# AUTO-RESEARCH ARENA: BENCHMARKING AND ADVANCING AUTO-RESEARCH IN LANGUAGE MODELING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

As autonomous research systems become more capable, benchmarks must both evaluate their performance and support meaningful research progress. We introduce Auto-Research Arena, a multi-target benchmark for evaluating autonomous research systems and discovering ideas in language modeling. Arena supports diverse research objectives within a shared experimental environment by separating *admissibility* from *improvement*: model quality and resource constraints determine whether a result qualifies, while the selected target measures progress among admissible results. We instantiate Arena in language model pretraining with nine targets spanning model, data, training, and inference efficiency, and evaluate seven research harnesses using two foundation models. Arena clearly differentiates existing harnesses across research objectives, with several sophisticated methods outperformed by our simple sequential search baseline, indicating substantial headroom in current auto-research systems. Beyond evaluation, Arena reveals a substantially broader frontier of model and system ideas aligned with active research directions. Reusing these discoveries in a new quality-driven search yields lower training loss under matched data and FLOPs and lower held-out BPB at the same model size. Together, Arena provides a common framework for benchmarking autonomous research and expanding its scope in language modeling.

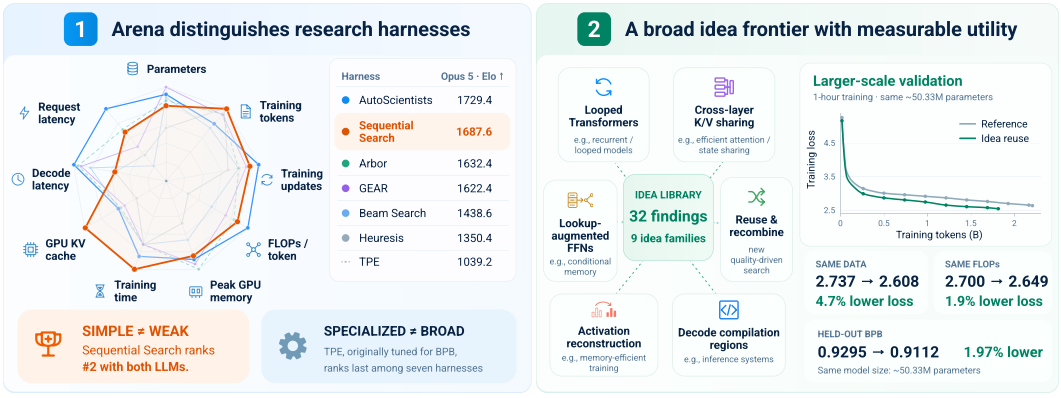



Figure 1: **Auto-Research Arena evaluates research systems and expands the idea frontier. Left:** Arena differentiates research harnesses across nine complementary objectives. Our simple Sequential Search baseline remains competitive, whereas the TPE method tailored to validation-loss minimization underperforms in this broader evaluation. **Right:** Multi-target research reveals a broad frontier of ideas. A program combining these discoveries achieves lower training loss than the reference program under matched data and FLOP budgets, as well as lower final held-out BPB at the same model size.

## 1 INTRODUCTION

Advances in LLMs and research harnesses have enabled increasingly capable autonomous research systems (Tang et al., 2025; Liu et al., 2026). These systems iteratively propose ideas, conduct experiments, and use the results to guide further research. Auto-research benchmarks provide a common basis for comparing these systems, but their role extends beyond evaluation. By defining clear objectives and constraints, a well-designed benchmark helps researchers identify, validate, and build on useful improvements. However, existing benchmarks remain limited both in evaluating research



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

systems and in supporting broad research ideas (Chan et al., 2025; Wijk et al., 2025; Nathani et al., 2025; Karpathy, 2026; Intology, 2026).

To address this gap, we introduce **Auto-Research Arena**, a comprehensive benchmark for evaluating auto-research systems and advancing idea discovery in language modeling. Departing from prior benchmarks that define progress primarily through a particular objective within each research problem, **Arena defines multiple research objectives within a common experimental environment**. These objectives address complementary bottlenecks across the model lifecycle, including model size, data consumption, training computation and memory, and inference memory and latency. The shared environment provides a common experimental basis for evaluating whether a harness remains effective across different research objectives. Meanwhile, different objectives reward different types of improvements. For example, reducing model size or inference memory can count as progress even when prediction loss remains unchanged. By recognizing these complementary forms of progress, Arena supports exploration of a broader range of model designs and systems optimizations.

To support this broad range of research objectives within a unified framework, Arena specifies both what to improve and which requirements a candidate must satisfy. That is, Arena separates the standard for *admissibility* from the direction of *improvement*. Each objective is formulated as a constrained optimization problem. Constraints on model quality and relevant non-target resources determine whether a candidate is *admissible*. Among admissible candidates, a reduction in the selected resource metric constitutes *improvement*. Changing the target metric defines a different research objective within the same framework. This formulation supports consistent evaluation across diverse objectives, while excluding gains that violate the specified quality or resource constraints. Beyond evaluating objectives independently, Arena supports research across objectives. Separate searches can pursue different objectives in parallel, each exploring improvements relevant to its target. Compatible improvements discovered in one search can then be incorporated into the starting program for a subsequent search targeting a different objective. This enables both *parallel exploration* of distinct research directions and *cumulative progress* through the reuse of earlier discoveries. By explicitly specifying objectives, constraints, budgets, and starting programs, Arena makes it possible to measure how prior discoveries affect subsequent research.

We evaluate six LLM-based auto-research methods and a Tree-structured Parzen Estimator baseline (Ferreira et al., 2026) across Arena's nine objectives. **Arena provides a discriminative and meaningful evaluation of research harnesses**: methods exhibit clear differences across objectives, with no single harness dominating the benchmark. Two results further support the evaluation. TPE, specialized for the original validation loss-oriented setting, performs worst in Arena's broader multi-target evaluation. Moreover, our simple Sequential Search baseline ranks second overall with both foundation models, outperforming several substantially more sophisticated harnesses and suggesting large headroom in current auto-research system design. The recorded search trajectories help explain how these final outcomes arise. Higher-performing harnesses tend to sustain progress throughout the search budget, whereas lower-performing methods often plateau earlier or improve less consistently.

**Multi-target research also exposes a substantially broader idea frontier**. We empirically find that different objectives lead to qualitatively different discoveries, yielding 32 findings across nine idea families spanning model architecture, optimization, memory, and inference systems. Many of these discoveries align with active directions in LLM research. For example, when optimizing parameter efficiency, agents discover looped Transformers through weight sharing. They further learn depth-specific residual gains to adapt the residual updates across repeated applications of the shared block. This finding closely parallels recent work identifying residual adaptation as a key ingredient of looped Transformers (Wang et al., 2026). Similar connections appear across other discoveries, although alignment with existing research does not imply identical solutions. Several discoveries follow design principles similar to those of prior work but implement them through distinct mechanisms, offering alternative approaches to the same research problems. Together, these findings show how Arena's diverse objectives expand the range of ideas explored by autonomous research systems.

Finally, we examine whether ideas discovered through Arena's lightweight searches remain useful beyond the search setting. The original searches train each candidate for only 10 minutes. We provide the collected discoveries to a new search focused on model quality, which can select, combine, and modify these ideas. We then train the resulting model for one hour from random initialization. Unlike the original searches, which optimize individual efficiency targets, this experiment evaluates improvements in model quality directly. Relative to the reference run, training loss decreases from 2.737 to 2.608 at matched training data and from 2.700 to 2.649 at matched cumulative FLOPs. Held-

out BPB also decreases from 0.9295 to 0.9112 at essentially the same model size. These results show that discoveries from Arena's small-scale, multi-target searches can improve model quality through combination and adaptation. Arena thus supports both the evaluation of autonomous research systems and the discovery of reusable research ideas.

