ISSN: 2096-3246
Volume 58, Issue 04, 2026

TRAINING AI AGENTS: OPTIMAL PIPELINES, METHODOLOGY, AND INFRASTRUCTURE

Reeshav Kumar
Independent Researcher, USA

Abstract

The modern AI agent has outgrown the language model that gave rise to it. Where earlier systems were measured by their ability to predict the next token in a sequence, today's agents reason through multi-step problems, invoke external tools, browse live web environments, execute and debug code, and refine their plans in response to real-world feedback. These capabilities do not come from any single technique; they emerge from a coherent pipeline of interdependent training stages that must be co-designed to work effectively together. This paper surveys the AI agent training stack through five distinct layers: (1) foundation pretraining, which establishes the base model's knowledge and in-context learning capacity; (2) instruction and preference alignment, which shapes behavioral compliance through reinforcement learning from human feedback (RLHF), Constitutional AI (CAI), and direct preference optimization (DPO); (3) reasoning-oriented post-training, which amplifies structured problem-solving through chain-of-thought (CoT) prompting and reinforcement learning from verifiable rewards (RLVR), achieving results such as 79.8% on the AIME 2024 mathematics competition; (4) tool use and environment interaction, evaluated on benchmarks including SWE-bench, WebArena, and tau-bench; and (5) distributed systems infrastructure, encompassing tensor and pipeline parallelism, ZeRO optimizer sharding, and IO-aware attention kernels. For each layer, the paper examines current methods, evaluation frameworks, infrastructure constraints, and open challenges. The central finding is that agent training is a co-design problem: architecture, data, inference-time computation, and infrastructure must be optimized as an integrated system rather than in isolation.

Keywords: AI Agents, Large Language Models, Pretraining, Post-Training Alignment, Reinforcement Learning, Tool Use, Chain-of-Thought Prompting, Distributed Training, Evaluation Benchmarks, Systems Infrastructure.

1. Introduction

Three converging trajectories have defined the development of large language model (LLM)-based AI agents: scale, alignment, and interaction. The transformer architecture provided the enabling substrate — a parallelizable sequence model that trained effectively at unprecedented scale [1]. Empirical scaling laws then mapped the gains available from that scaling, identifying smooth power-law relationships between model loss and the compute, data, and parameter budgets invested in training, with exponents of approximately -0.076 for parameter count and -0.095 for training data volume [2]. Post-training techniques — supervised fine-tuning (SFT), RLHF, and their successors — transformed raw pretrained representations into systems that follow instructions and operate under behavioral constraints [3]. Where these three trajectories converge, a new kind of system emerges: not a language model in the classical sense, but an agent whose quality depends as much on its alignment policy, its tool interfaces, and its multi-step planning reliability as on the underlying model weights.

The challenge of analyzing this field is that no single layer of the agent training pipeline can be understood in isolation from the others. Pretraining corpus composition affects the alignment margins available to post-training; alignment objectives shape the reasoning trace distributions that are useful during inference; benchmark design determines which research directions receive optimization pressure; and infrastructure capabilities determine which algorithms are computationally feasible to execute at scale. A training pipeline in which each layer is independently maximized but not co-designed with the others is almost always worse than one in which the layers are jointly tuned, even if the individual layer techniques are state of the art. This co-design principle is the central organizing theme of this paper and distinguishes the analysis here from surveys that treat the layers as separable.

Four organizing questions structure the analysis. First, what techniques compose the modern agent training pipeline, and how do they interact across layers? Second, which evaluation frameworks credibly measure agent progress, and which are susceptible to contamination, saturation, or mismeasurement? Third, how do infrastructure innovations alter the frontier of what agent training algorithms are economically feasible? Fourth, what are the most pressing open challenges facing the field after the first generation of RLHF-aligned assistants and reasoning-focused models? These questions are addressed in Sections 2 through 8, culminating in a synthesis of the co-design principles that govern the most capable deployed agent systems.

The application scope that motivates these questions is broad and expanding. AI agents are now deployed across software engineering, web-based task automation, mathematical and scientific reasoning, enterprise knowledge synthesis, and multi-step decision support. Each application domain imposes distinct requirements on the training pipeline — software agents need reliable code execution and repository navigation; web agents need multi-step browser planning and error recovery; reasoning agents need deep mathematical competence and process verification. The training infrastructure that supports this breadth of capability is the subject of the analysis that follows.

2. Background and Theoretical Foundations

The transformer decoder introduced by Vaswani et al. in 2017 established the architectural substrate on which every modern AI agent is built [1]. By replacing recurrent connections with multi-head self-attention, the transformer enabled efficient parallelization across the sequence dimension during training — a property that proved decisive for scaling to the token volumes needed to learn the statistical regularities of diverse natural language. The autoregressive training objective — predicting the next token given all preceding tokens — provided a loss signal that scaled predictably with both data and compute, making the architecture and objective jointly compatible with the empirical scaling framework that Kaplan et al. characterized three years later [2].

