On The Design Choices of Next Level LLMs

Yijun Tian[1], Xingjian Diao[2], Ming Cheng[2], Chunhui Zhang[2], Jiang Gui[2], Soroush Vosoughi[2], Xiangliang Zhang[1], Nitesh V. Chawla[1], and Shichao Pei[3]

[1]University of Notre Dame
[2]Dartmouth College
[3]University of Massachusetts Boston

June 19, 2025

Abstract

As large language models (LLMs) are evolving at a rapid pace, the design space for building high-performance and efficient models has expanded significantly. This position paper argues that the next level of LLMs will combine decoder-only architectures with modular enhancements in perception and specialization, supported by adaptive post-training and hardware-aware optimization strategies. We analyze 31 influential LLMs developed by 12 leading institutions, examining their design choices across three critical dimensions: model architecture, post-training, and optimization. By asking and answering 9 fundamental research questions surrounding these design choices, we identify key trends and propose forward-looking directions to guide the design of next-generation LLMs that are more capable, efficient, and adaptable to diverse real-world needs. To support the community, we maintain a GitHub repository that will be continuously updated at https://github.com/meettyj/LLMs-Design-Choices.

On The Design Choices of Next Level LLMs

Yijun Tian[1], Xingjian Diao[2], Ming Cheng[2], Chunhui Zhang[2], Jiang Gui[2], Soroush Vosoughi[2], Xiangliang Zhang[1], Nitesh V. Chawla[1], Shichao Pei[3]
[1]University of Notre Dame, [2]Dartmouth College,
[3]University of Massachusetts Boston

Abstract

As large language models (LLMs) are evolving at a rapid pace, the design space for building high-performance and efficient models has expanded significantly. This position paper argues that the next level of LLMs will combine decoder-only architectures with modular enhancements in perception and specialization, supported by adaptive post-training and hardware-aware optimization strategies. We analyze 31 influential LLMs developed by 12 leading institutions, examining their design choices across three critical dimensions: model architecture, post-training, and optimization. By asking and answering 9 fundamental research questions surrounding these design choices, we identify key trends and propose forward-looking directions to guide the design of next-generation LLMs that are more capable, efficient, and adaptable to diverse real-world needs. To support the community, we maintain a GitHub repository that will be continuously updated at `https://github.com/meettyj/LLMs-Design-Choices`.

1 Introduction

Large Language Models (LLMs) have rapidly evolved across academia, industry, and public life [83, 84, 15, 118]. As models proliferate, the design space for building high-performance, generalizable, and efficient LLMs has expanded dramatically. In this paper, we analyze **31** of the most influential LLMs to date from 12 institutes[1], examining their design choices on model architecture, post-training, and optimization. From this analysis, we identify key patterns, tradeoffs, and emerging paradigms that we believe will shape the next generation of LLMs.

On the design choices of **Model Architecture**, we focus on answering two research questions: ① What makes decoder-only architectures dominant? and ② When to use dense or MoE architectures? We conclude that decoder-only architectures dominate for their task unification, scalability, inference efficiency, and alignment to pre-training. In addition, dense and MoE model architectures reflect different strengths, where dense models support stable training and easy deployment, and MoE enables efficient scaling and expert specialization. Our position is that **next level of LLMs will retain decoder-only cores for general capabilities while incorporating encoder-style components for perception, and various expert modules for scalability and specialization.**

On the design choices of **Post-training**, we focus on answering three research questions: ① How to select task and data for post-training? ② How to choose post-training strategy? and ③ How to obtain reward for reinforcement learning? We conclude that the modern LLMs are trained on varied

[1]The included 31 models ranked by institute name: **AI2** (OLMo 2 [85], Tülu 3 [61]), **Alibaba** (Qwen2 [136], Qwen2.5 [137], Qwen2.5-1M [138], QwQ-32B [92], Qwen3 [135]), **Anthropic** (Aya 23 [9], Claude 3 [7], Claude 4 [8]), **Apple** (OpenELM [81], DCLM [65]), **DeepSeek** (DeepSeek-V3 [73], DeepSeek-R1 [36]), **Google** (Gemini 1.0 [106], Gemini 1.5 [107], Gemini 2.0 [32], Gemini 2.5 [33], Gemma 1 [108], Gemma 2 [29], Gemma 3 [28]), **Meta** (Llama 3 [35], Llama 4 [82]), **Microsoft** (Phi-3 [1], Phi-4 [2], Phi-4-Mini [3]), **Mistral AI** (Mistral 7B [52], Mixtral 8x7B [53]), **Moonshot AI** (Kimi K1.5 [109]), **OpenAI** (GPT-4 [4]), **Zhipu AI** (ChatGLM series [30]).

data types and refined through post-training that involves diverse strategies, including reinforcement learning and increasing use of synthetic data. Reward signals used in this process can come from human preferences, rule-based verification, or AI-generated feedback. Our position is that **next level of LLMs will emphasize data-centric learning, adaptive post-training pipelines, and dynamic reward modeling that integrates human and AI feedback to tailor models to specific tasks and different user needs.**

On the design choices of **Optimization**, we focus on answering four research questions: ① How to design prompt template? ② How to minimize memory usage? ③ How to obtain a smaller model? and ④ How to accelerate training and inference? We conclude that effective and efficient LLM development relies on well-crafted prompt design, memory-efficient techniques, advanced capability distillation, parallelism strategies, and hardware-aware optimizations. Our position is that **next level of LLM will unify prompt design, optimize memory and compute through hardware-algorithm co-design, and evolve into a layered ecosystem of super-teacher and specialized student models for greater efficiency, generalization, and customization.**

2 Design Choices on Model Architecture

2.1 What Makes Decoder-only Architectures Dominant?

> **Takeaway:** Decoder-only models dominate LLM design because of their ability that unifying tasks through next-token prediction, simplifying architecture for scalability, enabling efficient streaming inference, and aligning naturally with raw-text pre-training.
> **Position:** The next leap may come from modular extensions on top of decoder-only models, such as multimodal encoders for perception, simulators or search interfaces for interaction, and encoder-style modules for memory and personalization.

Decoder-only architectures have become the dominant and default choice for building LLMs nowadays, effectively replacing encoder-decoder and encoder-only designs. This transition is driven by four core advantages: ① Decoder-only architectures unify diverse NLP tasks such as summarization and translation under a single next-token prediction training objective. With proper instruction tuning, LLMs can generalize across tasks without training task-specific heads, as demonstrated in **Llama 3**, **Claude 3**, and **QwQ-32B**. ② Decoder-only architectures are simple and highly scalable. Removing encoders and cross-attentions simplifies implementation and eliminates coordination overhead, enabling efficient scaling to hundreds of billions of parameters in models like **Llama 4**, **Command A**, and **DeepSeek-R1&V3**. ③ Decoder-only architectures enable efficient inference via streaming-style generation, producing outputs token by token without re-encoding input. This makes them well-suited for latency-sensitive deployment tasks. Several models leverage this advantage, including **Phi-4**, **Kimi K1.5**, and **Qwen2.5**. ④ The autoregressive nature of decoder-only architectures aligns naturally with pre-training on vast amounts of unstructured web text, eliminating the need for obtaining input-output pairs. This compatibility has enabled highly scalable pre-training approaches in **Gemini 1.5**, **Gemma 3**, and **OLMo 2**. In contrast, encoder-decoder models [94, 134, 63, 105] rely on input-output pairs and increasingly limited to alignment-heavy tasks, many of which can be reformulated for decoder-only models via prompting. In addition, encoder-only models [23, 75, 62, 40] are not inherently suited for open-ended generation and are used mainly for discriminative tasks such as classification, retrieval, and reranking. These factors explain the dominance of decoder-only architectures in modern LLMs, and increasingly drive the development of modular components to enhance grounding, reasoning, and personalization [152, 78, 150].

2.2 When to Use Dense or MoE Architectures?

> **Takeaway:** Dense models offer strengths such as training stability and ease of deployment, while MoE enable efficient scaling, expert specialization, and resource-aware execution.
> **Position:** Future LLMs can adopt hybrid architectures that integrate a dense core for stable general-purpose capabilities with different expert modules for scalability and specialization.

Dense Architectures. Dense models activate all parameters at every step, resulting in a uniform execution pattern that simplifies both training and deployment. ① Dense architectures typically exhibit more stable convergence, as every parameter is updated in each optimization step. This consistency avoids routing-related instability in MoE training, and underpins the reliability of production-scale systems such as **Llama 3**, **Gemma 3**, and **Claude 3**. ② Dense architectures' simplicity also facilitates deployment, as dense models do not require dynamic routing or expert-specific memory layouts. **GPT-4** adopts dense backbones due to their stable inference behavior and alignment with existing deployment environments. In addition, Long-context systems like **Command A** and **Qwen2.5-1M** demonstrate that up to 256K or 1M token sequences can be processed efficiently. These advantages make dense architectures a reliable foundation [91, 72, 125].