## 2 Related Work

**Autonomous research benchmarks.** PaperBench (Starace et al., 2025) evaluates replication of published research, while EXP-Bench (Kon et al., 2026) assesses the design and execution of paper-derived experiments. These benchmarks emphasize implementing and validating specified research content. More open-ended settings include MLE-bench (Chan et al., 2025) and RE-Bench (Wijk et al., 2025), which evaluate ML engineering and research, alongside MLGym (Nathani et al., 2025), MLRC-Bench (Zhang et al., 2025c), and AstaBench (Bragg et al., 2026), which support standardized experimentation and evaluation across ML and scientific tasks. Within language modeling, Karpathy's autoresearch (Karpathy, 2026) minimizes validation loss under a fixed training-time budget, whereas NanoGPT Bench (Intology, 2026) targets faster training at a required model quality. Related benchmarks evaluate reproduction of NanoGPT improvements (Zhao et al., 2025) and correctness-constrained computational speedups (Ouyang et al., 2025; Press et al., 2025; Shetty et al., 2025). However, existing benchmarks provide limited evaluation of auto-research systems across complementary objectives. Arena instead unifies diverse objectives within a shared experimental setting, revealing how the research objective shapes both harness performance and the resulting idea frontier.

**Autonomous research systems.** Complementing these benchmarks, research harnesses determine how agents organize experiments and use accumulated evidence. End-to-end systems connect ideation, experimentation, and reporting (Lu et al., 2026; Tang et al., 2025), while AIDE (Jiang et al., 2025) and AIRA (Toledo et al., 2025) organize iterative code improvement through search. AutoScientists (Gao et al., 2026) coordinates decentralized teams, Arbor (Jin et al., 2026) maintains a persistent hypothesis tree, and GEAR (Jeddi et al., 2026) evolves populations of research states. Heuresis (Antoniades et al., 2026) further compares search strategies across quality, diversity, and novelty. Arena evaluates how the relative strengths of research harnesses vary across objectives.

## 3 Auto-Research Arena

Auto-Research Arena defines a family of research problems that optimize different resource targets within a shared language-modeling environment. Fig. 2 illustrates the research loop, from selecting a target and evaluating candidate programs to identifying each harness's best admissible program and comparing performance across targets. In the following, we present the general formulation in Sec. 3.1 and its language-modeling instantiation in Sec. 3.2.

### 3.1 Auto-Research as Constrained Optimization

Consider a testbed with a fixed data pool, evaluator, and execution environment. An auto-research harness searches over executable programs $p \in \mathcal{P}$, where candidate programs may modify the model architecture, optimizer, training schedule, and inference implementation.

For each objective $j$, Arena minimizes a resource metric $\mathrm{Target}_j(p)$ subject to a quality threshold $\tau$ (shared by all targets $j$), target-specific resource limits $\mathbf{b}_j$, and correctness checks:

$$\begin{aligned} \underset{p\in\mathcal{P}}{\text{minimize}} \quad & \mathrm{Target}_j(p) \\ \text{subject to} \quad & \mathcal{L}(p) < \tau, \\ & \mathrm{Resources}_j(p) \le \mathbf{b}_j. \end{aligned} \tag{1}$$

Here, $\mathcal{L}(p)$ denotes validation loss, for which lower values are better. The vector $\mathrm{Resources}_j(p)$ contains the constrained resource measurements, and $\mathbf{b}_j$ specifies their limits. Resource inequalities apply componentwise. This formulation separates *admissibility* from *improvement*: candidates qualify by satisfying the quality, resource, and correctness requirements, while lower target values indicate better outcomes among qualifying candidates.

**Efficiency targets.** Arena instantiates this formulation with nine objectives spanning model, training, and inference efficiency. These objectives address complementary bottlenecks in model development and deployment. Tab. 1 summarizes their measurements and practical motivations.

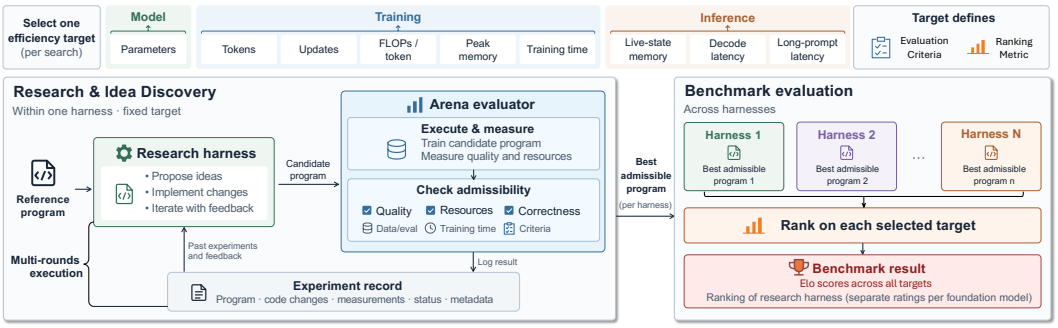



Figure 2: **Overview of Auto-Research Arena Pipeline.** Each search run optimizes one of nine efficiency targets. A research harness proposes and implements changes to a reference program. Arena evaluates the resulting candidates, records outcomes, and provides feedback. The best admissible candidate determines the benchmark result.

Each metric captures a distinct dimension of efficiency. Parameter efficiency measures the model size needed to achieve the required quality. Training efficiency measures the resources needed to reach that quality, while inference efficiency measures the resources needed to use the trained model. Optimizing these metrics separately defines distinct research problems within the same experimental environment.

### 3.2 Language-Modeling Instantiation

We instantiate Arena using Karpathy's autoresearch language-modeling testbed (Karpathy, 2026), in which candidate solutions are compact LLM pretraining programs. We retain its fixed training-data pool and lightweight experimental setup, replacing loss minimization with the constrained formulation in Eq. 1. Each track optimizes one of nine efficiency metrics subject to requirements on model quality and relevant non-target resources.

**Model quality.** Following the original testbed (Karpathy, 2026), we measure language-modeling quality using validation bits per byte (BPB). A candidate is admissible only if its validation BPB is below 1.05. This threshold is slightly above the reference program's BPB, providing flexibility for structural changes while limiting quality degradation.

**Resource constraints.** Each track limits relevant non-target resources to control the trade-offs permitted during optimization. These constraints cover quantities such as model size, training FLOPs, peak training memory, and available training data. Inference tracks also constrain other serving costs. For example, the live-state-memory track imposes a latency bound, while the decode-latency track imposes a live-state-memory bound. Each track therefore measures improvement in its selected metric while requiring other resource costs to remain within specified limits.

**Measurement.** The Arena evaluator measures all target metrics and constrained resources directly. Depending on the target, measurements are taken after a fixed 600-second training period or when the candidate first meets the quality requirement. Training metrics include consumed tokens, FLOPs per token, peak allocated GPU memory, elapsed training time, and optimizer updates. Inference metrics are measured after training using standardized prompt and generation workloads. Appendix A provides complete metric definitions and resource constraints for all nine tracks.

**Example: parameter efficiency.** The parameter-efficiency track seeks the smallest model that reaches the required quality within the specified training-compute and memory budgets. The remaining eight tracks follow the same formulation, each with its own target metric and corresponding resource constraints.

$$\begin{aligned} \underset{p}{\text{minimize}} \quad & N_{\text{param}}(p) \\ \text{subject to} \quad & \text{BPB}(p) < 1.05, \\ & \text{FLOPs}(p) \le 239.1\,\text{M/token}, \\ & \text{Mem}_{\text{train}}(p) \le 44.0\,\text{GiB}. \end{aligned}$$

The BPB threshold of 1.05 permits a modest quality decrease relative to the reference program to accommodate structural changes. The compute and memory limits match the reference program's measured costs, ensuring that parameter reductions do not exceed either resource budget.

## 4 Experiments

We evaluate Arena along its two central roles: benchmarking autonomous research systems and supporting broad idea discovery. We then use its controlled multi-target setting to study how research evolves under different objectives, constraints, budgets, and research histories.