The Chinchilla analysis refined the scaling prescription in a consequential way [5]. Kaplan et al. had identified power-law relationships between loss and both parameter count and training tokens but had not jointly optimized over both; the implicit practice in the field was to maximize parameter count within a compute budget. Hoffmann et al. showed this was a systematic error: the optimal strategy under a fixed compute budget allocates approximately 20 training tokens per parameter, a ratio that GPT-3 (175 billion parameters, approximately 300 billion training tokens) violated by roughly a factor of ten. Chinchilla's 70-billion parameter model, trained on 1.4 trillion tokens, outperformed Gopher at 280 billion parameters on most benchmarks. The lesson extended beyond the specific numbers: compute allocation across parameters and data is an empirical design variable, not an implicit default, and misjudging it has first-order consequences for model quality. LLaMA 3 operationalized this insight for the open-access tier, training an 8-billion parameter model on over 15 trillion tokens and producing performance that rivaled models with five to ten times as many parameters [30].

Richard Sutton's "bitter lesson" provides a theoretical lens through which this history becomes coherent [6]. Sutton's argument — that general methods exploiting computation outperform hand-crafted domain-specific approaches over long horizons — explains why the field has consistently converged on simple objectives (next-token prediction), general architectures (transformers), and large data volumes rather than sophisticated priors about language structure. The Chinchilla modification is that the "computation" the bitter lesson refers to must include training data as a scalable resource alongside model parameters: the lesson is not that more parameters always win, but that more effectively allocated compute — across both parameters and tokens — consistently does.

3. Training Pipeline Taxonomy

AI agent training is best understood as a five-layer pipeline in which each layer adds capabilities that prior layers cannot provide. The layers are not strictly sequential in practice — post-training techniques are applied iteratively, infrastructure choices constrain algorithms at every stage, and evaluation benchmarks span multiple layers simultaneously — but the analytical distinction between them clarifies both the strengths and the failure modes of each technique family. Table 1 summarizes the five layers with representative scale results.

Table 1. Five-Layer AI Agent Training Pipeline: Techniques, Objectives, and Representative Results

Layer	Primary Technique(s)	Core Objective	Representative Scale / Result
1. Foundation Pretraining	Autoregressive LM on diverse corpora; dense / sparse MoE	Next-token prediction; broad knowledge and in-context learning	LLaMA 3: 8B params on 15T tokens; Chinchilla: ~20 tokens/param optimal
2. Instruction & Preference Align.	SFT, RLHF, Constitutional AI (CAI), DPO	Follow instructions; comply with behavioral policy	InstructGPT 1.3B preferred over 175B raw GPT-3 (~70% of comparisons)
3. Reasoning Post-Training	CoT prompting, GRPO, RLVR, self-consistency	Reliable multi-step problem solving under process constraints	DeepSeek-R1: 79.8% AIME 2024; 97.3% MATH-500 via RL alone
4. Tool Use & Environment	Toolformer, ReAct, function calling, browser interaction	Act on external state; recover from errors; complete tasks	SWE-bench top agents: >40% issue resolution on real GitHub repos
5. Systems Infrastructure	Tensor/pipeline/data parallel; ZeRO; FlashAttention	Economic feasibility of large-scale training	ZeRO-3: 8x memory reduction; FlashAttention-2: 2–4x attention speedup

Foundation pretraining establishes the base model's knowledge, language competence, and in-context learning capability through next-token prediction at scale. The architectural choices at this layer — dense versus sparse mixture-of-experts (MoE), context window length, tokenizer vocabulary — determine the representational capacity available to subsequent layers. Critically, pretraining corpus composition affects the alignment margins available post-training: models pretrained on data with higher proportions of instruction-style text, code, and structured reasoning tend to respond more effectively to post-training alignment techniques [5]. The second layer, instruction and preference

alignment, reshapes model behavior after pretraining to follow operator policies and user instructions through SFT, RLHF, CAI, or DPO [3], [7], [8], [9]. The third layer, reasoning post-training, addresses the gap between an aligned model and one that reliably solves multi-step problems, using CoT prompting, Group Relative Policy Optimization (GRPO), and RLVR to train models that generate verifiably correct reasoning traces [10], [11], [12].

The fourth layer — tool use and environment interaction — is the transition point at which a language model becomes an agent. Without external tool access, a model can only describe and predict; with tool access, it can retrieve, compute, modify, and execute. Toolformer demonstrated that tool-use behavior can be learned from small amounts of supervision by training models to call application programming interfaces (APIs) at positions where the API output reduces prediction loss [13]. ReAct showed that explicitly interleaving reasoning traces with grounded actions in a "reason-then-act" loop outperformed both reasoning-only and action-only baselines on knowledge-intensive tasks [14]. The fifth layer, systems infrastructure, governs the computational envelope for all other layers. Megatron-LM, ZeRO optimizer sharding, and FlashAttention are not implementation conveniences; they are what makes the training algorithms at layers one through four economically feasible at the scales that produce capable agents [15], [16], [17].