Mixture-of-Experts (MoE) Architectures. MoE architectures increase capacity by routing each token to a small subset of experts. ① MoE enables parameter scaling without linear growth in compute. Models like **Phi-3.5-MoE**, and **DeepSeek-R1** reach massive scale while keeping inference tractable, aided by techniques such as top-k routing and attention compression. ② MoE supports expert specialization, where different subnetworks learn to handle distinct tasks or domains. Mechanisms like SparseMixer (**Phi-3.5-MoE**), expert segmentation (**Qwen2.5-MoE**), global-batch load balancing (**Qwen3-30B-A3B** & **Qwen3-235B-A22B**), and auxiliary-loss-free load balancing (**DeepSeek-V3**) enhance expert diversity and utilization, making MoE promising for multitask and domain-adaptive learning. ③ MoE also reduces FLOPs and runtime cost by activating only a small subset of experts per token, enabling more efficient inference than dense models with similar total parameter count. This makes them appealing for compute-efficient settings, as seen in **Llama 4** and **Gemini 1.5 Pro**, which handle long sequences and multimodal inputs with high efficiency. Building on these strengths, MoE provides a scalable and specialized design pattern, enabling hybrid architectures that combine dense and MoE components for optimal generalization and specialization [31, 55, 122].

3 Design Choices on Post-Training

3.1 How to Select Task and Data for Post-training?

Takeaway: Pretrained LLMs are unlocked to a wide range of tasks during post-training, including reasoning, modeling human preferences, tool use, and multi-modality.
Position: Beyond scaling model size and pre-training data, future LLMs will increasingly emphasize task-specific data-centric post-training. This approach aims to equip models with more complex behaviors that are often rare or underrepresented in the pre-training corpus.

Post-training empowers models with critical real-world abilities that are typically absent from pre-training using diverse types of tasks and data.

Reasoning Task and Data. Enhancing LLM reasoning relies on datasets centered on structured step-by-step problem solving. Most models incorporate chain-of-thought (CoT) data. **Llama 3** and **Llama 4** utilize instruction data with step-by-step solutions. **Qwen2 & 2.5** integrate filtered CoT and math solutions, coding tasks, and logical reasoning questions. **QwQ-32B** emphasizes complex multi-step problems with verified CoT trajectories, solver-checked math, and execution-tested code. **Qwen3** adds a long CoT cold-start stage using diverse synthetic multi-step reasoning data from expert models like Qwen2.5-72B and QwQ-32B. **DeepSeek R1&V3** use curated filtered reasoning prompts to generate around 600K high-quality step-by-step reasoning traces across math, coding, and logic tasks using filtered prompts. **Phi-4** curates CoT data from synthetic and filtered public sources. **Gemma 2&3** are fine-tuned on supervised CoT datasets. **OLMo 2** and **Tülu 3** include CoT demonstrations, math solutions, and logic puzzles, including new logical reasoning queries with step-by-step answers. **Claude 4** adopts a hybrid reasoning architecture and supports an extended thinking mode for more thorough problem solving. It utilizes advanced data cleaning, deduplication, and the collection of high-quality data, such as through trusted crowdsourcing platforms. This trend underscores a shift toward data-centric post-training, where carefully curated reasoning datasets play a pivotal role in instilling capabilities that general pre-training alone cannot achieve [119, 27, 132].

Alignment Task and Human Preference Data. Aligning LLMs with human preferences for helpfulness and safety relies on carefully constructed preference datasets, typically comprising preference pairs. ① Most contemporary LLMs use human feedback, including comparisons or ratings of model outputs, to train reward models or guide fine-tuning. In addition to human annotation, synthetic preference data is often generated using larger models (e.g., GPT-4), as in the UltraFeedback [18] pipeline for **OLMo 2** and **Tülu 3**. ② Safety alignment involves datasets targeting harmful content, sometimes derived from manual red-teaming [89] or evaluation across specific harm categories (e.g., **Llama 3**, **DeepSeek V3/R1**, **Phi-3**, **Gemini 2.0**, **GPT-4**, **Claude 4**). High-quality human preference data enables LLMs to internalize nuanced human values and safety expectations that are underrepresented in pre-training corpora [22, 58, 14].

Agentic Task and Tool-use Data. Integrating tool-use capabilities into LLMs requires specialized datasets that teach models to identify tool-use opportunities, format calls, and interpret results. These datasets often include structured API-call demonstrations, code execution feedback, and simulated agentic dialogues. **Llama 4**, **Gemini 2.0**, and **Gemma 2 and 3** are fine-tuned on structured API-call demonstrations. **Qwen2 and 2.5** incorporate data featuring interactions with tools and APIs, execution-informed coding data, and tool-driven dialogue tasks. **QwQ-32B** also incorporates agent-tool interaction and execution-based coding tasks with environmental feedback. Training models on agentic and tool-use data enables the model to orchestrate complex, multi-step workflows to solve problems beyond the scope of pure language understanding [102, 101, 140].

Multimodal Understanding Task and Data. The development of multimodal capabilities in language models is achieved through training on datasets that integrate information across different modalities, primarily text and vision, but also extending to audio and video. Most multimodal LLMs, including **Llama 4**, **Gemini 2.0**, and **Gemma 3**, were fine-tuned on diverse data types such as text, images, video, and audio to enable the model to perceive, reason over, and generate outputs grounded in multiple modalities. This integration of cross-modal data during post-training exemplifies the data-centric approach needed to equip LLMs with perceptual and contextual understanding beyond what is possible with text-only pre-training [39, 148, 10].

3.2 How to Choose Post-training Strategy?

Takeaway: Post-training incorporates multiple training strategies and reinforcement learning algorithms, with growing reliance on rejection sampling to enhance capabilities.
Position: The future of post-training lies in developing multiple stages that iteratively use diverse strategies, adapting LLMs' behaviors to real-world environments and needs.

Post-Training Strategies. Post-training plays a critical role in aligning LLMs with human preferences and task demands. ① Supervised fine-tuning (SFT) constitutes a foundational step in most LLM post-training pipelines, enabling models to acquire instruction-following capabilities and align with user intent. Models such as **GPT-4**, **Gemini 1.0**, **Gemini 1.5**, and **Claude 3** primarily adopt SFT to instill basic instruction adherence. Others, including **Llama 4**, **Qwen3**, **Kimi k1.5**, **Phi-4**, **OpenELM**, and **DCLM 7B**, apply extended or multi-phase SFT procedures to support broader task coverage and enhanced generalization. ② Multi-stage pipelines incorporating reinforcement learning (RL) are increasingly adopted to refine alignment and performance. **DeepSeek-R1** exemplifies this trend with multi-phase pipelines combining SFT and RL. **Qwen2.5** integrates offline DPO [93] and online GRPO [100], while **Qwen3** executes iterative cycles of SFT and RL. **Llama 4** combines lightweight SFT with continuous online RL and lightweight DPO. **Kimi k1.5** follows a sequence of vanilla SFT, Long-CoT SFT, and RL. **Claude 3** employs Constitutional AI, which combines supervised learning with RL from AI-generated feedback. **Phi-3**, **Phi-4**, and **OpenELM** adopt both SFT and DPO. **Gemma 3** combines distillation with reinforcement objectives with various techniques [99, 96]. ③ Model merging is another complementary technique that aggregates knowledge across checkpoints. **Command A** constructs "SFT Soup" and "RL Soup" by averaging expert models trained on domain-specific data. **Gemma 2** and **OLMo 2** perform checkpoint averaging across SFT and RL stages. These post-training strategies together form a robust framework for aligning LLMs with human intent and advancing their overall capabilities [59, 141, 116].

Reinforcement Learning Algorithms. Reinforcement learning is a core component of post-training, enabling models to align more effectively with human intent, safety preferences, and task-specific demands. ① PPO [98] is a widely used on-policy method adopted by **GPT-4**, **Llama 3**, **Gemini 1.0**, **Tülu 3**, **OLMo 2**, and **ChatGLM** series. ② DPO, a scalable off-policy alternative, is employed by **Llama 3**, **Llama 4** (lightweight tuning), **Qwen2** (offline and online), **Qwen2.5** (offline), **Phi-3** and **Phi-4** (with Pivotal Token Search and judge-guided reward), as well as **OpenELM**, **Tülu 3**, **OLMo 2**, and the **ChatGLM** series. ③ GRPO enhances reasoning via group-based advantage estimation, and is used in **DeepSeek-V2**, **DeepSeek-R1**, **DeepSeek-V3**, **Qwen2.5** (online RL), **Qwen3** (reasoning RL), **Tülu 3.1**, and **OLMo 2 32B** for RLVR. **Gemma 1** adopts an RL algorithm with an LM-based rater. ④ Online Policy Mirror Descent optimizes expected returns under policy regularization toward a reference model [112]. **Kimi k1.5** applies this method with a CoT Reward Model, excluding a value network to improve training efficiency and exploration. ⑤ Other approaches include **Command A**'s

Self-improving Robust Preference Optimization for continual alignment, and the **Claude 3** series optimizes PPO using AI-generated feedback and aligns the model to a predefined constitution. A common pattern is the use of an SFT-based reference model with KL regularization to balance alignment and generalization. **DCLM 7B** is a notable exception, omitting RL entirely after SFT. Collectively, these diverse RL algorithms underscore a clear trend toward fine-grained control in enhancing capabilities and aligning models with human needs [117, 47, 44].