Table 1: **Efficiency targets in Auto-Research Arena.** Each track minimizes one target subject to a shared model-quality threshold and target-specific resource constraints.

| Target | Measured quantity | Why it matters |
|---|---|---|
| **Model** | | |
| Total parameters | Number of model parameters | Model storage and weight-memory footprint (Lan et al., 2019). |
| **Training** | | |
| Data consumption | Tokens used to reach target quality | Learning efficiency under a token budget (Hoffmann et al., 2022). |
| Optimizer updates | Updates to reach the quality threshold | Convergence in fewer updates (Gupta et al., 2018). |
| Training FLOPs | FLOPs per training token | Compute demand per token (Hoffmann et al., 2022). |
| Peak memory | Peak allocated GPU memory | Training feasibility on memory limits (Chen et al., 2016). |
| Training time | Time to reach the quality threshold | Training turnaround (Mattson et al., 2020). |
| **Inference** | | |
| Live-state memory | Memory per active request | Concurrent serving capacity (Kwon et al., 2023). |
| Decode latency | Sequential decoding latency | Autoregressive generation latency (Shazeer, 2019). |
| Long-prompt latency | Prefill + decoding latency | Long-context response latency (Zhong et al., 2024). |

**Benchmark–harness interface.** Arena provides the environment, task specification, reference program, evaluator, and experiment budget, while each harness determines how to conduct the search. Starting from the reference program, a harness proposes code modifications, submits candidates for evaluation, observes the measured outcomes, and decides what to explore next. Arena records every experiment and independently reports the best recorded candidate satisfying the quality, resource, and correctness requirements, regardless of which program the harness ultimately retains. Each search receives 100 charged experiments; an experiment is charged once GPU execution begins, including runs that subsequently fail or are abandoned. Appendix B provides the full interface and protocol.

**Reference baselines.** We introduce two simple reference harnesses that differ primarily in how many research directions they retain. *Sequential Search* maintains one program and evaluates one modification at a time, adopting an admissible improvement before continuing. *Beam Search* retains up to four programs and evaluates two modifications of each before selecting the next frontier. Both condition subsequent proposals on the accumulated experimental history, including unsuccessful attempts. Appendix B provides their detailed proposal, selection, and stopping policies.

**Tasks and resources.** We evaluate all nine Arena objectives defined in Sec. 3, each with target-specific resource constraints and the shared quality requirement BPB $< 1.05$. Each candidate runs on one NVIDIA A100 80GB GPU with seed 42 and up to 600 seconds of training, excluding compilation, validation, and the first ten warmup updates. All searches use the same approximately 631M-token ClimbMix (Diao et al., 2025) training pool and held-out BPB evaluator; training-token consumption includes repeated exposure to the data.

**Models and harnesses.** We evaluate six LLM-based research harnesses with Claude Opus 5 (Anthropic, 2026) and GPT-6 Astra (OpenAI, 2026), both using medium reasoning effort. In addition to Sequential Search and Beam Search, we evaluate AutoScientists (Gao et al., 2026), Arbor (Jin et al., 2026), GEAR (Jeddi et al., 2026), and Heuresis (Antoniades et al., 2026), which respectively use multi-agent coordination, hypothesis trees, evolutionary search, and diversity-preserving archives. We additionally include TPE (Ferreira et al., 2026), a non-agentic baseline restricted to a predefined hyperparameter space, in the Opus 5 experiments.

**Scoring.** Each objective is scored by the best recorded admissible target value, with lower values better. For training-token consumption, optimizer updates, and training time, we use the first 30-second validation checkpoint satisfying BPB $< 1.05$. We summarize performance across objectives with different units and scales using Elo ratings from pairwise harness comparisons within each objective. Appendix D gives the scoring procedure.

### 4.1 Benchmarking Auto-Research Harnesses

Tab. 2 reports the best results across Arena's nine objectives, while Fig. 3 shows representative search trajectories.

**Arena provides a discriminative evaluation.** Research harnesses exhibit substantial differences across Arena's objectives, with no single method dominating the benchmark. With Opus 5, AutoScientists achieves the highest aggregate Elo, followed by Sequential Search and Arbor; with GPT-6, GEAR ranks first, followed by Sequential Search. The variation across objectives shows that Arena

Table 2: **Benchmark performance across nine research objectives.** Lower objective values and higher Elo ratings are better; bold indicates the best result within each foundation-model group. *Charged experiments* reports the mean number of budgeted experiments across objectives. Elo is computed separately within each foundation-model group. TPE is a non-agentic baseline and is included only in the Opus 5 comparison.

| Agent model | Harness | Model | Training efficiency | | | | | Inference efficiency | | | Charged experiments (mean) | Elo ↑ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Params. (M) | Tokens (M) | Updates | Time (s) | FLOPs (M/token) | Memory (GiB) | KV cache (KiB) | Decode (ms) | Request (ms) | | |
| **Reference starting point** | | 50.33 | 398.98 | 761 | 600 | 239.08 | 43.957 | 33,790 | 465.28 | 641.60 | – | – |
| Opus 5 | AutoScientists (Gao et al., 2026) | 13.91 | 184.55 | **267** | 240 | **52.15** | 0.521 | 58.00 | **49.70** | **29.46** | 84.7 | **1729.4** |
| | Sequential Search | 15.15 | **152.46** | 295 | **210** | 56.66 | 0.567 | **1.000** | 98.56 | 61.30 | 87.7 | 1687.6 |
| | Arbor (Jin et al., 2026) | 14.55 | 159.78 | 337 | 270 | 58.30 | **0.373** | 94.50 | 57.09 | 57.88 | 54.0 | 1632.4 |
| | GEAR (Jeddi et al., 2026) | **13.20** | 164.43 | 306 | 270 | 62.23 | 0.434 | 121.5 | 56.01 | 97.84 | 100.0 | 1622.4 |
| | Beam Search | 14.22 | 197.66 | 335 | 270 | 69.85 | 0.951 | 68.00 | 65.15 | 180.56 | 96.9 | 1438.6 |
| | Heuresis (Antoniades et al., 2026) | 17.83 | 156.17 | 377 | 270 | 84.15 | 1.582 | 488.5 | 97.82 | 129.03 | 100.0 | 1350.4 |
| | TPE (Ferreira et al., 2026) | 26.35 | 387.71 | 753 | 510 | 100.27 | 5.949 | 17,790 | 230.74 | 295.39 | 100.0 | 1039.2 |
| GPT-6 | AutoScientists (Gao et al., 2026) | 19.84 | 311.16 | 508 | 420 | 103.81 | 6.006 | 423.5 | 177.78 | 472.56 | 30.2 | 1298.7 |
| | Sequential Search | **15.28** | 223.61 | 350 | 330 | 74.54 | **0.646** | 362.0 | 72.39 | 116.05 | 100.0 | 1702.4 |
| | Arbor (Jin et al., 2026) | 18.87 | 263.19 | 405 | 480 | 77.71 | 1.317 | 3177 | 137.75 | 163.25 | 14.7 | 1459.6 |
| | GEAR (Jeddi et al., 2026) | 18.61 | **159.91** | **302** | **240** | **67.25** | 0.709 | **51.51** | **62.74** | **47.34** | 100.0 | **1835.3** |
| | Beam Search | 27.26 | 274.46 | 339 | 480 | 81.00 | 3.914 | 272.5 | 142.69 | 405.06 | 95.8 | 1423.3 |
| | Heuresis (Antoniades et al., 2026) | 33.03 | 264.37 | 355 | 420 | 125.24 | 2.318 | 8193 | 317.78 | 503.57 | 86.8 | 1280.6 |

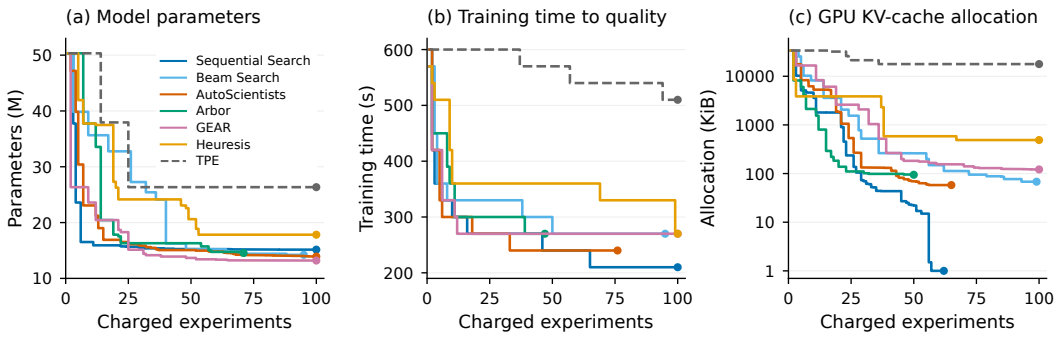



Figure 3: **Research trajectories behind the Opus 5 benchmark results.** Curves track the best admissible result found as each search proceeds, ordered by charged experiments; lower is better. Counts include the reference evaluation and charged failures, and dots mark the end of each recorded search. Training time is measured at 30-second validation checkpoints, and KV-cache allocation is shown on a logarithmic scale.

distinguishes complementary strengths of research systems rather than reducing their performance to a single research setting.