The co-design interactions between layers are significant. Infrastructure choices constrain which algorithms are feasible: clusters without efficient sequence parallelism cannot train on the long reasoning traces that layer three requires; serving systems without memory-efficient attention cannot deliver throughput adequate for reasoning-heavy inference at production volume. Alignment choices at layer two affect the quality of rollouts generated during layer-three reinforcement learning, because the policy being optimized by the reinforcement learning algorithm is initialized from the layer-two aligned model. Benchmark choices at the evaluation stage determine which failure modes receive optimization attention — if the evaluation suite does not include long-horizon agentic tasks, post-training improvements in multi-step reliability will be undervalued. These interactions are why the co-design framing is necessary and why single-layer optimization analyses often fail to predict deployed system behavior.

4. Data Curation and Corpus Design

The quality and composition of the pretraining corpus is as consequential for downstream agent capability as the choice of training algorithm. The Pile established source diversity — combining 22 subsets spanning books, scientific papers, code, web text, and domain-specific corpora — as a first-class design principle for pretraining corpora, demonstrating that diverse multi-source data produced better downstream performance than any single high-quality source used alone [18]. The diversity principle reflects the broader aims of agent training: an agent that is strong in language but weak in code, or strong in factual recall but weak in mathematical reasoning, is a specialized tool rather than a general-purpose agent. Diversity in pretraining is the mechanism by which general competence is established.

Deduplication emerged as an equally important design variable. Lee et al. demonstrated that repeated and near-duplicate content in pretraining corpora increases memorization of training examples while reducing generalization to held-out evaluation data [19]. Models trained on deduplicated corpora of equivalent token count consistently outperformed models trained on undeduplicated corpora on downstream evaluation benchmarks, particularly those requiring compositional generalization rather than pattern matching. Dolma operationalized this insight at the multi-trillion-token scale, providing an open and fully documented curation pipeline with explicit records of filtering heuristics, source mixing ratios, and quality scoring methods — enabling the scientific community to reproduce and audit corpus design decisions rather than treating them as proprietary variables [20].

FineWeb and FineWeb-Edu produced the most striking quantitative demonstration of quality-over-scale in recent corpus research. Penedo et al. showed that FineWeb-Edu — approximately 1.3 trillion tokens of web content scored and filtered for educational quality — achieved downstream benchmark performance that rivaled or exceeded models trained on corpora ten times larger in raw token count on several reasoning tasks [21]. This result challenged a persistent assumption: that raw scale dominates careful curation. The implication for agent training economics is significant: organizations that invest in principled corpus design may be able to achieve frontier-competitive pretraining at a fraction of the compute cost of brute-force scaling. The legal constraints analyzed by Longpre et al. add urgency to this finding — as the proportion of permissively licensed web content available for training decreases, quality-efficient corpus design becomes not merely advantageous but necessary [22]. Table 2 summarizes representative open pretraining corpora.

Table 2. Representative Open Pretraining Corpora: Design Choices and Limitations

Corpus	Scale	Key Design Choice	Downstream Strength	Limitation
The Pile	825B tokens	22-subset diversity across domains	Broad coverage including code and scientific text	No deduplication; some low-quality subsets
C4	~750B tokens	Common Crawl + heuristic quality filtering	Clean general English text at scale	English-only; over-filtering removes useful content
Dolma	3T tokens	Open multi-source pipeline with full docs	Reproducible; academically auditable	Web-heavy; limited non-English coverage
FineWeb	15T tokens	Quality filtering + large-scale deduplication	Strong downstream reasoning benchmarks	English-centric; biased toward web prose
FineWeb-Edu	1.3T tokens	Educational content scoring on filtered web	Outperforms 10x larger corpora on reasoning tasks	Narrow domain; may underfit general conversation
RedPajama-v2	30T tokens	Largest openly documented web-scale corpus	Raw scale available for filtering experiments	Minimal quality filtering in base form

