow generation,” *arXiv preprint arXiv:2410.10762*, 2024. [22](#)
- [358] H. D. Le, X. Xia, and Z. Chen, “Multi-agent causal discovery using large language models,” *arXiv preprint arXiv:2407.15073*, 2024. [22](#)
- [359] D. M. Owens, R. A. Rossi, S. Kim, T. Yu, F. Dernoncourt, X. Chen, R. Zhang, J. Gu, H. Deilamsalehy, and N. Lipka, “A multi-llm debiasing framework,” *arXiv preprint arXiv:2409.13884*, 2024. [22](#)
- [360] H. Zou, Q. Zhao, L. Bariah, Y. Tian, M. Bennis, S. Lasaulce, M. Debbah, and F. Bader, “Genainet: Enabling wireless collective intelligence via knowledge transfer and reasoning,” *ArXiv*, vol. abs/2402.16631, 2024. [22](#)
- [361] R. Lee, O. J. Mengshoel, A. Saksena, R. Gardner, D. Genin, J. Silbermann, M. Owen, and M. J. Kochenderfer, “Adaptive stress testing: Finding likely failure events with reinforcement learning,” 2020. [22](#)
- [362] A. G. Baydin, B. A. Pearlmutter, D. Syme, F. Wood, and P. Torr, “Gradients without backpropagation,” 2022. [22](#)
- [363] Y. Liu, C. Cai, X. Zhang, X. Yuan, and C. Wang, “Arondight: Red teaming large vision language models with auto-generated multi-modal jailbreak prompts,” in *ACM Multimedia*, 2024. [22](#)
- [364] D. Kim, K. Lee, J. Shin, and J. Kim, “Aligning large language models with self-generated preference data,” *arXiv preprint arXiv:2406.04412*, 2024. [22](#)
- [365] S. Ebrahimi, S. Ö. Arik, T. Nama, and T. Pfister, “Crome: Cross-modal adapters for efficient multimodal llm,” *ArXiv*, vol. abs/2408.06610, 2024. [22](#), [22](#)
- [366] H. Xia, Y. Li, C. T. Leong, W. Wang, and W. Li, “Tokenskip: Controllable chain-of-thought compression in llms,” 2025. [22](#)
- [367] Z. Ma, W. Wu, Z. Zheng, Y. Guo, Q. Chen, S. Zhang, and X. Chen, “Leveraging speech ptm, text llm, and emotional tts for speech emotion recognition,” *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 11146–11150, 2023. [22](#)
- [368] Z. Xi, W. Chen, B. Hong, S. Jin, R. Zheng, W. He, Y. Ding, S. Liu, X. Guo, J. Wang, H. Guo, W. Shen, X. Fan, Y. Zhou, S. Dou, X. Wang, X. Zhang, P. Sun, T. Gui, Q. Zhang, and X. Huang, “Training large language models for reasoning through reverse curriculum reinforcement learning,” *ArXiv*, vol. abs/2402.05808, 2024. [22](#)
- [369] O. Y. Lee, A. Xie, K. Fang, K. Pertsch, and C. Finn, “Affordance-guided reinforcement learning via visual prompting,” *ArXiv*, vol. abs/2407.10341, 2024. [22](#)
- [370] H. Xu, Z. Zhu, D. Ma, S. Zhang, S. Fan, L. Chen, and K. Yu, “Rejection improves reliability: Training llms to refuse unknown questions using rl from knowledge feedback,” *ArXiv*, vol. abs/2403.18349, 2024. [22](#)
- [371] X. Chen, J. Xu, T. Liang, Z. He, J. Pang, D. Yu, L. Song, Q. Liu, M. Zhou, Z. Zhang, R. Wang, Z. Tu, H. Mi, and D. Yu, “Do not think that much for 2+3=? on the overthinking of o1-like llms,” *ArXiv*, vol. abs/2412.21187, 2024. [22](#)
- [372] M. Kemmerling, D. Lütticke, and R. H. Schmitt, “Beyond games: a systematic review of neural monte carlo tree search applications,” *Applied Intelligence*, vol. 54, no. 1, pp. 1020–1046, 2024. [22](#)
- [373] Y. Li, H. Wen, W. Wang, X. Li, Y. Yuan, G. Liu, J. Liu, W. Xu, X. Wang, Y. Sun, R. Kong, Y. Wang, H. Geng, J. Luan, X. Jin, Z.-L. Ye, G. Xiong, F. Zhang, X. Li, M. Xu, Z. Li, P. Li,

- Y. Liu, Y. Zhang, and Y. Liu, "Personal llm agents: Insights and survey about the capability, efficiency and security," *ArXiv*, vol. abs/2401.05459, 2024. [22](#)
- [374] H. Li, L. Ding, M. Fang, and D. Tao, "Revisiting catastrophic forgetting in large language model tuning," 2024. [22](#)
- [375] N. Alzahrani, H. A. Alyahya, Y. Alnumay, S. Alrashed, S. Alsubaie, Y. Almushaykeh, F. Mirza, N. Alotaibi, N. Altwaresh, A. Alowisheq, M. S. Bari, and H. Khan, "When benchmarks are targets: Revealing the sensitivity of large language model leaderboards," 2024. [23](#)