**Simple search is surprisingly competitive.** Despite its minimal design, Sequential Search ranks second overall with both foundation LLMs, outperforming several substantially more sophisticated harnesses. Its strong performance suggests substantial headroom in current auto-research system design: more elaborate orchestration does not yet consistently translate into stronger performance.

**Specialization does not necessarily transfer across research problems.** TPE provides a complementary sanity check. Designed for Karpathy's autoresearch setting (Ferreira et al., 2026), TPE performs substantially worse than all LLM-based harnesses across Arena's broader set of objectives. Strong performance in a specialized research setting therefore need not translate into broad research capability, illustrating the value of evaluating research systems across complementary objectives.

**Research trajectories support the benchmark scores.** The endpoint differences in Tab. 2 are also reflected in the search trajectories in Fig. 3. Stronger harnesses typically make sustained progress over the experiment budget, whereas weaker methods plateau earlier or improve less consistently. For example, TPE stops improving relatively early in the parameter search, while GEAR and AutoScientists continue to find improvements near the end of their recorded searches. These trajectories provide complementary evidence that the benchmark scores reflect systematic differences in research progress rather than isolated final outcomes.

**Similar outcomes can hide different research efficiency.** Endpoint performance does not fully characterize how a result is reached. On the Opus 5 training-time objective, GEAR, Arbor, Beam Search, and Heuresis all reach 270 seconds, but first attain this value after 12, 39, 50, and 99 charged

experiments, respectively. Thus, harnesses with identical final scores can differ substantially in how efficiently they discover the same level of improvement.

### 4.2 A Broader Idea Frontier

**Multi-target research substantially broadens the idea frontier.** Beyond routine hyperparameter tuning, Karpathy's reported loss-driven search identifies two architectural changes, *smear* and *back-out* (Karpathy, 2026). Across Arena's efficiency objectives, our search catalogue contains 32 distinct findings, including 20 that were accepted as improvements during search, which we consolidate into nine idea families spanning model architecture, training, and inference. Different objectives expose qualitatively different research directions, from architectural changes under model-efficiency targets to memory and systems optimizations under training and inference targets.

**The discovered idea frontier mirrors active LLM research.** The breadth of Arena's discoveries is not merely a larger collection of benchmark-specific optimizations: many of them reach questions central to current LLM research. A representative example emerges from parameter efficiency (Tab. 3). Agents first discover recurrent weight sharing, effectively turning the model into a looped Transformer, and further identify the need to adapt residual updates across repeated depth, learning separate attention and MLP residual gains at each depth. This closely mirrors recent work studying residual adaptation as a key component of looped Transformers (Wang et al., 2026). Similar correspondences appear across attention-state reduction through cross-layer K/V sharing (Brandon et al., 2024), MQA/MLA (Shazeer, 2019; DeepSeek-AI, 2024), conditional memory (Cheng et al., 2026), optimizer-state reduction (Zhang et al., 2025d), activation-memory efficiency (Korthikanti et al., 2022), and inference optimization (Aminabadi et al., 2022). Importantly, these discoveries are not simply reproductions of existing methods: they often share the same underlying principles while arriving at different technical mechanisms, offering alternative perspectives on active research questions. The resulting idea frontier therefore demonstrates a broader value of Arena: its diverse objectives expand not only what research systems are evaluated on, but also the space of meaningful research directions they explore.

**Table 3:** A representative searched idea by improving parameter efficiency. Blue highlights searched changes; full discoveries are shown in Appendix C.

**Discovered idea:** Loop Transformer with depth-specific residual gain

**Optimization target:** Parameter efficiency

**BEFORE: INDEPENDENT BLOCKS**

**for** $\ell = 0, \ldots, 7$ :
$u \leftarrow h + \mathrm{Attn}_\ell(h)$
$h \leftarrow u + \mathrm{MLP}_\ell(u)$

**AFTER: SHARED BLOCKS, DEPTH-AWARE GAINS**

**for** $\ell = 0, \ldots, 7$ :
$u \leftarrow h + \alpha_\ell \, \mathrm{Attn}_{\ell \bmod B}(h)$
$h \leftarrow u + \beta_\ell \, \mathrm{MLP}_{\ell \bmod B}(u)$

$B = 1$: one stored block, eight applications. The learned $\alpha_\ell, \beta_\ell$ control attention and MLP residual updates separately at each depth.

**RELATION TO PRIOR WORK**

Recent work scales residual updates with loop count (Wang et al., 2026); here the gains are learned separately for each depth.

**Notation.** $h$: hidden state; $\ell$: depth; $B$: stored blocks. Schematic: normalization and auxiliary paths omitted.

**The broader idea frontier accompanies substantial measured progress.** The searches producing these discoveries also achieve large admissible gains across distinct resource dimensions. In the Opus 5 group, the best programs reduce model size from 50.33M to 13.20M parameters, training computation from 239.08M to 52.15M FLOPs per token, and peak training memory from 43.96 to 0.373 GiB, while satisfying BPB $< 1.05$ and the corresponding resource constraints. However, these endpoint programs combine multiple changes accumulated during search, so their gains do not establish whether the discovered ideas themselves provide reusable knowledge beyond the searches that found them. We test this directly in Sec. 4.3.

### 4.3 Validating Discoveries Beyond the Search Setting

Arena discovers mechanisms under specific objectives and short experimental budgets. To assess their usefulness beyond these conditions, we first test whether a library of discovered ideas can guide a new search for better model quality. We then ask a stricter question: can winning programs from the original Arena searches retain their advantages when retrained for longer without further search?

**Combining discovered ideas improves model quality.** We collect discoveries across Arena's objectives and harnesses into an idea library to guide a quality-driven search. The agent can select, modify, and combine mechanisms from different research trajectories rather than continue from a single winning program. We extend the training time budget to 1 hour in this search.

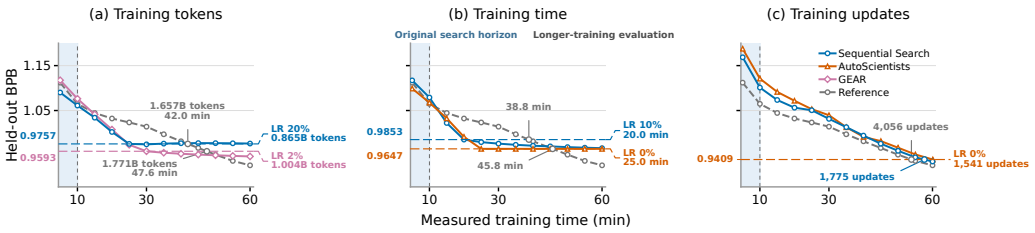



Figure 4: **Discovered programs can transfer beyond the search horizon.** Programs discovered with 10-minute candidate training are retrained from random initialization for 60 minutes alongside the reference. Several retain their advantages for substantially longer than the original search horizon, although these gains can diminish later as horizon-specific training policies take effect. Curves show held-out BPB against training time.

The resulting design improves model quality under both matched-budget evaluations (Fig. 1). At matched training data, it reduces training loss from 2.737 to 2.608 (4.7%); at matched training FLOPs, it reduces training loss from 2.700 to 2.649 (1.9%). Final held-out BPB also decreases from 0.9295 to 0.9112 at the same model size (50.33M parameters). The resulting design selectively adapts and simplifies ideas rather than merely combining their original implementations. These results demonstrate that efficiency-targeted discoveries can serve as reusable model-design knowledge for improving quality. Appendix C.3 details how these ideas are selected and transformed.

**Discovered programs can transfer far beyond the search horizon.** We next ask a stricter question: can a program found under a short training budget be scaled directly, without further search? Arena normally evaluates candidate programs after 10 minutes of training. We select two programs from each of the training-time, token, and update objectives and retrain them from random initialization for 60 minutes alongside the reference program.

Several discoveries retain their advantages far beyond the horizon at which they were found. A token-optimized GEAR program reaches 0.9593 BPB after 30 minutes using 1.004B tokens, while the reference requires approximately 47.6 minutes and 1.771B tokens to reach the same quality, corresponding to 43.3% fewer tokens. Its quality advantage persists until approximately 51 minutes, extending to five times the original search horizon. Similarly, a training-time program discovered by AutoScientists outperforms the reference over the 15–46-minute interval and reaches 0.9647 BPB after 25 minutes, a quality level that the reference reaches only after approximately 45.8 minutes, corresponding to 45.4% less training time (Figure 4; Table 4). Reference costs at matched quality are estimated by linear interpolation.