Iterative Training with Rejection Sampling. Rejection sampling is pivotal for enhancing LLM performance, often serving as a strong alternative to RL for refining problem-solving behaviors and overall response quality. ① Rejection sampling has been employed for data quality assurance and model refinement. For instance, **Llama 3** utilizes rejection sampling to choose optimal responses, while **Qwen2** applies it for tasks with definitive answers like math, by generating multiple reasoning paths and preserving those leading to accurate conclusions, a process that also aids in creating preference data. Expanding on this, **Qwen2.5** employs execution feedback-based rejection sampling to filter SFT data for instruction-following, ensuring only high-quality and verified data demonstrating faithful instruction adherence is selected. **Qwen3** further uses this method to generate "thinking" SFT data from earlier stage queries using a later stage model to maintain performance. **DeepSeek-V3** implements rejection sampling after RL training to curate high-quality SFT data using expert models as sources, and **DeepSeek-R1** generates reasoning trajectories by performing rejection sampling from RL checkpoints. Similarly, **OpenELM** uses statistical rejection sampling on the UltraFeedback dataset [19]. ② LLM-powered evaluation and AI judges play a crucial role in more data filtration processes of rejection sampling. **Llama 4** employs AI judges for data filtration, removing more than 50% data for lightweight SFT and RL. **Kimi k1.5** integrates LLMs directly into its rejection sampling process, particularly by compressing its long CoT reasoning into shorter forms. By orchestrating successive stages of rejection sampling alongside complementary tuning methods, post-training pipelines can dynamically refine LLM behaviors to meet diverse real-world demands [88, 129, 130].

3.3 How to Obtain Reward for RL?

Takeaway: Reward signal used for reinforcement learning can be obtained from a diverse sources, including human preference, rule-based verifications, and AI-generated feedback.
Position: The frontier of LLM reward modeling lies in integrating human and AI feedback with automated and verifiable evaluations. Future models should pioneer dynamic, self-improving reward modeling that adapt to new tasks based on user needs and AI capabilities.

Human Preference Reward. Human preference data remains central to reward modeling in LLMs, with leading models adopting various strategies. ① Many models train reward models on human-annotated preference data. Specifically, **GPT-4** trains its reward model using human-ranked responses to predict annotator preferences. **Llama 3** enhances this paradigm by incorporating four graded preference levels for the annotations. **Gemini 1.0** integrates multi-dimensional human evaluations, while **Gemini 1.5** combines RLHF with continuous safety monitoring. **ChatGLM** integrates RL from human feedback and presents **ChatGLM-RLHF** to improve safety, multilingual coherence, and response acceptance in multi-turn settings. In contrast, **Gemma 2** narrows its focus to English-only human preference data to specialize in conversational competence. ② Beyond simple human ranking, some models implement refined reward learning strategies to combat overfitting. **Command A** adopts a two-stage Bradley-Terry-based method, first training on large-scale low-quality preference data, then refining with high-quality data based on strong human preferences. ③ Several models explore hybrid pipeline with offline and online learning to improve hard-to-measure capabilities. **Qwen2** and **Qwen2.5** combine offline and online learning to improve reasoning and factual accuracy, using pre-compiled preference datasets and RL. Specifically, **Qwen 2.5** targets offline training toward reward-intractable skills such as instruction-following. These evolving methodologies demonstrate the effectiveness of human preference data and highlight a growing need for advanced reward modeling methods that move beyond simply using human feedback [67, 54, 154].

Rule-based Rewards. A growing class of LLMs adopts rule-based and verifiable rewards to directly optimize for tasks with measurable correctness. ① Rule-base rewards offer strong supervision in structured domains such as math and coding by enforcing factual and logical consistency. **Kimi k1.5** and **Command A** use code execution and unit tests as reward signals. **OLMo2** introduces RL with Verifiable Rewards (RLVR), rewarding only benchmark-correct solutions. **Tülu 3** employs a binary verifier and observes that initializing RLVR's value function using a general reward model leads to improved performance. **DeepSeek-R1** uses accuracy and format-based rewards to ensure output correctness. **Phi-3** and **Phi-4-Mini** label correct outputs as preferred, and incorrect ones as dis-preferred to create the rewards. ② Models trained with diverse sources of reward signals can further enhance their general capabilities. **Gemma 3** combines rewards from human feedback, executable feedback, and ground-truth labels, maximizing coverage of both creative and verifiable domains. **Qwen3** integrates rule-based correctness, model-based rewards with and without reference answers. **DeepSeek-V3** applies rule-based rewards for deterministic tasks and model-based preference rewards for subjective tasks. **QwQ-32B** follows a staged RL process, starting with rule-based rewards (e.g., accuracy checkers and code test), and transitioning to general RL using reward models and verifiers. Together, these models showcase a maturing reward modeling strategy in which diverse rewards are important in achieving alignment and driving performance improvement [87, 127, 113, 104].

AI-based Reward Signal. RL from AI Feedback (RLAIF) presents an alternative to RLHF by utilizing powerful LLMs to generate preference labels. ① Several LLMs combine human annotations with AI-generated feedback for efficient preference modeling. **Claude 3** series employs the reward model trained on both human and AI-generated preference data to support subsequent RLAIF fine-tuning. Similarly, **Qwen2** and **Qwen2.5** adopt RLAIF and curate preference pairs using both human and automated review. ② Advanced models like **GPT-4o** are increasingly used as automatic judges to reduce reliance on human raters. **Phi-4** uses GPT-4o to label preference pairs, while **Tülu 3** and **OLMo 2** prompt GPT-4o to rate completions based on helpfulness, truthfulness, honesty, and instruction-following, yielding rich feedback signals. ③ AI-based reward has been adopted for multi-stage RL training. **Llama 3** and **Llama 4** select top candidates via reward models or the model itself to conduct multi-round RL with rejection sampling, without external human labeling. **Claude 4** uses RL for prompt injection resistance. These diverse reward modeling strategies highlight a growing shift toward more autonomous and scalable reward generation systems that can accommodate both user needs and AI capabilities [121, 46, 110].

4 Design Choices on Optimization

4.1 How to Design Prompt Template?

> **Takeaway:** Prompt design plays a critical role in LLM training and inference, spanning system prompting, standard prompting, and reasoning prompting.
> **Position:** Unifying and systematizing prompt design, across role formatting, and reasoning scaffolds, is critical for advancing LLM reliability and generalization.

System Prompting. System prompts initialize the behavior of LLMs for user interactions. Currently, **Grok**, **Claude 3**, **Claude 4**, **ChatGPT** series, **Gemini** series employ long, manually curated system prompts that define tool usage, search strategies, content generation rules, and even specific behavioral fixes for known LLM quirks [156, 124]. Released or leaked examples reveal details such as API interaction protocols, verbosity controls (e.g., OpenAI o3/o4-mini API's "Yap score"), multi-tool integration (e.g., OpenAI o4-mini with generation and automations), and safety policies (e.g., ChatGPT-4o multimodal safety). This highlights the need for standardizing system prompting practices as a foundation for more reliable and generalizable LLM behavior [17, 149].

Standard Prompting. Many mainstream LLMs use structured prompt templates to define conversational roles and guide dialogue flow. For example, **Qwen** series follows the structure with `<|im_start|>` and `<|im_end|>` tags. **Llama** family uses tokens `<|start_header_id|>role<|end_header_id|>` to indicate roles and wraps prompts with `<|begin_of_text|>` and `<|end_of_text|>`. **DeepSeek-V3** adopts similar special tokens. **Phi-3** adopts a minimal structure with tags like `<|system|>`, `<|user|>`, `<|assistant|>`, and `<|end|>`. In contrast, **Gemini 2.0** emphasizes natural language task formulation and persona cues, rather than relying on explicit token-based role tags. **GPT-4** uses JSON-style messages (`system`, `user`, `assistant`) but also handles natural language prompts well. This diversity in prompt structuring highlights the need for a unified framework to systematically design and evaluate prompting strategies for reliable LLM behavior [38, 80, 13].