ISSN: 2096-3246
Volume 58, Issue 04, 2026

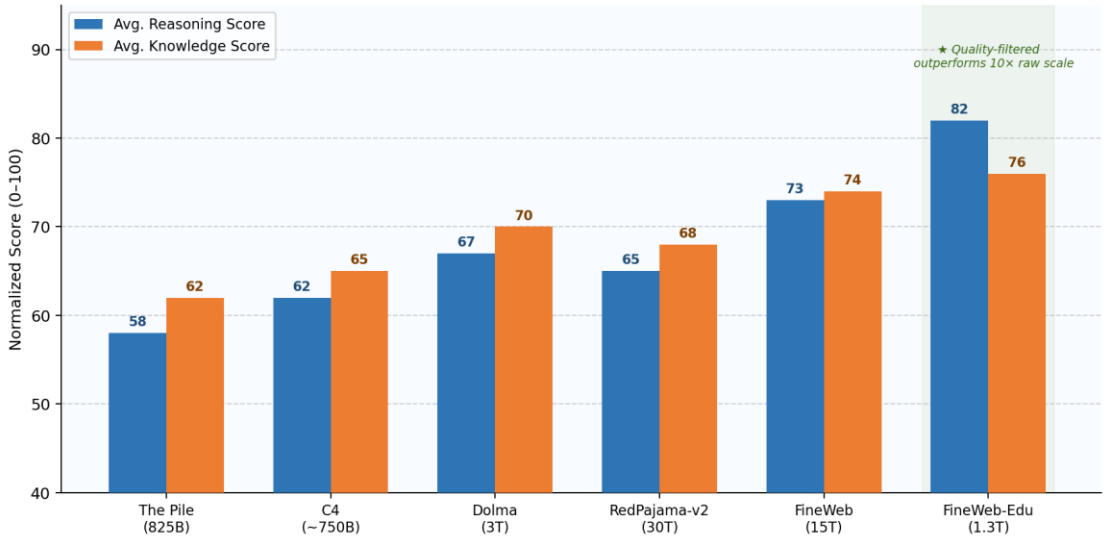

Fig. 1. Downstream benchmark performance (normalized) vs. corpus design across six open pretraining corpora.

5. Post-Training Alignment and Reasoning Optimization

InstructGPT's central finding was not merely that alignment improved model outputs — it was that alignment could substitute for raw scale to a degree the field had not anticipated [3]. A 1.3-billion parameter model, fine-tuned with RLHF on a relatively small set of human preference comparisons, was preferred by human evaluators over the raw 175-billion parameter GPT-3 on approximately 70% of prompts drawn from the target distribution. This 134x parameter advantage for the larger model was insufficient to overcome the behavioral advantage of the smaller aligned one. The practical implication is that for any organization deploying LLMs in production, the question is not "what is the largest model we can afford?" but "what alignment and adaptation pipeline closes the gap between our base model and user requirements?"

The alignment toolkit has evolved to reduce both cost and dependence on human annotation. CAI addressed RLHF's scalability constraint — the requirement for large volumes of costly human preference labels — by using model-generated critiques and revisions as the primary training signal, guided by a stated set of constitutional principles [7]. Empirically, CAI-trained models rated as more harmless than RLHF-trained models by human evaluators while maintaining helpfulness, demonstrating that AI-generated feedback could substitute for human feedback without sacrificing alignment quality. DPO simplified the pipeline further by reformulating preference alignment as a supervised objective over pairwise comparisons — eliminating the explicit reward model and proximal policy optimization (PPO) reinforcement learning stage entirely [9]. At scale, DPO achieves alignment quality comparable to full RLHF at substantially lower computational and engineering cost, making it the default alignment technique in most open-source fine-tuning pipelines.

Reasoning-oriented post-training represents the most consequential recent development in agent capability. Wei et al. demonstrated that CoT prompting — structuring the model to generate intermediate reasoning steps before producing a final answer — improved accuracy on three-digit addition from 18.8% to 98.8% at sufficient model scale, and produced systematic gains across arithmetic, symbolic, and commonsense reasoning task categories [10]. GRPO, introduced by Shao et al. in DeepSeekMath, improved over standard PPO for mathematical reasoning by replacing the value model with group-relative advantage estimation, reducing memory requirements substantially while maintaining policy gradient stability [11]. DeepSeek-R1 took the principle to its logical extreme: training an agent to achieve 79.8% on AIME 2024 and 97.3% on the MATH-500 benchmark through

RLVR alone, without any supervised CoT examples, demonstrating that structured reasoning emerges from verifiable reward signals without requiring human demonstration of reasoning traces [12]. This finding reshapes the understanding of what post-training can achieve — structured reasoning is not just a prompting trick but a trainable capability.

6. Tool Use, Benchmarking, and Evaluation Methodology

Tool use is the mechanism by which a language model becomes an agent in the operational sense. Without tool access, a model can describe the world within its parametric knowledge; with tool access, it can retrieve current information, execute computations, modify files, and interact with external systems. Toolformer established that tool-use behavior could be learned from minimal supervision by training models to insert API calls at sequence positions where the API output would reduce next-token prediction loss — a self-supervised signal requiring only a small number of human-authored demonstrations per tool [13]. The resulting model learned when to call tools and how to integrate their outputs into coherent responses, without requiring explicit supervision of the call/response integration pattern. ReAct demonstrated that the quality of tool use improved substantially when the model was trained to reason explicitly about the current state of the environment before selecting an action — the "reason-then-act" interleaving pattern that is now standard in production agent frameworks [14]. WebGPT extended this to the full web browsing setting, training a model to navigate real websites, issue search queries, follow links, and synthesize retrieved information into cited responses to open-domain questions [23].