The transfer is substantial but not indefinite. The GEAR program's learning rate reaches its 2% floor around 27 minutes, while the AutoScientists schedule reaches zero around 24 minutes. Their early advantages consequently diminish as training continues; by 60 minutes, for example, the reference reaches 0.9295 BPB compared with 0.9478 for the GEAR program. Thus, short-horizon search can discover programs whose advantages persist for four to five times the original search horizon, while horizon-dependent training policies can ultimately limit direct extrapolation to longer training.

### 4.4 The Science of Auto-Research

Arena also enables controlled study of what shapes the auto-research process. We examine quality constraints, budget allocation between evaluation depth and search breadth, and the role of research history in cross-target search.

**Constraints shape the research space.** We vary the BPB threshold across 1.02, 1.05, and 2.0 for parameter- and memory-efficiency searches with Sequential Search and Beam Search (Tab. 5). Relaxing the threshold from 1.05 to 2.0 allows Sequential Search to reduce parameters from 18.94M to 0.69M, while its separate peak-memory search reduces memory from 0.83 GiB to 0.035 GiB. The additional quality slack permits substantially more aggressive efficiency improvements. Conversely, tighter thresholds restrict which structural changes remain admissible. Quality constraints thus shape not only the final results, but also which regions of the design space can be developed.

**Budget allocation creates a depth–breadth trade-off.** For parameter minimization under BPB $< 2.0$, we vary candidate training time from 1,200 to 6,000 seconds under a shared 20 GPU-hour training budget, trading fewer, longer evaluations for more, shorter ones (Tab. 8). The shortest setting

Table 4: **Direct transfer beyond the search horizon.** Programs discovered with 10-minute training are retrained for up to 60 minutes without further search. Savings compare each program with the reference at the same held-out BPB.

| Objective | Harness | Resource saving | Quality advantage to | 60-min BPB |
|---|---|---|---|---|
| Reference | | – | – | 0.9295 |
| Training time | Seq. Search | 28.5% | ∼44 min | 0.9662 |
| | AutoScientists | **45.4%** | ∼46 min | 0.9647 |
| Training tokens | GEAR | **43.3%** | ∼51 min | 0.9478 |
| | Seq. Search | 34.4% | ∼42 min | 0.9765 |
| Training updates | AutoScientists | 30.1% | – | 0.9409 |
| | Seq. Search | 39.6% | – | 0.9365 |

Table 5: **Looser quality constraints enable more aggressive efficiency gains.** Results use 30 completed runs per search; bold marks each harness's best value.

| Harness | BPB Limit | Parameters | | Peak GPU Memory | |
|---|---|---|---|---|---|
| | | Params. (M) | BPB | Memory (GiB) | BPB |
| Beam Search | $< 1.02$ | 23.5934 | **1.0185** | 3.9753 | **1.0175** |
| | $< 1.05$ | 19.6610 | 1.0448 | 1.6707 | 1.0371 |
| | $< 2.0$ | **0.4649** | 1.5030 | **0.1625** | 1.2312 |
| Sequential Search | $< 1.02$ | 26.4767 | **1.0195** | 6.8889 | **1.0162** |
| | $< 1.05$ | 18.9399 | 1.0497 | 0.8341 | 1.0402 |
| | $< 2.0$ | **0.6924** | 1.9963 | **0.0348** | 1.7200 |

Table 6: **Research can build across objectives.** Each sequence continues from a program discovered under the first objective and optimizes a new objective. The second search can preserve the earlier gain or trade part of it for progress on the new objective. $\Delta_2$ reports the change from the first-search program; negative values indicate improvements.

| Research sequence | Metric | Reference | After first search | After second search | $\Delta_2$ |
|---|---|---|---|---|---|
| Parameters → FLOPs | Parameters (M) | 50.33 | 14.09 | 14.09 | 0.0% |
| | Training FLOPs (M/token) | 239.08 | 113.80 | 75.46 | $-\mathbf{33.7}\%$ |
| FLOPs → Decode latency | Training FLOPs (M/token) | 239.08 | 66.55 | 66.55 | 0.0% |
| | Uncached decode latency (ms) | 552.09 | 520.36 | 65.17 | $-\mathbf{87.5}\%$ |
| Parameters → Training time | Parameters (M) | 50.33 | 14.09 | 17.56 | +24.6% |
| | Time to BPB $< 1.05$ (s) | 600 | 570 | 450 | $-\mathbf{21.1}\%$ |

evaluates 60 candidates and finds the smallest model; longer settings yield better BPB but more parameters. These outcomes illustrate the trade-off between evaluation depth and search breadth, with different allocations favoring different quality–efficiency balances.

**Cross-target research builds on prior discoveries.** We initialize searches under new objectives from programs discovered under earlier objectives, transferring code while reinitializing model weights (Tab. 6). Subsequent searches can accumulate compatible gains: parameter savings are fully preserved while training FLOPs decrease by a further 33.7%, and a FLOP-optimized program preserves its FLOP count while uncached decode latency decreases by 87.5%. Cross-target continuation can also introduce trade-offs: optimizing training time from the parameter-optimized program reduces time to quality by 21.1% but increases parameters by 24.6%. Prior discoveries therefore provide foundations for both cumulative gains and new trade-offs across objectives.

Together, these studies show how constraints define admissible designs, budgets balance evaluation depth against search breadth, and research history supports cumulative gains or trade-offs across targets. Therefore, Arena enables systematic study of how autonomous research develops over time.

## 5 Conclusion

We introduced Auto-Research Arena, a multi-target benchmark for autonomous research across nine quality-constrained language-model efficiency objectives. Arena reveals complementary harness strengths and a broad frontier of architectural and systems ideas. Discoveries can seed searches under new objectives, while a program combining selected ideas can improve model quality. Controlled studies further show how constraints, budgets, and research history shape progress. Together, these results support evaluating autonomous research across complementary objectives and studying how useful discoveries accumulate.

**Limitations.** Our scope is explicitly limited to auto-research in language modeling, rather than the full end-to-end research process or scientific novelty. Idea-family groupings are interpretive, and noisy system measurements require care when interpreting small differences.

### AI USE STATEMENT

LLM-based research agents are the subjects of evaluation in this work; their roles in proposing ideas, implementing programs, and conducting experiments are described in the paper.

Separately, we used AI tools to assist with literature search, compilation of experimental records, text polishing for readability and figure preparation.

We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, supporting qualitative and thematic data analysis, and interpreting results.

Generating synthetic data sets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, cleaning and reformating dataset, are not applicable to this work.

We have reviewed AI-assisted content against the relevant literature, source code, and experimental records. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

This work studies autonomous research systems in a controlled language-modeling environment. The experiments use existing language-model training data and do not involve human subjects, personal data collection, or deployment to users.

The evaluated agents can autonomously propose code changes and execute experiments, which may raise broader concerns about increasingly capable autonomous research systems. In our setting, their actions are restricted to a sandboxed experimental environment, a fixed program interface, predefined compute budgets, and standardized evaluation. We do not evaluate or enable autonomous interaction with external systems beyond the research environment.

We report both successful and unsuccessful research outcomes and distinguish measured improvements from interpretations of scientific novelty. We hope that the benchmark can support more systematic study of the capabilities and limitations of autonomous research systems.

### REPRODUCIBILITY STATEMENT

We provide the benchmark formulation, all nine target definitions, quality and resource constraints, measurement procedures, and the interface between Arena and research harnesses in the paper and Appendix A–B. Experimental hardware, training data, random seeds, training budgets, experiment accounting, and scoring procedures are also specified.

Arena records candidate programs, code changes, execution outcomes, measurements, admissibility decisions, and search lineages. Appendix C provides additional discovery evidence and protocols for the supplementary studies, including idea-reuse experiments. Appendix D specifies the Elo aggregation procedure.

We will release the Arena implementation, evaluator, reference program, benchmark configurations, research traces, and scripts used to reproduce the reported tables and figures.