Reasoning Prompting. Reasoning prompt enables LLMs to generate thinking process and solve complex problems. **Qwen3** and **Qwen2.5-math** supports structured reasoning prompts. **QwQ-32B** introduces `<think>` tokens to externalize the intermediate reasoning steps, supporting "reason step by step" prompt and box the final answer `\boxed{}` to enhance clarity. **DeepSeek-R1** omits system messages and instead uses <think> and <answer> tags. Effective prompts for **DeepSeek-R1** explicitly request stepwise reasoning, self-reflection, and comparison of solution paths, along with clear output format instructions. **Gemini 2.0** and **2.5** are designed to output internal reasoning and respond well to prompts encouraging sequential explanations. **GPT-4** and **Claude 3** also excel in complex reasoning tasks and benefit from CoT prompts like "Let's think this through step by step". Ultimately, unifying role formatting and structured reasoning scaffolds into a cohesive prompting framework empowers LLMs to decompose complex problems into intermediate steps and consistently generalize their thinking capability across diverse tasks [151, 133, 66].

4.2 How to Minimize Memory Usage?

> **Takeaway:** Memory efficiency in LLMs is achieved through a multifaceted approach involving quantization, efficient attention mechanisms, and memory-saving training tricks.
> **Position:** As LLMs continue to scale and diversify in deployment environments, the next frontier in memory optimization lies in designing hardware-aware frameworks that adapt memory-saving strategies based on computational context and task requirements.

Quantization. To minimize memory usage in LLMs, a spectrum of quantization techniques spanning 16-bit, 8-bit, and 4-bit precision [71, 76, 77] have been developed. ① Models such as **Qwen2.5** and **ChatGLM** series (e.g., **GLM-4** and **GLM-4-Air**) typically utilize BF16 precision, balancing memory savings and accuracy for general deployment. ② Moving toward lower precision, models like **Llama 3**, **Llama 4 Behemoth**, **DeepSeek-V3**, and **Command A** leverage FP8 quantization to significantly enhance computational efficiency. Specifically, **Llama 3** implements FP8 quantization to boost throughput by 50% during inference, using dynamic scaling factors [126] for precision retention. Similarly, **Llama 4 Behemoth** employs FP8 during pre-training, while **DeepSeek-V3** integrates fine-grained online quantization strategies including tile-wise and block-wise grouping, caching activations in FP8 and optimizing memory during MoE training. **Command A** also exploits FP8 tensor cores for computational efficiency, though it maintains weights and optimizer state in FP32 for accuracy preservation. ③ At the ultra-low precision end, lightweight models including **phi-3-mini**, **Gemini 1.0 Nano**, and **ChatGLM-6B** adopt aggressive 4-bit quantization strategies optimized explicitly for resource-constrained environments like smartphones. **Gemma 3** employs quantization-aware training methods [51] to produce highly optimized, low-precision model variants. Collectively, these advancements highlight the critical role of hardware-aware, context-adaptive quantization strategies in optimizing memory use across diverse computational environments [56, 128, 57, 131].

Efficient Attention Mechanisms. To enhance memory efficiency in LLMs, many models employ optimized efficient attention mechanisms, such as Grouped Query Attention (GQA), key-value (KV) caching, sparse attention, and FlashAttention [42, 146, 21]. ① GQA reduces compute and memory usage by sharing keys and values across queries. It is widely adopted in **Qwen2**, **Gemma 3**, **Llama 3**, **Aya 23**, and **Command A**. ② KV cache can reduce the inference memory, as demonstrated in **Llama 3**, **Qwen2**, **Qwen2.5**, **OpenELM**, and **GLM-4**. Moreover, **Llama 3** implements PagedAttention [60] to dynamically allocate KV caches for higher throughput. **Gemma 3** adopts FP8 quantization on KV cache storage, addressing memory bottlenecks by compressing cache storage demands while sustaining long-context capabilities. ③ Sparse and hybrid attention techniques are introduced by several models to further optimize KV cache memory use. Specifically, **Phi-3-small** employs blocksparse attention modules, alternating between sparse and dense layers to optimize KV cache savings while maintaining long context retrieval performance, while **Phi-4-Mini** reduces KV cache consumption to one-third of standard size. **DeepSeek-V3** utilizes multi-head latent attention for superior cache compression compared to regular multi-head attention. In contrast, **Mistral 7B** uses a rolling buffer cache coupled with sliding window attention to cut cache usage by 8x without quality loss. **Command A** applies a hybrid architecture of interleaved sliding window and full attention in a 3:1 ratio, significantly reducing memory footprint. ④ FlashAttention is adopted in multiple models as another key technique: **Phi-3-small** builds a custom Triton kernel for training, **Mistral 7B** enhances its sliding window attention with FlashAttention for a 2x speedup over vanilla attention, **ChatGLM-6B** expands context length from 2K to 32K using FlashAttention, and **OpenELM** leverages it to compute scaled dot-product attention. Together, these advancements highlight the growing importance and widespread adoption of memory-optimized attention strategies in modern LLM development [79, 143, 90].

Memory-Saving Training Tricks. Beyond quantization and efficient attention, several models implement distinct memory-saving training techniques to reduce GPU usage [61, 73]. Notably, **Tülu 3** introduces two training tricks: caching DPO reference log probabilities to avoid keeping the model in GPU memory, and using separate forward passes for chosen and rejected sequences instead of concatenating them. This reduces GPU memory usage from about 60% to 40% on 8xH100 clusters without affecting training loss. Similarly, **DeepSeek-V3** adopts a multifaceted strategy: ① recomputing all RMSNorm and Multi-latent Attention up-projections during backpropagation, ② storing model parameters on CPU, and ③ sharing embedding and output head parameters and gradients. As these examples show, the next phase of memory efficiency will require dynamically adjusted memory-saving strategies in line with the training tasks and computational environment [139, 5, 145].

4.3 How to Obtain a Smaller Model?

> **Takeaway:** Distillation has become the dominant strategy for transferring capabilities and performance from larger LLMs to smaller models.
> **Position:** As the field matures, we envision a layered LLM ecosystem: a few ever-evolving "super teacher" models pushing the boundaries of intelligence, paired with a diverse landscape of compact student models that are distilled and trained for targeted domains and tasks.

Anchored by our takeaway, three broad patterns dominate the current practice of distillation in obtaining a smaller model, including intra-family, cross-family, and task-targeted distillation [41, 34]. ① Intra-family distillation compresses capabilities within the same model family: Meta distills **Llama 4 Maverick** from **Llama 4 Behemoth** with an innovative loss function that dynamically adjust the weights of soft and hard targets. Alibaba applies this strategy by distilling large models (**Qwen3-32B**, **Qwen3-235B-A22B**) into the smaller models (**Qwen3-14B/8B/4B/1.7B/0.6B**, **Qwen3-30B-A3B**). **DeepSeek-V3** inherits reasoning capabilities from **DeepSeek-R1**. Google distills **Gemini 1.0 Nano** and **Gemini 1.5 Flash** from their larger Gemini models. Moreover, **Gemma 2** trains the smaller models with distillation instead of relying on the next token prediction objective, and all **Gemma 3** models across varying model sizes are trained with distillation. ② Cross-family distillation indicates the learning of a smaller model from a different model family. For example, Phi series (**Phi-3**, **Phi-4**, and **Phi-4-Mini**) leverage distillation to learn from a strong teacher model such as GPT-4 and GPT-4o, demonstrating the capability of building a strong but smaller LLM cross model families. ③ Task-targeted distillation aims to specialize smaller models for a particular task or capability. To illustrate, **DeepSeek-R1** demonstrates strong reasoning performance on a variety of tasks, and to transfer this capability, **DeepSeek-R1-Distill-Qwen** and **DeepSeek-R1-Distill-Llama** were trained using the 800k-sample dataset curated by DeepSeek-R1 itself. Similarly, **Command A** distills smaller models using synthetic data generated by multiple larger teacher models, each specialized in distinct tasks such as coding, explanation, or reasoning. **Kimi k1.5** adopts a similar strategy by distilling the reasoning capabilities and thinking priors from long-CoT models into more efficient short-CoT models. Collectively, these efforts verify the effectiveness of distillation and point toward a future where a few frontier models enables the emergence of task-specific efficient models and sustains an expanding and layered LLM ecosystem [111, 86, 20, 123].

4.4 How to Accelerate Training and Inference?

> **Takeaway:** Existing methods depend heavily on parallelism strategies and hardware-aware optimizations to accelerate LLM training and inference.
> **Position:** The next frontier in LLM acceleration lies in integrating hardware-adaptive parallelism that aligns with model architectures and balances training and inference workloads intelligently across heterogeneous compute resources.

Although minimizing memory usage (Section 4.2) and training a smaller model (Section 4.3) can incidentally lead to acceleration, the following techniques are specifically designed for accelerating training and inference.