The benchmark landscape for AI agents has expanded beyond the static question-answering evaluations that dominated LLM assessment through 2022, and this expansion reflects the field's growing recognition that agent capability requires interactive, multi-step evaluation. SWE-bench evaluates real software engineering capability by testing whether models can resolve actual GitHub issues in full repository contexts, requiring file navigation, root-cause analysis, multi-file editing, and test passing [24]. As of 2024, the best agentic systems achieved above 40% resolution rates on SWE-bench's verified split — a substantial improvement from near-zero for standard generation models just two years earlier. WebArena evaluates browser-based decision-making across four realistic web environments — shopping, content management, discussion forums, and collaborative development — testing multi-step completion rates on tasks that mimic real web application use [25]. tau-bench extends the evaluation to stateful tool-use reliability, measuring whether an agent passes the same task across multiple independent trials rather than succeeding once by chance, with a pass^k metric that better reflects production reliability requirements [26]. Table 3 summarizes the benchmark landscape.

Table 3. Representative Evaluation Benchmarks and Agent Environments

Benchmark	Type	What It Measures	Reported Frontier Score (2024–25)	Key Limitation
MMLU	Static QA	Academic knowledge — 57 subjects	~90%+ for GPT-4 class models	Saturation; contamination risk
GPQA	Static QA	Expert-level science (graduate difficulty)	~53% best models (vs 65% expert human)	Small dataset; narrow coverage

Benchmark	Type	What It Measures	Reported Frontier Score (2024–25)	Key Limitation
GSM8K / MATH	Reasoning	School / competition mathematics	DeepSeek-R1: 97.3% (MATH-500)	Contamination; not open-ended
HumanEval	Code synthesis	Function-level code generation (pass@k)	~90%+ for top models	Single-file; no repository context
SWE-bench	Code agent	Real GitHub issue resolution in full repos	>40% for best agentic systems (2024)	Slow eval; environment complexity
WebArena	Web agent	Multi-step browser task completion	~35–45% for leading agents	Environment fragility; limited task diversity
tau-bench	Tool agent	Stateful tool reliability across repeated trials	~50–60% pass^k for top agents	Domain-specific (retail / airline)
GAIA	Generalist	Real-world tasks: web + tools + multimodal reasoning	~50–55% for top systems vs 92% human	Expensive human annotation required
BFCL	Function calling	API schema fidelity across 2,000 tasks	~85–90% for frontier models	Synthetic tasks; limited real-world transfer
LiveBench	Dynamic	Monthly-refreshed — contamination-resistant	Actively maintained frontier separation	Evaluation cost; narrow monthly scope

Three evaluation methodology lessons emerge from this landscape. First, static benchmarks saturate and become contaminated as training corpora expand: when benchmark questions appear in pretraining data, reported accuracy overestimates genuine capability. LiveBench addresses this by refreshing questions monthly, sourcing them from events recent enough to be unlikely to appear in training data, and maintaining a continuously updated frontier separation [28]. Second, agentic capability cannot be captured by static question-answering alone. Interactive environments that expose the agent to multi-step decision-making — where early errors compound through later steps — reveal failure modes that static evaluation conceals. A 90% per-step success rate over a ten-step task yields only a 35% end-to-end success rate, which means evaluation protocols that measure individual-step accuracy dramatically overestimate task completion capability. Third, no single benchmark family can simultaneously assess knowledge, reasoning, tool use, and long-horizon interaction reliability. MMLU, SWE-bench, WebArena, and tau-bench measure qualitatively different aspects of agent capability and must be administered in combination to produce a credible capability profile.