## REFERENCES

Reza Yazdani Aminabadi, Samyam Rajbhandari, Minjia Zhang, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Jeff Rasley, Shaden Smith, Olatunji Ruwase, and Yuxiong He. DeepSpeed Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale. *arXiv preprint arXiv:2207.00032*, 2022.

Anthropic. Introducing claude opus 5. `https://www.anthropic.com/news/claude-opus-5`, July 2026. Anthropic model release announcement.

Antonis Antoniades, Deepak Nathani, Ritam Saha, Alfonso Amayuelas, Ivan Bercovich, Zhaotian Weng, Vignesh Baskaran, Kunal Bhatia, and William Yang Wang. Heuresis: Search strategies for autonomous AI research agents across quality, diversity and novelty. *arXiv preprint arXiv:2606.25198*, 2026.

Ben Bogin, Kejuan Yang, Shashank Gupta, Kyle Richardson, Erin Bransom, Peter Clark, Ashish Sabharwal, and Tushar Khot. SUPER: Evaluating agents on setting up and executing tasks from research repositories. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 12622–12645, 2024.

Jonathan Bragg, Mike D'Arcy, Nishant Balepur, Dan Bareket, Bhavana Dalvi Mishra, Sergey Feldman, Dany Haddad, Jena Hwang, Peter Jansen, Varsha Kishore, Bodhisattwa Prasad Majumder, Aakanksha Naik, Sigal Rahamimov, Kyle Richardson, Amanpreet Singh, Harshit Surana, Aryeh Tiktinsky, Rosni Vasu, Guy Wiener, Chloe Anastasiades, Stefanus Candra, Jason Dunkelberger, Daniel Emery, Rob Evans, Malachi Hamada, Regan Huff, Rodney Kinney, Matt Latzke, Jaron Lochner, Ruben Lozano-Aguilera, Ngoc-Uyen Nguyen, Smita Rao, Amber Tanaka, Brooke Vlahos, Peter Clark, Doug Downey, Yoav Goldberg, Ashish Sabharwal, and Daniel Weld. AstaBench: Rigorous benchmarking of AI agents with a scientific research suite. In *International Conference on Learning Representations*, pp. 110136–110223, 2026.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. Reducing Transformer Key-Value Cache Size with Cross-Layer Attention. *arXiv preprint arXiv:2405.12981*, 2024.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*, 2025.

Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. MLR-Bench: Evaluating AI agents on open-ended machine learning research. In *Advances in Neural Information Processing Systems*, volume 38, 2025a.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training Deep Nets with Sublinear Memory Cost. *arXiv preprint arXiv:1604.06174*, 2016.

Zhi Chen, Zhensu Sun, Yuling Shi, David Lo, and Lingxiao Jiang. Are performance-optimization benchmarks reliably measuring coding agents? *arXiv preprint arXiv:2607.01211*, 2026.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Xin Cheng, Rui Tian, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Chengqi Deng, Shangyan Zhou, Chenggang Zhao, Zhewen Hao, Yukun Li, Han Zhang, Zhengyan Zhang, Yixu Wei, M. Y Xu, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. Conditional Memory via Scalable Lookup: A New Axis of Sparsity for Large Language Models. *arXiv preprint arXiv:2601.07372*, 2026.

Jiale Cui, Yueyao Yuan, Kaixi Zhong, Xiaogang Xu, Jiafei Wu, and Zhe Liu. ARAC: Benchmarking auto-research's alignment and completeness on end-to-end researchs. *arXiv preprint arXiv:2608.12788*, 2026.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model, 2024.

Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, Mostofa Patwary, Yingyan (Celine) Lin, Jan Kautz, and Pavlo Molchanov. Nemotron-CLIMB: Clustering-based iterative data mixture bootstrapping for language model pre-training. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Nicholas Edwards, Yukyung Lee, Yujun Audrey Mao, Yulu Qin, Sebastian Schuster, and Najoung Kim. RExBench: Can coding agents autonomously implement AI research extensions? In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16380–16417, 2026.

Haoyang Fang, Boran Han, Nick Erickson, Xiyuan Zhang, Su Zhou, Anirudh Dagar, Jiani Zhang, Ali Caner Turkmen, Tony Hu, Huzefa Rangwala, Ying Nian Wu, Yuyang (Bernie) Wang, and George Karypis. MLZero: A multi-agent system for end-to-end machine learning automation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 69001–69070, 2025.

Fabio Ferreira, Lucca Wobbe, Arjun Krishnakumar, Frank Hutter, and Arber Zela. Can llms beat classical hyperparameter optimization algorithms? a study on autoresearch. 2026.

Shanghua Gao, Ada Fang, and Marinka Zitnik. AutoScientists: Self-organizing agent teams for long-running scientific experimentation. *arXiv preprint arXiv:2605.28655*, 2026.

Aniketh Garikaparthi, Manasi Patwardhan, and Arman Cohan. ResearchGym: Evaluating language model agents on real-world AI research. In *ICLR 2026 Workshop on Agents in the Wild*, 2026.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, Anil Palepu, Keran Rong, Ryutaro Tanno, Khaled Saab, Fan Zhang, Jacob Blum, Andrew Carroll, Kavita Kulkarni, Nenad Tomašev, Dina Zverinski, Ivor Rendulic, Elahe Vedadi, Florian Hasler, Luka Rimanic, Marina Boia, Ivan Budiselic, Ben Feinstein, Mathias Bellaiche, Tom Sheffer, Jan Freyberg, Jeremy Ratcliff, Ottavia Bertolli, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R. D. Costa, José R. Penadés, Gary Peltz, Yossi Matias, James Manyika, Demis Hassabis, Yunhan Xu, Pushmeet Kohli, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Accelerating scientific discovery with Co-Scientist. *Nature*, 655:487–496, 2026.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1842–1850, 10–15 Jul 2018.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030, 2022.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025.

Tianyu Hua, Harper Hua, Violet Xiang, Benjamin Klieger, Sang Truong, Weixin Liang, Fan-Yun Sun, and Nick Haber. ResearchCodeBench: Benchmarking LLMs on implementing novel machine learning research code. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Intology. NanoGPT-Bench: Evaluating autonomous research agents on the NanoGPT speedrun. `https://github.com/IntologyAI/NanoGPT-Bench`, 2026. GitHub repository.

Ahmadreza Jeddi, Minh Ngoc Le, Hakki C. Karaimer, Konstantinos G. Derpanis, and Babak Taati. GEAR: Genetic AutoResearch for agentic code evolution. *arXiv preprint arXiv:2605.13874*, 2026.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024.

Jiajie Jin, Yuyang Hu, Kai Qiu, Qi Dai, Chong Luo, Guanting Dong, Xiaoxi Li, Tong Zhao, Xiaolong Ma, Gongrui Zhang, Zhirong Wu, Bei Liu, Zhengyuan Yang, Linjie Li, Lijuan Wang, Hongjin Qian, Yutao Zhu, and Zhicheng Dou. Toward generalist autonomous research via hypothesis-tree refinement. *arXiv preprint arXiv:2606.11926*, 2026.

Andrej Karpathy. autoresearch: AI agents running research on single-GPU nanochat training automatically. `https://github.com/karpathy/autoresearch`, 2026. GitHub repository.

Patrick Tser Jern Kon, Qiuyi Ding, Jiachen Liu, Xinyi Zhu, Jingjia Peng, Jiarong Xing, Yibo Huang, Yiming Qiu, Jayanth Srinivasa, Myungjin Lee, Mosharaf Chowdhury, Matei Zaharia, and Ang Chen. EXP-bench: Can AI conduct AI research experiments? In *The Fourteenth International Conference on Learning Representations*, 2026.

Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing Activation Recomputation in Large Transformer Models. *arXiv preprint arXiv:2205.05198*, 2022.

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*, pp. 611–626, 2023.

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.

Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32201–32223, 2024.

Jiaqi Liu, Shi Qiu, Mairui Li, Bingzhou Li, Haonian Ji, Siwei Han, Xinyu Ye, Peng Xia, Zihan Dong, Meng Chen, Congyu Zhang, Letian Zhang, Guiming Chen, Haoqin Tu, Xinyu Yang, Lu Feng, Xujiang Zhao, Haifeng Chen, Jiawei Zhou, Xiao Wang, Weitong Zhang, Hongtu Zhu, Yun Li, Jieru Mei, Hongliang Fei, Jiaheng Zhang, Linjie Li, Linjun Zhang, Yuyin Zhou, Sheng Wang, Caiming Xiong, James Zou, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. AutoResearchClaw: Self-reinforcing autonomous research with human-AI collaboration. *arXiv preprint arXiv:2605.20025*, 2026.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of AI research. *Nature*, 651:914–919, 2026.