Inference Parallelization. To accelerate inference, LLMs increasingly adopt hybrid parallelism aligned with model architecture and hardware [95, 24, 155]. ① Many LLMs accelerate inference through parallelism strategies such as tensor, pipeline, data, and expert parallelism. **Tülu 3** uses 16-way tensor parallelism with vLLM. **DeepSeek-V3** uses 4-way tensor, sequence, and 8-way data parallelism for attention, 32-way expert parallelism for MoE, and 1-way tensor for dense MLPs.

it also explores multi-token prediction for performance and speculative decoding. ② However, employing every parallelism method is not always optimal. **Llama 3** adopts pipeline but avoids tensor parallelism due to inter-node bandwidth and latency limits, highlighting the need for hardware-aware parallelism strategies. ③ Parallelization can be integrated with hardware optimization techniques and synchronization to further accelerate processing speed. **Qwen2.5-1M** combines multi-stage pipeline parallelism with sparse attention kernels and proposes dynamic chunked pipeline parallelism, which balances chunk sizes during long-context prefilling to eliminate pipeline bubbles. Additionally, **Qwen2.5-1M** adopts the totally asynchronous generator to decouple the scheduler, model runner, and decoder into separate processes, alleviating synchronization dependencies. Similarly, **Phi-3** implements a kernel for the prefilling phase and extends the paged attention kernel in vLLM for the decoding phase. Together, these approaches illustrate the trend toward hardware-aware inference parallelism strategies that push the boundaries of LLM deployment efficiency [64, 16, 142].

Training Efficiency. Modern LLM training efficiency is increasingly determined by how well fine-grained parallelism and parameter-efficient techniques are orchestrated [147, 49, 70]. ① LLMs employ a variety of parallelism techniques to enhance training throughput and efficiency. Models such as **Llama 3**, **Command A**, **Kimi k1.5**, **Gemini 1.0**, and **Gemma 2** use combinations of tensor, pipeline, expert, and data parallelisms to scale across large clusters. Specifically, **Llama 3** employs tensor, pipeline, context, and data parallelisms and optimizes their order to match network hierarchy: inner parallelism requires low-latency and high-bandwidth, while outer tolerates multi-hop latency. **Command A** uses a JAX-based [25] distributed training framework to implement complex sharding, including data, fully sharded data, sequence, and tensor parallelisms. Meanwhile, **Kimi k1.5** only keep tensor parallelism in the checkpoints and relies on Megatron [103] to implement pipeline and expert parallelisms. At the infrastructure level, **Gemini 1.0** and **Gemma 2** both employ model and data parallelisms across Google's TPU clusters to effectively scale training. ② Expert parallelism is further emphasized in architectures involving MoEs. **Mixtral 8x7B** applies model and expert parallelism to distribute MoE layers across GPUs, while **DeepSeek-V3** employs 64-way expert, 16-way pipeline, and ZeRO-1 data parallelism to achieve fine-grained model distribution over 8 nodes. ③ Parameter-efficient training methods like low-rank approximations [37, 45] help accelerate training by reducing the number of trained parameters. For example, **OpenELM** integrates LoRA [43] and DoRA [74], while **DeepSeek-V3** compresses key-value matrices with low-rank projection. Overall, these strategies reveal that scalable and efficient LLM training depends on parallelism strategies and training efficiency techniques [115, 114, 120].

Maximizing hardware utilization. LLMs accelerate training/inference by maximizing hardware utilization via diverse strategies [69, 97, 11, 50]. ① Several models leverage low-level software optimizations and specialized compilation. Specifically, **OLMo 2** boosts efficiency by compiling PyTorch [6], reducing sync, offloading bookkeeping, and explicitly managing garbage collection. In addition, **Mixtral 8x7B** boosts execution speed through high-performance specialized GPU kernel using Megablocks [26]. ② Hardware-aware design principles are also crucial. **DeepSeek-V3** adapts to current hardware and proposes future chips integrate higher precision, group-scaled Tensor Cores, fused FP8 cast/memory operations, and direct transposed reads. A series of **Gemini** and **Gemma** models utilize frameworks like JAX [12] and ML Pathways to effectively leverage the latest generation of hardware. ③ Efficient workload management can further increase hardware utilization. Beyond Megatron and vLLM, **Kimi k1.5** uses Kubernetes Sidecars to share all available GPUs and co-locate training and inference in a single pod. ④ Hardware utilization can also be enhanced through asynchronous training strategies, especially for RL. For example, **Tülu 3** uses async PPO to decouple inference and training, cutting GPU idle time. **Llama 4**'s async RL framework enables flexible GPU allocation and achieves a 10x training efficiency gain over **Llama 3**. Ultimately, these approaches reflect a growing emphasis on harmonizing software, hardware, and scheduling to intelligently balance training and inference across diverse computational infrastructures [68, 48, 144, 153].

5 Conclusion

Our analysis of 31 influential LLMs reveals that the next generation of language models will be shaped by strategic design choices that blend decoder-only architectures with modular enhancements, adaptive post-training, and hardware-aware optimization. By synthesizing current trends and proposing forward-looking directions, this paper provides a roadmap for building LLMs that are more capable, efficient, and aligned with real-world demands and deployment constraints.

References

[1] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024. https://arxiv.org/pdf/2404.14219.

[2] Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024. https://arxiv.org/pdf/2412.08905.

[3] Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025. https://arxiv.org/pdf/2503.01743.

[4] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. https://arxiv.org/pdf/2303.08774.

[5] Keivan Alizadeh, Seyed Iman Mirzadeh, Dmitry Belenko, S Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. Llm in a flash: Efficient large language model inference with limited memory. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. https://aclanthology.org/2024.acl-long.678.pdf.

[6] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, et al. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2024. https://dl.acm.org/doi/pdf/10.1145/3620665.3640366.

[7] Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

[8] Anthropic. System card: Claude opus 4 & claude sonnet 4, 2025. https://www-cdn.anthropic.com/6be99a52cb68eb70eb9572b4cafad13df32ed995.pdf.

[9] Viraat Aryabumi, John Dang, Dwarak Talupuru, Saurabh Dash, David Cairuz, Hangyu Lin, Bharat Venkitesh, Madeline Smith, Jon Ander Campos, Yi Chern Tan, et al. Aya 23: Open weight releases to further multilingual progress. *arXiv preprint arXiv:2405.15032*, 2024. https://arxiv.org/pdf/2405.15032.

[10] Tianyi Bai, Hao Liang, Binwang Wan, Yanran Xu, Xi Li, Shiyu Li, Ling Yang, Bozhou Li, Yifan Wang, Bin Cui, et al. A survey of multimodal large language model from a data-centric perspective. *arXiv preprint arXiv:2405.16640*, 2024. https://arxiv.org/pdf/2405.16640.

[11] Paul Barham, Aakanksha Chowdhery, Jeff Dean, Sanjay Ghemawat, Steven Hand, Daniel Hurt, Michael Isard, Hyeontaek Lim, Ruoming Pang, Sudip Roy, et al. Pathways: Asynchronous distributed dataflow for ml. *Proceedings of Machine Learning and Systems*, 2022. https://proceedings.mlsys.org/paper_files/paper/2022/file/37385144cac01dff38247ab11c119e3c-Paper.pdf.

[12] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs, 2018. http://github.com/google/jax.

[13] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020. https://arxiv.org/pdf/2005.14165.

[14] Souradip Chakraborty, Jiahao Qiu, Hui Yuan, Alec Koppel, Furong Huang, Dinesh Manocha, Amrit Singh Bedi, and Mengdi Wang. Maxmin-rlhf: Alignment with diverse human preferences. *arXiv preprint arXiv:2402.08925*, 2024. https://arxiv.org/pdf/2402.08925.

[15] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 2024. https://arxiv.org/pdf/2307.03109.

[16] Jaehong Cho, Minsu Kim, Hyunmin Choi, Guseul Heo, and Jongse Park. Llmservingsim: A hw/sw co-simulation infrastructure for llm inference serving at scale. In *IEEE International Symposium on Workload Characterization*, 2024. https://ieeexplore.ieee.org/abstract/document/10763697.

[17] Yumin Choi, Jinheon Baek, and Sung Ju Hwang. System prompt optimization with meta-learning. *arXiv preprint arXiv:2505.09666*, 2025. https://arxiv.org/pdf/2505.09666.

[18] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, et al. Ultrafeedback: Boosting language models with scaled ai feedback. In *International Conference on Machine Learning*, 2024. https://arxiv.org/pdf/2310.01377.

[19] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*, 2023. https://arxiv.org/pdf/2310.01377.

[20] Yu Cui, Feng Liu, Pengbo Wang, Bohao Wang, Heng Tang, Yi Wan, Jun Wang, and Jiawei Chen. Distillation matters: empowering sequential recommenders to match the performance of large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, 2024. https://arxiv.org/pdf/2405.00338.