ISSN: 2096-3246
Volume 58, Issue 04, 2026

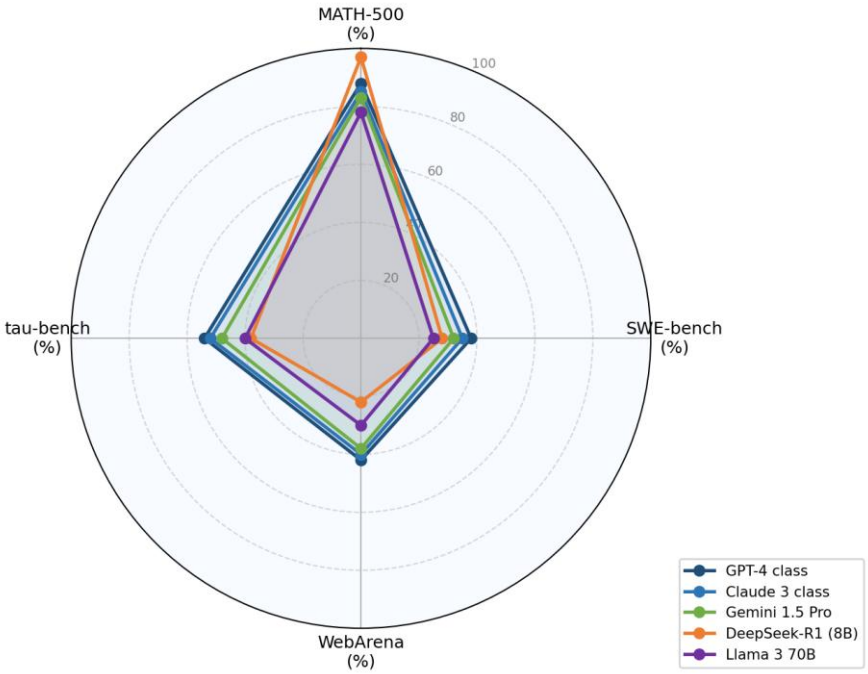

Fig. 2. Agent capability radar across four benchmark dimensions for five leading model families.

7. Distributed Systems Infrastructure

The distributed systems infrastructure that supports large-scale agent training is not background to the algorithmic story — it determines the boundary of what that story can include. Megatron-LM established the foundation by demonstrating that tensor parallelism, pipeline parallelism, and data parallelism could be composed to train models with hundreds of billions of parameters across thousands of GPUs, achieving near-linear scaling for attention computation and acceptable parallel efficiency for the full training pipeline [15]. The algorithms that the literature presents as advances in agent capability — RLHF, CAI, GRPO, RLVR — are only feasible in their full form because the infrastructure to execute large-scale gradient updates efficiently at these model sizes exists. Without Megatron-LM-class infrastructure, the largest experiments in the alignment and reasoning literature would not be computationally tractable.

DeepSpeed's ZeRO optimizer family addressed the memory bottleneck that previously prevented large-scale training on cost-accessible GPU clusters [16]. Standard distributed data parallelism (DDP) stores a complete copy of the optimizer states, gradients, and model parameters on each worker, introducing enormous memory redundancy. ZeRO-1 shards optimizer states; ZeRO-2 adds gradient sharding; ZeRO-3 shards all three, achieving an 8x reduction in per-GPU memory relative to DDP at the 175-billion parameter scale. ZeRO-Infinity extends this to NVMe offloading, making trillion-parameter training feasible on GPU clusters without NVMe-resident models. These memory reductions are what enable post-training experiments on large models — RLHF rollout generation, GRPO policy gradient updates, DPO optimization — in research settings without access to the largest-scale GPU clusters.

FlashAttention addressed the attention computation bottleneck at the kernel level [17]. Standard attention implementations materialize the full N×N attention matrix in high-bandwidth memory (HBM), creating a memory bandwidth bottleneck that grows quadratically with sequence length. FlashAttention fuses the attention computation into a single GPU kernel that keeps intermediate results in on-chip SRAM, avoiding the HBM round-trips entirely and reducing memory traffic by 2–4x. End-to-end training speedups of 15–20% for BERT and 3x for GPT-2 follow from this reduction.

FlashAttention-2 improved further through better parallelization over sequence and batch dimensions and tighter thread scheduling, approaching theoretical peak floating-point operations utilization on H100 hardware [29]. These gains are most significant for agent training, where long reasoning traces and multi-turn context windows are the norm — making IO-aware kernel design a first-order concern rather than a secondary optimization.

Table 4 summarizes infrastructure techniques relevant to agent training with reported efficiency gains. The cumulative effect of Megatron-LM, ZeRO, and FlashAttention together is that the practical frontier of model scale and training algorithm complexity has expanded by multiple orders of magnitude over the past five years — not because the underlying algorithmic ideas changed, but because the infrastructure enabling their efficient execution was built. This expansion is the mechanism by which the reasoning and tool-use capabilities described in Sections 5 and 6 became computationally accessible.

Table 4. Infrastructure Techniques: System Layer, Primary Benefit, and Reported Efficiency Gains

Technique	System Layer	Primary Benefit	Reported Efficiency Gain
Tensor parallelism (Megatron-LM)	Layer-level model sharding	Enables large hidden dimensions per GPU	Near-linear scaling to 8 GPUs for attention layers
Pipeline parallelism (Megatron-LM)	Stage-level model sharding	Supports very deep models across node boundaries	Scales to 1T+ parameters across GPU clusters
ZeRO-3 (DeepSpeed)	Full state sharding across workers	Eliminates redundant optimizer, gradient, param storage	8x per-GPU memory reduction vs. DDP baseline
ZeRO-Infinity	NVMe offloading	Exceeds GPU memory limits via NVMe-backed params	Enables trillion-param training on modest clusters
FlashAttention (Dao et al.)	Attention kernel — IO-aware	Reduces HBM memory traffic; avoids full N×N matrix	2–4x attention speedup; 15–20% E2E training speedup
FlashAttention-2 (Dao)	Improved thread scheduling	Better parallelism over sequence and batch dims	Near-theoretical peak FLOP utilization on H100
Selective activation recompute	Backpropagation memory	Trades ~20% compute for ~35% activation memory	~30–40% memory reduction at modest compute overhead
Sequence parallelism	Long-sequence training	Distributes sequence dimension across GPUs	Enables >128K context-length training at scale