[21] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, 2022. https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf.

[22] Xun Deng, Han Zhong, Rui Ai, Fuli Feng, Zheng Wang, and Xiangnan He. Less is more: Improving llm alignment via preference data selection. *arXiv preprint arXiv:2502.14560*, 2025. https://arxiv.org/pdf/2502.14560.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American Chapter of the Association for Computational Linguistics*, 2019. https://aclanthology.org/N19-1423.pdf.

[24] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 2022. https://www.jmlr.org/papers/volume23/21-0998/21-0998.pdf.

[25] Roy Frostig, Matthew James Johnson, and Chris Leary. Compiling machine learning programs via high-level tracing. *Systems for Machine Learning*, 2018. https://mlsys.org/Conferences/doc/2018/146.pdf.

[26] Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. In *Proceedings of Machine Learning and Systems*, 2023. https://proceedings.mlsys.org/paper_files/paper/2023/file/5a54f79333768effe7e8927bcccffe40-Paper-mlsys2023.pdf.

[27] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025. https://arxiv.org/pdf/2503.01307.

[28] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. https://arxiv.org/pdf/2503.19786.

[29] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024. https://arxiv.org/pdf/2408.00118.

[30] Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024. https://arxiv.org/pdf/2406.12793.

[31] Seokjin Go and Divya Mahajan. Moetuner: Optimized mixture of expert serving with balanced expert placement and token routing. *arXiv preprint arXiv:2502.06643*, 2025. https://arxiv.org/pdf/2502.06643.

[32] Google DeepMind. Introducing gemini 2.0: our new ai model for the agentic era, 2024. https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/.

[33] Google DeepMind. Gemini 2.5: Our most intelligent ai model, 2025. https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/.

[34] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 2021. https://dl.acm.org/doi/pdf/10.1145/3701551.3703577.

[35] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. https://arxiv.org/pdf/2407.21783.

[36] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. https://arxiv.org/pdf/2501.12948.

[37] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Transactions on Machine Learning Research*, 2024. https://openreview.net/forum?id=lIsCS8b6zj.

[38] Jia He, Mukund Rungta, David Koleczek, Arshdeep Sekhon, Franklin X Wang, and Sadid Hasan. Does prompt formatting have any impact on llm performance? *arXiv preprint arXiv:2411.10541*, 2024. https://arxiv.org/pdf/2411.10541.

[39] Muyang He, Yexin Liu, Boya Wu, Jianhao Yuan, Yueze Wang, Tiejun Huang, and Bo Zhao. Efficient multimodal learning from data-centric perspective. *arXiv preprint arXiv:2402.11530*, 2024. https://arxiv.org/pdf/2402.11530.

[40] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020. https://arxiv.org/pdf/2006.03654.

[41] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. https://arxiv.org/pdf/1503.02531.

[42] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. In *Advances in Neural Information Processing Systems*, 2024. https://proceedings.neurips.cc/paper_files/paper/2024/file/028fcbcf85435d39a40c4d61b42c99a4-Paper-Conference.pdf.

[43] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 2022. https://arxiv.org/pdf/2106.09685.

[44] Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025. https://arxiv.org/pdf/2501.03262.

[45] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. https://aclanthology.org/2023.emnlp-main.319/.

[46] Chenghua Huang, Zhizhen Fan, Lu Wang, Fangkai Yang, Pu Zhao, Zeqi Lin, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. Self-evolved reward learning for llms. *arXiv preprint arXiv:2411.00418*, 2024. https://arxiv.org/pdf/2411.00418.

[47] Chenghua Huang, Lu Wang, Fangkai Yang, Pu Zhao, Zhixu Li, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. Lean and mean: Decoupled value policy optimization with global value guidance. *arXiv preprint arXiv:2502.16944*, 2025. https://arxiv.org/pdf/2502.16944.

[48] Mingqiang Huang, Ao Shen, Kai Li, Haoxiang Peng, Boyu Li, Yupeng Su, and Hao Yu. Edgellm: A highly efficient cpu-fpga heterogeneous edge accelerator for large language models. *IEEE Transactions on Circuits and Systems I*, 2025. https://arxiv.org/pdf/2407.21325.

[49] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems*, 2019. https://proceedings.neurips.cc/paper_files/paper/2019/file/093f65e080a295f8076b1c5722a46aa2-Paper.pdf.

[50] Yingbing Huang, Lily Jiaxin Wan, Hanchen Ye, Manvi Jha, Jinghua Wang, Yuhong Li, Xiaofan Zhang, and Deming Chen. New solutions on llm acceleration, optimization, and application. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024. https://dl.acm.org/doi/pdf/10.1145/3649329.3663517.

[51] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Conference on Computer Vision and Pattern Recognition*, 2018. https://openaccess.thecvf.com/content_cvpr_2018/papers/Jacob_Quantization_and_Training_CVPR_2018_paper.pdf.

[52] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. https://arxiv.org/pdf/2310.06825.

[53] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024. https://arxiv.org/pdf/2401.04088.

[54] Ruili Jiang, Kehai Chen, Xuefeng Bai, Zhixuan He, Juntao Li, Muyun Yang, Tiejun Zhao, Liqiang Nie, and Min Zhang. A survey on human preference learning for large language models. *arXiv preprint arXiv:2406.11191*, 2024. https://arxiv.org/pdf/2406.11191.

[55] Chao Jin, Ziheng Jiang, Zhihao Bai, Zheng Zhong, Juncai Liu, Xiang Li, Ningxin Zheng, Xi Wang, Cong Xie, Wen Heng, et al. Megascale-moe: Large-scale communication-efficient training of mixture-of-experts models in production. *arXiv preprint arXiv:2505.11432*, 2025. https://arxiv.org/pdf/2505.11432.

[56] Rohan Juneja, Shivam Aggarwal, Safeen Huda, Tulika Mitra, and Li-Shiuan Peh. Halo: Hardware-aware quantization with low critical-path-delay weights for llm acceleration. *arXiv preprint arXiv:2502.19662*, 2025. https://arxiv.org/pdf/2502.19662.

[57] Christoforos Kachris. A survey on hardware accelerators for large language models. *Applied Sciences*, 2025. https://www.mdpi.com/2076-3417/15/2/586.

[58] Dongyoung Kim, Kimin Lee, Jinwoo Shin, and Jaehyung Kim. Spread preference annotation: Direct preference judgment for efficient llm alignment. In *International Conference on Learning Representations*, 2025. https://openreview.net/pdf?id=BPgK5XW1Nb.

[59] Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*, 2025. https://arxiv.org/pdf/2502.21321.

[60] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023. https://arxiv.org/pdf/2309.06180,.

[61] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024. https://arxiv.org/pdf/2411.15124.

[62] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019. https://arxiv.org/pdf/1909.11942.

[63] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019. https://arxiv.org/pdf/1910.13461.

[64] Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. Llm inference serving: Survey of recent advances and opportunities. *arXiv preprint arXiv:2407.12391*, 2024. https://arxiv.org/pdf/2407.12391.

[65] Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems*, 2024. https://arxiv.org/pdf/2406.11794.

[66] Jia Li, Ge Li, Yongmin Li, and Zhi Jin. Structured chain-of-thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology*, 2025. https://arxiv.org/pdf/2305.06599.

[67] Jiaxiang Li, Siliang Zeng, Hoi-To Wai, Chenliang Li, Alfredo Garcia, and Mingyi Hong. Getting more juice out of the sft data: Reward learning from human demonstration improves sft for llm alignment. In *Advances in Neural Information Processing Systems*, 2024. https://proceedings.neurips.cc/paper_files/paper/2024/file/e0c9b65fb3e41aaa86576df3ec33ad2e-Paper-Conference.pdf.

[68] Jindong Li, Tenglong Li, Guobin Shen, Dongcheng Zhao, Qian Zhang, and Yi Zeng. Pushing up to the limit of memory bandwidth and capacity utilization for efficient llm decoding on embedded fpga. *arXiv preprint arXiv:2502.10659*, 2025. https://arxiv.org/pdf/2502.10659.

[69] Jinhao Li, Jiaming Xu, Shan Huang, Yonghua Chen, Wen Li, Jun Liu, Yaoxiu Lian, Jiayi Pan, Li Ding, Hao Zhou, et al. Large language model inference acceleration: A comprehensive hardware perspective. *arXiv preprint arXiv:2410.04466*, 2024. https://arxiv.org/pdf/2410.04466.

[70] Shenggui Li, Fuzhao Xue, Chaitanya Baranwal, Yongbin Li, and Yang You. Sequence parallelism: Long sequence training from system perspective. *arXiv preprint arXiv:2105.13120*, 2021. https://arxiv.org/pdf/2105.13120.

[71] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In *Proceedings of Machine Learning and Systems*, 2024. https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf.

[72] Xihui Lin, Yunan Zhang, Suyu Ge, Liliang Ren, Barun Patra, Vishrav Chaudhary, Hao Peng, and Xia Song. S2-attention: Hardware-aware context sharding among attention heads. *arXiv preprint arXiv:2407.17678*, 2024. https://arxiv.org/pdf/2407.17678.

[73] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. https://arxiv.org/pdf/2412.19437.

[74] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning*, 2024. https://openreview.net/pdf?id=3d5CIRG1n2.

[75] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. https://arxiv.org/pdf/1907.11692.

[76] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. https://aclanthology.org/2024.findings-acl.26/.

[77] Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: LLM quantization with learned rotations. In *International Conference on Learning Representations*, 2025. https://openreview.net/forum?id=ogO6DGE6FZ.

[78] Alejo Lopez-Avila and Jinhua Du. A survey on large language models in multimodal recommender systems. *arXiv preprint arXiv:2505.09777*, 2025. https://arxiv.org/pdf/2505.09777.

[79] Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Neo Y. Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025. https://arxiv.org/pdf/2502.13189.

[80] Qianou Ma, Weirui Peng, Chenyang Yang, Hua Shen, Kenneth Koedinger, and Tongshuang Wu. What should we engineer in prompts? training humans in requirement-driven llm use. *ACM Transactions on Computer-Human Interaction*, 2025. https://dl.acm.org/doi/pdf/10.1145/3731756.

[81] Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, et al. Openelm: An efficient language model family with open training and inference framework. *arXiv preprint arXiv:2404.14619*, 2024. https://arxiv.org/pdf/2404.14619.

[82] Meta AI. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation, 2025. https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

[83] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 2023. https://arxiv.org/pdf/2111.01243.

[84] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*, 2023. https://arxiv.org/pdf/2307.06435.

[85] Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024. https://arxiv.org/pdf/2501.00656.

[86] Hao Peng, Xin Lv, Yushi Bai, Zijun Yao, Jiajie Zhang, Lei Hou, and Juanzi Li. Pre-training distillation for large language models: A design space exploration. *arXiv preprint arXiv:2410.16215*, 2024. https://arxiv.org/pdf/2410.16215.

[87] Hao Peng, Yunjia Qi, Xiaozhi Wang, Zijun Yao, Bin Xu, Lei Hou, and Juanzi Li. Agentic reward modeling: Integrating human preferences with verifiable correctness signals for reliable reward systems. *arXiv preprint arXiv:2502.19328*, 2025. https://arxiv.org/pdf/2502.19328.

[88] Xiangyu Peng, Congying Xia, Xinyi Yang, Caiming Xiong, Chien-Sheng Wu, and Chen Xing. Regenesis: Llms can grow into reasoning generalists via self-improvement. In *Advances in Neural Information Processing Systems*, 2025. https://openreview.net/pdf?id=YUYJsHOf3c.

[89] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Conference on Empirical Methods in Natural Language Processing*, 2022. https://arxiv.org/pdf/2202.03286.

[90] Ramya Prabhu, Ajay Nayak, Jayashree Mohan, Ramachandran Ramjee, and Ashish Panwar. vattention: Dynamic memory management for serving llms without pagedattention. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2025. https://doi.org/10.1145/3669940.3707256.

[91] Krishna C Puvvada, Faisal Ladhak, Santiago Akle Serrano, Cheng-Ping Hsieh, Shantanu Acharya, Somshubra Majumdar, Fei Jia, Samuel Kriman, Simeng Sun, Dima Rekesh, et al. Swan-gpt: An efficient and scalable approach for long-context language modeling. *arXiv preprint arXiv:2504.08719*, 2025. https://arxiv.org/pdf/2504.08719.

[92] Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, 2024. https://qwenlm.github.io/blog/qwq-32b/.

[93] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023. https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

[94] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 2020. https://arxiv.org/pdf/1910.10683.

[95] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020. https://arxiv.org/pdf/1910.02054.

[96] Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. Warm: On the benefits of weight averaged reward models. *arXiv preprint arXiv:2401.12187*, 2024. https://arxiv.org/pdf/2401.12187.

[97] Adam Roberts, Hyung Won Chung, Gaurav Mishra, Anselm Levskaya, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, et al. Scaling up models and data with t5x and seqio. *Journal of Machine Learning Research*, 2023. https://www.jmlr.org/papers/volume24/23-0795/23-0795.pdf.

[98] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. https://arxiv.org/pdf/1707.06347.

[99] Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Nino Vieillard, Alexandre Ramé, Bobak Shariari, Sarah Perrin, Abe Friesen, Geoffrey Cideron, et al. Bond: Aligning llms with best-of-n distillation. *arXiv preprint arXiv:2407.14622*, 2024. https://arxiv.org/pdf/2407.14622.

[100] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. https://arxiv.org/pdf/2402.03300.

[101] Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. Small llms are weak tool learners: A multi-llm agent. *arXiv preprint arXiv:2401.07324*, 2024. https://arxiv.org/pdf/2401.07324.

[102] Zhengliang Shi, Shen Gao, Lingyong Yan, Yue Feng, Xiuyi Chen, Zhumin Chen, Dawei Yin, Suzan Verberne, and Zhaochun Ren. Tool learning in the wild: Empowering language models as automatic tool agents. In *Proceedings of the ACM on Web Conference*, 2025. https://dl.acm.org/doi/pdf/10.1145/3696410.3714825.

[103] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019. https://arxiv.org/pdf/1909.08053.

[104] Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. Crossing the reward bridge: Expanding rl with verifiable rewards across diverse domains. *arXiv preprint arXiv:2503.23829*, 2025. https://arxiv.org/pdf/2503.23829.

[105] Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, et al. Ul2: Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*, 2022. https://arxiv.org/pdf/2205.05131.

[106] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. https://arxiv.org/pdf/2312.11805.

[107] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024. https://arxiv.org/pdf/2403.05530.

[108] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. https://arxiv.org/pdf/2403.08295.

[109] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025. https://arxiv.org/pdf/2501.12599.

[110] Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Lei Han, Haitao Mi, and Dong Yu. Toward self-improvement of llms via imagination, searching, and criticizing. In *Advances in Neural Information Processing Systems*, 2024. https://proceedings.neurips.cc/paper_files/paper/2024/file/5e5853f35164e434015716a8c2a66543-Paper-Conference.pdf.

[111] Yijun Tian, Yikun Han, Xiusi Chen, Wei Wang, and Nitesh V. Chawla. Beyond answers: Transferring reasoning capabilities to smaller llms using multi-teacher knowledge distillation. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, 2025. https://doi.org/10.1145/3701551.3703577.

[112] Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. *arXiv preprint arXiv:2005.09814*, 2020. https://arxiv.org/pdf/2005.09814.

[113] Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Lucas Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, et al. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*, 2025. https://arxiv.org/pdf/2504.20571.

[114] Yujie Wang, Shiju Wang, Shenhan Zhu, Fangcheng Fu, Xinyi Liu, Xuefeng Xiao, Huixia Li, Jiashi Li, Faming Wu, and Bin Cui. Flexsp: Accelerating large language model training via flexible sequence parallelism. *arXiv preprint arXiv:2412.01523*, 2025. https://arxiv.org/pdf/2412.01523.

[115] Zheng Wang, Anna Cai, Xinfeng Xie, Zaifeng Pan, Yue Guan, Weiwei Chu, Jie Wang, Shikai Li, Jianyu Huang, Chris Cai, et al. Wlb-llm: Workload-balanced 4d parallelism for large language model training. *arXiv preprint arXiv:2503.17924*, 2025. https://arxiv.org/pdf/2503.17924.

[116] Zhenting Wang, Guofeng Cui, Kun Wan, and Wentian Zhao. Dump: Automated distribution-level curriculum learning for rl-based llm post-training. *arXiv preprint arXiv:2504.09710*, 2025. https://arxiv.org/pdf/2504.09710.

[117] Zhichao Wang, Bin Bi, Shiva Kumar Pentyala, Kiran Ramnath, Sougata Chaudhuri, Shubham Mehrotra, Xiang-Bo Mao, Sitaram Asur, et al. A comprehensive survey of llm alignment techniques: Rlhf, rlaif, ppo, dpo and more. *arXiv preprint arXiv:2407.16216*, 2024. https://arxiv.org/pdf/2407.16216.

[118] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. https://openreview.net/forum?id=yzkSU5zdwD.

[119] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022. https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

[120] Bingyang Wu, Ruidong Zhu, Zili Zhang, Peng Sun, Xuanzhe Liu, and Xin Jin. {dLoRA}: Dynamically orchestrating requests and adapters for {LoRA}{LLM} serving. In *18th USENIX Symposium on Operating Systems Design and Implementation*, 2024. https://www.usenix.org/system/files/osdi24-wu-bingyang.pdf.