8. Applications and Open Challenges

Software engineering represents the most mature deployment domain for AI agents. HumanEval measures function-level code synthesis with a pass@k metric, but SWE-bench better represents actual software engineering practice: resolving real GitHub issues requires reading multi-file codebases, identifying root causes from bug reports, editing the relevant files, and producing patches that pass the repository's existing test suite [24]. This task profile — multi-file navigation, causal reasoning about code behavior, constrained editing — is qualitatively more demanding than function synthesis from a docstring. The improvement of agentic systems on SWE-bench from near-zero resolution rates in 2023 to above 40% in 2024 reflects the compound effect of better base models, improved tool-use scaffolding, and more targeted evaluation-driven development. Web-based task automation — evaluated on WebArena and GAIA — represents the second major deployment domain, testing whether agents can complete multi-step browser tasks in realistic, dynamic environments rather than static retrieval settings [25], [27]. The reliability challenge here is compounded by environmental noise: web interfaces change, server responses vary, and error recovery is required across steps in a way that simple generation tasks do not expose.

Five open challenges define the most consequential research directions. Data governance under emerging legal constraints is the first: the regime of unrestricted web-scale scraping that enabled first-generation LLM pretraining is becoming legally unstable, with the proportion of permissively licensed content declining and litigation over training data use increasing [22]. Future pretraining pipelines will require licensing documentation, consent tracking, and synthetic data augmentation at scales that current practices do not support. Evaluation validity is the second challenge: as static benchmark scores approach ceiling levels and training corpora increasingly cover benchmark question distributions, the field requires dynamic, contamination-resistant evaluation frameworks that measure genuine generalization. Process reliability in multi-step agent loops is the third challenge: the compound failure rate of long-horizon tasks means that process supervision, step verification, and explicit error-recovery training are necessary for production-grade agents — and these capabilities are substantially undersolved relative to single-step task performance [26]. Communication and memory bottlenecks in distributed training represent the fourth challenge, as context lengths and model scales continue to increase beyond what current interconnect and memory hierarchy designs efficiently support. Reproducibility is the fifth: despite improving transparency in some research groups, the field still lacks standardized documentation of filtering heuristics, post-training recipe details, and evaluation protocols that would allow independent scientific replication of claimed agent capability results [20].

The intersection of these challenges points toward a common research agenda: evaluation frameworks that remain valid as models improve, training pipelines that are economically viable under tighter data licensing constraints, agent architectures that are reliable over long horizons rather than just capable on individual steps, and infrastructure designs that scale efficiently as the demands of reasoning and tool-use training continue to grow. Progress on any one of these challenges in isolation will be limited; the co-design framing that characterizes the best current agent systems applies equally to the open problems that remain.

9. Conclusion

The modern AI agent is the product of a co-designed training stack, not any single algorithmic breakthrough. Foundation pretraining on compute-optimally curated corpora — approximately 20 training tokens per parameter, as Chinchilla established — provides the base competence that every subsequent layer refines [5]. Instruction and preference alignment through RLHF, CAI, and DPO transforms pretrained representations into policy-compliant systems, with InstructGPT demonstrating that a 1.3-billion parameter aligned model outperforms a 175-billion parameter unaligned one on user-

facing tasks [3]. Reasoning post-training — culminating in RLVR-trained models like DeepSeek-R1 achieving 79.8% on AIME 2024 through reinforcement learning alone — demonstrates that structured problem-solving capability can be substantially amplified beyond the pretrained baseline [12]. Tool use and environment interaction, evaluated on benchmarks including SWE-bench (>40% issue resolution for top agents) and WebArena, converts the language model into an operational agent that acts on and receives feedback from the real world [13], [14], [24]. Distributed infrastructure — Megatron-LM for model parallelism, ZeRO for memory efficiency, FlashAttention-2 for IO-aware attention — determines the computational envelope within which all of these layers operate [15], [16], [17], [29].

The co-design principle articulated throughout this paper has a direct implication for intelligent systems engineering practice. Organizations building production AI agents that optimize each pipeline layer independently — choosing the largest available base model, applying standard alignment, prompting for reasoning, and deploying on default infrastructure — will consistently underperform teams that treat the pipeline as an integrated system. The gains available from co-design are empirically demonstrated: Chinchilla showed that reallocating compute from parameters to tokens improved quality at equivalent cost; FineWeb-Edu showed that quality-filtered corpora outperformed ten times as many raw web tokens; InstructGPT showed that alignment narrowed a 134x parameter gap; DeepSeek-R1 showed that reasoning RL produced math competition results from an 8-billion parameter model. Each of these results is a case study in the value of treating the training stack as a design problem rather than a selection problem.