[121] Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. *arXiv preprint arXiv:2407.19594*, 2024. https://arxiv.org/pdf/2407.19594.

[122] Yongji Wu, Xueshen Liu, Shuowei Jin, Ceyu Xu, Feng Qian, Z Morley Mao, Matthew Lentz, Danyang Zhuo, and Ion Stoica. Hetermoe: Efficient training of mixture-of-experts models on heterogeneous gpus. *arXiv preprint arXiv:2504.03871*, 2025. https://arxiv.org/pdf/2504.03871.

[123] Zhuofeng Wu, He Bai, Aonan Zhang, Jiatao Gu, VG Vydiswaran, Navdeep Jaitly, and Yizhe Zhang. Divide-or-conquer? which part should you distill your llm? *arXiv preprint arXiv:2402.15000*, 2024. https://arxiv.org/pdf/2402.15000.

[124] xAI. Grok prompts, 2025. https://github.com/xai-org/grok-prompts.

[125] Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. Muddformer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections. *arXiv preprint arXiv:2502.12170*, 2025. https://arxiv.org/pdf/2502.12170.

[126] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, 2023. https://arxiv.org/pdf/2211.10438.

[127] Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025. https://arxiv.org/pdf/2502.14768.

[128] Xilong Xie, Liang Wang, Limin Xiao, Meng Han, Lin Sun, Shuai Zheng, and Xiangrong Xu. Fineq: Software-hardware co-design for low-bit fine-grained mixed-precision quantization of llms. *arXiv preprint arXiv:2504.19746*, 2025. https://arxiv.org/pdf/2504.19746.

[129] Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, Caiming Xiong, et al. A minimalist approach to llm reasoning: from rejection sampling to reinforce. *arXiv preprint arXiv:2504.11343*, 2025. https://arxiv.org/pdf/2504.11343.

[130] Hongshen Xu, Zichen Zhu, Situo Zhang, Da Ma, Shuai Fan, Lu Chen, and Kai Yu. Rejection improves reliability: Training llms to refuse unknown questions using rl from knowledge feedback. *arXiv preprint arXiv:2403.18349*, 2024. https://arxiv.org/pdf/2403.18349.

[131] Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. On-device language models: A comprehensive review. *arXiv preprint arXiv:2409.00088*, 2024. https://arxiv.org/pdf/2409.00088.

[132] Xinyi Xu, Zhaoxuan Wu, Rui Qiao, Arun Verma, Yao Shu, Jingtan Wang, Xinyuan Niu, Zhenfeng He, Jiangwei Chen, Zijian Zhou, Gregory Kang Ruey Lau, Hieu Dao, Lucas Agussurja, Rachael Hwee Ling Sim, Xiaoqiang Lin, Wenyang Hu, Zhongxiang Dai, Pang Wei Koh, and Bryan Kian Hsiang Low. Position paper: Data-centric AI in the age of large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024. https://aclanthology.org/2024.findings-emnlp.695/.

[133] Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. *arXiv preprint arXiv:2502.12134*, 2025. https://arxiv.org/pdf/2502.12134.

[134] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020. https://arxiv.org/pdf/2010.11934.

[135] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. https://arxiv.org/pdf/2505.09388.

[136] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024. https://arxiv.org/pdf/2407.10671.

[137] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. https://arxiv.org/pdf/2412.15115.

[138] An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, et al. Qwen2. 5-1m technical report. *arXiv preprint arXiv:2501.15383*, 2025. https://arxiv.org/pdf/2501.15383.

[139] Hanmei Yang, Jin Zhou, Yao Fu, Xiaoqun Wang, Ramine Roane, Hui Guan, and Tongping Liu. Protrain: Efficient llm training via memory-aware techniques. *arXiv preprint arXiv:2406.08334*, 2024. https://arxiv.org/pdf/2406.08334.

[140] Junjie Ye, Yilong Wu, Sixian Li, Yuming Yang, Tao Gui, Qi Zhang, Xuanjing Huang, Peng Wang, Zhongchao Shi, Jianping Fan, et al. Tl-training: A task-feature-based framework for training large language models in tool use. *arXiv preprint arXiv:2412.15495*, 2024. https://arxiv.org/pdf/2412.15495.

[141] Haoran You, Yipin Guo, Yichao Fu, Wei Zhou, Huihong Shi, Xiaofan Zhang, Souvik Kundu, Amir Yazdanbakhsh, and Yingyan Celine Lin. Shiftaddllm: Accelerating pretrained llms via post-training multiplication-less reparameterization. In *Advances in Neural Information Processing Systems*, 2024. https://proceedings.neurips.cc/paper_files/paper/2024/file/2c30a37c75f062e0bf79297c73db8c6c-Paper-Conference.pdf.

[142] Chengye Yu, Tianyu Wang, Zili Shao, Linjie Zhu, Xu Zhou, and Song Jiang. Twinpilots: A new computing paradigm for gpu-cpu parallel llm inference. In *Proceedings of the 17th ACM International Systems and Storage Conference*, 2024. https://dl.acm.org/doi/pdf/10.1145/3688351.3689164.

[143] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2025. https://arxiv.org/pdf/2502.11089.

[144] Yichao Yuan, Lin Ma, and Nishil Talati. Moe-lens: Towards the hardware limit of high-throughput moe llm serving under resource constraints. *arXiv preprint arXiv:2504.09345*, 2025. https://arxiv.org/pdf/2504.09345.

[145] Patrick Yubeaton, Tareq Mahmoud, Shehab Naga, Pooria Taheri, Tianhua Xia, Arun George, Yasmein Khalil, Sai Qian Zhang, Siddharth Joshi, Chinmay Hegde, et al. Huff-llm: End-to-end lossless compression for efficient llm inference. *arXiv preprint arXiv:2502.00922*, 2025. https://arxiv.org/pdf/2502.00922.

[146] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, 2020. https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

[147] Fanlong Zeng, Wensheng Gan, Yongheng Wang, and Philip S. Yu. Distributed training of large language models. In *2023 IEEE 29th International Conference on Parallel and Distributed Systems*, 2023. https://www.researchgate.net/profile/Wensheng-Gan/publication/379320620_Distributed_Training_of_Large_Language_Models/links/66095106390c214cfd2c968b/Distributed-Training-of-Large-Language-Models.pdf.

[148] Haotian Zhang, Mingfei Gao, Zhe Gan, Philipp Dufter, Nina Wenzel, Forrest Huang, Dhruti Shah, Xianzhi Du, Bowen Zhang, Yanghao Li, et al. Mm1. 5: Methods, analysis & insights from multimodal llm fine-tuning. *arXiv preprint arXiv:2409.20566*, 2024. https://arxiv.org/pdf/2409.20566.

[149] Lechen Zhang, Tolga Ergen, Lajanugen Logeswaran, Moontae Lee, and David Jurgens. Sprig: Improving large language model performance by system prompt optimization. *arXiv preprint arXiv:2410.14826*, 2024. https://arxiv.org/pdf/2410.14826.

[150] Xinjie Zhang, Jintao Guo, Shanshan Zhao, Minghao Fu, Lunhao Duan, Guo-Hua Wang, Qing-Guo Chen, Zhao Xu, Weihua Luo, and Kaifu Zhang. Unified multimodal understanding and generation models: Advances, challenges, and opportunities. *arXiv preprint arXiv:2505.02567*, 2025. https://arxiv.org/pdf/2505.02567.

[151] Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. Chain of preference optimization: Improving chain-of-thought reasoning in llms. In *Advances in Neural Information Processing Systems*, 2024. https://proceedings.neurips.cc/paper_files/paper/2024/file/00d80722b756de0166523a87805dd00f-Paper-Conference.pdf.

[152] Zhehao Zhang, Ryan A Rossi, Branislav Kveton, Yijia Shao, Diyi Yang, Hamed Zamani, Franck Dernoncourt, Joe Barrow, Tong Yu, Sungchul Kim, et al. Personalization of large language models: A survey. *arXiv preprint arXiv:2411.00027*, 2024. https://arxiv.org/pdf/2411.00027.

[153] Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Huazuo Gao, Jiashi Li, Liyue Zhang, Panpan Huang, Shangyan Zhou, Shirong Ma, Wenfeng Liang, Ying He, Yuqing Wang, Yuxuan Liu, and Y. X. Wei. Insights into deepseek-v3: Scaling challenges and reflections on hardware for ai architectures. *arXiv preprint arXiv:2505.09343*, 2025. https://arxiv.org/pdf/2505.09343.

[154] Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. *arXiv preprint arXiv:2504.12328*, 2025. https://arxiv.org/pdf/2504.12328.

[155] Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, et al. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*, 2024. https://arxiv.org/pdf/2404.14294.

[156] Ásgeir Thor Johnson. System prompts leaks, 2025. https://github.com/asgeirtj/system_prompts_leaks.