The open challenges that remain — data governance under legal constraints, contamination-resistant evaluation, process reliability in long-horizon agent loops, infrastructure scaling for extended context and reasoning traces — are each tractable individually but are most efficiently addressed as a system. The field's most important methodological contribution going forward may not be any single new training technique but the development of principled frameworks for co-designing evaluation, data curation, training algorithms, and infrastructure together, rather than advancing each in isolation. The analysis in this paper is intended as a step toward that more integrated understanding of what makes AI agent training work.

References

A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1706.03762

1. J. Kaplan et al., "Scaling laws for neural language models," arXiv preprint arXiv:2001.08361, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2001.08361
2. L. Ouyang et al., "Training language models to follow instructions with human feedback," in Advances in Neural Information Processing Systems, vol. 35, pp. 27730–27744, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2203.02155
3. T. Brown et al., "Language models are few-shot learners," in Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2005.14165
4. J. Hoffmann et al., "Training compute-optimal large language models," arXiv preprint arXiv:2203.15556, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2203.15556
5. R. Sutton, "The bitter lesson," Online essay, 2019. [Online]. Available: http://www.incompleteideas.net/IncIdeas/BitterLesson.html
6. Y. Bai et al., "Constitutional AI: Harmlessness from AI feedback," arXiv preprint arXiv:2212.08073, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2212.08073

7. H. W. Chung et al., "Scaling instruction-finetuned language models," Journal of Machine Learning Research, vol. 25, no. 70, pp. 1–53, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2210.11416
8. R. Rafailov et al., "Direct preference optimization: Your language model is secretly a reward model," in Advances in Neural Information Processing Systems, vol. 36, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.18290
9. J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in Advances in Neural Information Processing Systems, vol. 35, pp. 24824–24837, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2201.11903
10. Z. Shao et al., "DeepSeekMath: Pushing the limits of mathematical reasoning in open language models," arXiv preprint arXiv:2402.03300, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.03300
11. D. Guo et al., "DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning," arXiv preprint arXiv:2501.12948, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2501.12948
12. T. Schick et al., "Toolformer: Language models can teach themselves to use tools," in Advances in Neural Information Processing Systems, vol. 36, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.04761
13. S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," in International Conference on Learning Representations, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2210.03629
14. M. Shoeybi et al., "Megatron-LM: Training multi-billion parameter language models using model parallelism," arXiv preprint arXiv:1909.08053, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1909.08053
15. S. Rajbhandari et al., "ZeRO: Memory optimizations toward training trillion parameter models," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.1910.02054
16. T. Dao et al., "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," in Advances in Neural Information Processing Systems, vol. 35, pp. 16344–16359, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2205.14135
17. L. Gao et al., "The Pile: An 800GB dataset of diverse text for language modeling," arXiv preprint arXiv:2101.00027, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2101.00027
18. K. Lee et al., "Deduplicating training data makes language models better," in Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 8424–8445, 2022. [Online]. Available: https://doi.org/10.18653/v1/2022.acl-long.577
19. L. Soldaini et al., "Dolma: An open corpus of three trillion tokens for language model pretraining research," arXiv preprint arXiv:2402.00159, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.00159
20. G. Penedo et al., "The FineWeb datasets: Decanting the web for the finest text data at scale," arXiv preprint arXiv:2406.17557, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2406.17557
21. S. Longpre et al., "The data provenance initiative: A large scale audit of dataset licensing and attribution in AI," arXiv preprint arXiv:2310.16787, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2310.16787
22. R. Nakano et al., "WebGPT: Browser-assisted question-answering with human feedback," arXiv preprint arXiv:2112.09332, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2112.09332

B. Jimenez et al., "SWE-bench: Can language models resolve real-world GitHub issues?" in International Conference on Learning Representations, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2310.06770
23. S. Zhou et al., "WebArena: A realistic web environment for building autonomous agents," in International Conference on Learning Representations, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2307.13854
24. S. Yao et al., "tau-bench: A benchmark for tool-agent-user interaction in real-world domains," arXiv preprint arXiv:2406.12045, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2406.12045
25. G. Mialon et al., "GAIA: A benchmark for general AI assistants," in International Conference on Learning Representations, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2311.12983
C. White et al., "LiveBench: A challenging, contamination-free LLM benchmark," arXiv preprint arXiv:2406.19314, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2406.19314
26. T. Dao, "FlashAttention-2: Faster attention with better parallelism and work partitioning," in International Conference on Learning Representations, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2307.08691
27. AI@Meta, "The Llama 3 herd of models," arXiv preprint arXiv:2407.21783, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2407.21783