Alisia Lupidi, Bhavul Gauri, Thomas Simon Foster, Bassel Al Omari, Despoina Magka, Alberto Pepe, Alexis Audran-Reiss, Muna Aghamelu, Nicolas Baldwin, Lucia Cipolina-Kun, Jean-Christophe Gagnon-Audet, Chee Hau Leow, Sandra Lefdal, Hossam Mossalam, Abhinav Moudgil, Saba Nazir, Emanuel Tewolde, Isabel Urrego, Jordi Armengol Estape, Amar Budhiraja, Gaurav Chaurasia, Abhishek Charnalia, Derek Dunfield, Karen Hambardzumyan, Daniel Izcovich, Martin Josifoski, Ishita Mediratta, Kelvin Niu, Parth Pathak, Michael Shvartsman, Edan Toledo, Anton Protopopov, Roberta Raileanu, Alexander Miller, Tatiana Shavrina, Jakob Foerster, and Yoram Bachrach. AIRS-Bench: a suite of tasks for frontier AI research science agents. *arXiv preprint arXiv:2602.06855*, 2026.

Bodhisattwa Prasad Majumder, Harshit Surana, Dhruv Agarwal, Bhavana Dalvi Mishra, Abhijeetsingh Meena, Aryan Prakhar, Tirth Vora, Tushar Khot, Ashish Sabharwal, and Peter Clark. DiscoveryBench: Towards data-driven discovery with large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Peter Mattson, Christine Cheng, Gregory Diamos, Cody Coleman, Paulius Micikevicius, David Patterson, Hanlin Tang, Gu-Yeon Wei, Peter Bailis, Victor Bittorf, David Brooks, Dehao Chen, Debo Dutta, Udit Gupta, Kim Hazelwood, Andy Hock, Xinyuan Huang, Daniel Kang, David Kanter, Naveen Kumar, Jeffery Liao, Deepak Narayanan, Tayo Oguntebi, Gennady Pekhimenko, Lillian Pentecost, Vijay Janapa Reddi, Taylor Robie, Tom St John, Carole-Jean Wu, Lingjie Xu, Cliff

Young, and Matei Zaharia. MLPerf Training Benchmark. In I. Dhillon, D. Papailiopoulos, and V. Sze (eds.), *Proceedings of Machine Learning and Systems*, volume 2, pp. 336–349, 2020.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

Henry E. Miller, Matthew Greenig, Benjamin Tenmann, and Bo Wang. BioML-bench: Evaluation of AI agents for end-to-end biomedical ML. *bioRxiv*, 2025.

Ludovico Mitchener, Angela Yiu, Benjamin Chang, Mathieu Bourdenx, Tyler Nadolski, Arvis Sulovari, Eric C. Landsness, Daniel L. Barabasi, Siddharth Narayanan, Nicky Evans, Shriya Reddy, Martha Foiani, Aizad Kamal, Leah P. Shriver, Fang Cao, Asmamaw T. Wassie, Jon M. Laurent, Edwin Melville-Green, Mayk Caldas, Albert Bou, Kaleigh F. Roberts, Sladjana Zagorac, Timothy C. Orr, Miranda E. Orr, Kevin J. Zwezdaryk, Ali E. Ghareeb, Laurie McCoy, Bruna Gomes, Euan A. Ashley, Karen E. Duff, Tonio Buonassisi, Tom Rainforth, Randall J. Bateman, Michael Skarlinski, Samuel G. Rodriques, Michaela M. Hinks, and Andrew D. White. Kosmos: An AI scientist for autonomous discovery. *arXiv preprint arXiv:2511.02824*, 2025.

Yongyu Mu, Yuzhang Wu, Yuchun Fan, Chenglong Wang, Hengyu Li, Jiali Zeng, Qiaozhi He, Murun Yang, Fandong Meng, Jie Zhou, Tong Xiao, and Jingbo Zhu. Cross-layer Attention Sharing for Pre-trained Large Language Models. *arXiv preprint arXiv:2408.01890*, 2024.

Nitya Nadgir, Sayash Kapoor, Kangheng Liu, Peter Kirgis, Matilda Orona, Stephan Rabanser, Tilman Bayer, Abhishek Shetty, Yue Ling, Derrick Chan-Sew, Rumi Nakagawa, Saiteja Utpala, Zachary S. Siegel, and Arvind Narayanan. Life after benchmark saturation: A case study of CORE-Bench. *arXiv preprint arXiv:2606.26158*, 2026.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Mikhail Plekhanov, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, Dieuwke Hupkes, Ricardo Silveira Cabral, Tatiana Shavrina, Jakob Nicolaus Foerster, Yoram Bachrach, William Yang Wang, and Roberta Raileanu. MLGym: A new framework and benchmark for advancing AI research agents. In *Second Conference on Language Modeling*, 2025.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

OpenAI. GPT-6 astra: A new generation of intelligence. `https://openai.com/index/gpt-6-astra/`, September 2026. OpenAI model release announcement.

Anne Ouyang, Simon Guo, Simran Arora, Alex L. Zhang, William Hu, Christopher Ré, and Azalia Mirhoseini. KernelBench: Can LLMs write efficient GPU kernels? In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47356–47415, 2025.

Ofir Press and Lior Wolf. Using the Output Embedding to Improve Language Models. *arXiv preprint arXiv:1608.05859*, 2016.

Ori Press, Brandon Amos, Haoyu Zhao, Yikai Wu, Samuel Ainsworth, Dominik Krupke, Patrick Kidger, Touqir Sajed, Bartolomeo Stellato, Jisun Park, Nathanael Bosch, Eli Meril, Albert Steppi, Arman Zharmagambetov, Fangzhao Zhang, David Pérez-Piñeiro, Alberto Mercurio, Ni Zhan, Talor Abramovich, Kilian Lieret, Hanlin Zhang, Shirley Huang, Matthias Bethge, and Ofir Press. AlgoTune: Can language models speed up general-purpose numerical programs? In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. PostTrainBench: Can LLM agents automate LLM post-training? In *International Conference on Machine Learning*, 2026.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent Laboratory: Using LLM agents as research assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, 2025.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Manish Shetty, Naman Jain, Jinjian Liu, Vijay Kethanaboyina, Koushik Sen, and Ion Stoica. GSO: Challenging software optimization tasks for evaluating SWE-Agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Zachary S. Siegel, Sayash Kapoor, Nitya Nadgir, Benedikt Stroebl, and Arvind Narayanan. CORE-Bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Dhruv Srikanth, Bingchen Zhao, Dixing Xu, Yuxiang Wu, and Zhengyao Jiang. Recursive self-improvement of AI research agents. *arXiv preprint arXiv:2609.26457*, 2026.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's ability to replicate AI research. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 56843–56873. PMLR, 2025.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. AI-researcher: Autonomous scientific innovation. In *Advances in Neural Information Processing Systems*, 2025.

Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, Shengyan Liu, Di Luo, Yutao Ma, Hao Tong, Kha Trinh, Chenyu Tian, Zihan Wang, Bohao Wu, Yanyu Xiong, Shengzhu Yin, Minhui Zhu, Kilian Lieret, Yanxin Lu, Genglin Liu, Yufeng Du, Tianhua Tao, Ofir Press, Jamie Callan, Eliu Huerta, and Hao Peng. SciCode: A research coding benchmark curated by scientists. In *Advances in Neural Information Processing Systems*, volume 37, pp. 30624–30650, 2024.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Lupidi, Andrei Lupu, Roberta Raileanu, Tatiana Shavrina, Kelvin Niu, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander Miller, Abhishek Charnalia, Derek Dunfield, Carole-Jean Wu, Pontus Lars Erik Saito Stenetorp, Nicola Cancedda, Jakob Foerster, and Yoram Bachrach. AI research agents for machine learning: Search, exploration, and generalization in MLE-bench. In *Advances in Neural Information Processing Systems*, volume 38, pp. 35309–35348, 2025.

Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. AutoML-Agent: A multi-agent LLM framework for full-pipeline AutoML. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 60099–60146, 2025.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. PutnamBench: Evaluating neural theorem-provers on the putnam mathematical competition. In *Advances in Neural Information Processing Systems*, volume 37, pp. 11545–11569, 2024.

Xinming Tu, Tianze Wang, Yingzhou (Minta) Lu, Kexin Huang, Yuanhao Qu, and Sara Mostafavi. BenchGuard: Who guards the benchmarks? automated auditing of LLM agent benchmarks. *arXiv preprint arXiv:2604.24955*, 2026.

Shaowen Wang, Bingrui Li, Ge Zhang, Wenhao Huang, Shen Yan, and Jian Li. On the Residual Scaling of Looped Transformers: Stability and Transferability. *arXiv preprint arXiv:2606.18524*, 2026.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

Hjalmar Wijk, Tao Roa Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Joshua M Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Jun Koba Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66772–66832, 2025.

Yunze Wu, Dayuan Fu, Weiye Si, Zhen Huang, Mohan Jiang, Keyu Li, Shijie Xia, Jie Sun, Tianze Xu, Yang Xiao, Pengrui Lu, Xiaojie Cai, Lyumanshan Ye, Wenhong Zhu, Xiangkun Hu, and Pengfei Liu. InnovatorBench: Evaluating agents' ability to conduct innovative AI research. In *International Conference on Learning Representations*, pp. 36144–36185, 2026.

Lei Xiong, Kun Luo, Ziyi Xia, Wenbo Zhang, Jin-Ge Yao, Zheng Liu, Jingying Shao, Jianlyu Chen, Hongjin Qian, Xi Yang, Qian Yu, Hao Li, Chen Yue, Xiaan Du, Yuyang Wang, Yesheng Liu, Haiyu Xu, and Zhicheng Dou. AutoResearchBench: Benchmarking AI agents on complex scientific literature discovery. *arXiv preprint arXiv:2604.25256*, 2026.

Wanghan Xu, Shuo Li, Tianlin Ye, Qinglong Cao, Yixin Chen, Hengjian Gao, Yiheng Wang, Qi Li, Kun Li, Sheng Xu, et al. ResearchClawBench: A benchmark for end-to-end autonomous scientific research. *arXiv preprint arXiv:2606.07591*, 2026.

Shuo Yan, Ruochen Li, Ziming Luo, Zimu Wang, Daoyang Li, Liqiang Jing, Kaiyu He, Peilin Wu, Juntong Ni, George Michalopoulos, Yue Zhang, Ziyang Zhang, Mian Zhang, Zhiyu Chen, and Xinya Du. LMR-BENCH: Evaluating LLM agent's ability on reproducing language modeling research. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6164–6186, 2025.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware Singular Value Decomposition for Compressing Large Language Models. *arXiv preprint arXiv:2312.05821*, 2023.

Pedram Zamirai, Jian Zhang, Christopher R. Aberger, and Christopher De Sa. Revisiting BFloat16 Training. *arXiv preprint arXiv:2010.06192*, 2020.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025a.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, XiongHui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Yunxiang Zhang, Muhammad Khalifa, Shitanshu Bhushan, Grant Murphy, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. MLRC-Bench: Can language agents solve machine learning research challenges? In *Advances in Neural Information Processing Systems*, volume 38, 2025c.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik P. Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use Fewer Learning Rates To Gain More. In *International Conference on Learning Representations*, 2025d.

Zhengxin Zhang, Ning Wang, Sainyam Galhotra, and Claire Cardie. How far are we from true auto-research? *arXiv preprint arXiv:2605.19156*, 2026.

Bingchen Zhao, Despoina Magka, Minqi Jiang, Xian Li, Roberta Raileanu, Tatiana Shavrina, Jean-Christophe Gagnon-Audet, Kelvin Niu, Shagun Sodhani, Michael Shvartsman, Andrei Lupu, Alisia Lupidi, Karen Hambardzumyan, Martin Josifoski, Edan Toledo, Thomas Foster, Lucia Cipolina Kun, Derek Dunfield, Abhishek Charnalia, Alexander Miller, Oisin Mac Aodha, Jakob

Foerster, and Yoram Bachrach. The automated LLM speedrunning benchmark: Reproducing NanoGPT improvements. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 193–210, Santa Clara, CA, July 2024.

# A Efficiency Target Definitions

The following specifications use the objective names in Table 1. Displayed resource limits are rounded; each data-pool constraint fixes the available pool rather than the number of tokens consumed.

**Total parameters**

**Why it matters.** Fewer parameters shrink checkpoints and reduce model storage and transfer costs.

**How we measure it.** The model's total parameter count after the fixed 600-second training schedule.

| | |
|---|---|
| **Optimize** | Minimize total parameters. |
| **Maintain quality** | Validation BPB $< 1.05$ (fixed quality threshold) |
| **Stay within limits** | Training FLOPs $\leq$**239.1 M/token** |
| | Peak training memory $\leq$**44.0 GiB** |
| | Available training-data pool **Fixed 631.2 M-token pool** |

**Training tokens to quality**

**Why it matters.** Consuming fewer tokens reduces the data processing needed to train a model of the required quality.

**How we measure it.** Total training tokens consumed by the first scheduled validation check with BPB below 1.05, including warmup updates, gradient accumulation, and repeated use of data. This is token consumption, not the number of unique dataset positions visited.

| | |
|---|---|
| **Optimize** | Minimize consumed training tokens. |
| **Maintain quality** | Validation BPB $< 1.05$ (fixed quality threshold) |
| **Stay within limits** | Training FLOPs $\leq$**239.1 M/token** |
| | Total parameters $\leq$**50.3 M** |
| | Peak training memory $\leq$**44.0 GiB** |
| | Available training-data pool **Fixed 631.2 M-token pool** |

**Training FLOPs**

**Why it matters.** Fewer FLOPs reduce training computation independently of hardware and implementation speed.

**How we measure it.** Forward-and-backward FLOPs per token, measured on the uncompiled model using executed operations and registered counting formulas. The count includes matrix, convolution, and attention work, but excludes optimizer updates and unregistered operations such as elementwise operations and normalization.

| | |
|---|---|
| **Optimize** | Minimize FLOPs per token. |
| **Maintain quality** | Validation BPB $< 1.05$ (fixed quality threshold) |
| **Stay within limits** | Total parameters $\leq$**50.3 M** |
| | Peak training memory $\leq$**44.0 GiB** |
| | Available training-data pool **Fixed 631.2 M-token pool** |

**Peak training memory**

**Why it matters.** Lower memory lets training fit on devices with less capacity or use larger batches.

**How we measure it.** Peak PyTorch-allocated GPU memory from process setup through the reporting read, including FLOPs probing, training, and validation under the fixed 600-second schedule. Validation uses a fixed batch of 128 sequences of 2,048 tokens and can determine the peak. This is allocated memory, not allocator-reserved memory or total device usage.

| | |
|---|---|
| **Optimize** | Minimize peak allocated memory. |
| **Maintain quality** | Validation BPB $< 1.05$ (fixed quality threshold) |
| **Stay within limits** | Training FLOPs $\leq$**239.1 M/token** |
| | Total parameters $\leq$**50.3 M** |
| | Available training-data pool **Fixed 631.2 M-token pool** |

**Training time**

**Why it matters.** Shorter training time reaches the required quality sooner and captures hardware and implementation effects that FLOPs miss.

**How we measure it.** Measured training time at the first scheduled validation check with BPB below 1.05. Startup, compilation, evaluation, and the first ten warmup updates are excluded; each subsequent update includes all accumulation microbatches and the optimizer step, with GPU synchronization at its boundaries.

| | |
|---|---|
| **Optimize** | Minimize measured training time. |
| **Maintain quality** | Validation BPB $<$ 1.05 (fixed quality threshold) |
| **Stay within limits** | Training FLOPs $\leq$**239.1 M/token** |
| | Total parameters $\leq$**50.3 M** |
| | Peak training memory $\leq$**44.0 GiB** |
| | Available training-data pool **Fixed 631.2 M-token pool** |

**Training updates**

**Why it matters.** Fewer updates indicate faster optimization progress, separate from the duration or arithmetic cost of each update.

**How we measure it.** Total logical optimizer updates completed by the first scheduled validation check with BPB below 1.05, including warmup updates. Accumulation microbatches within one update are not counted as separate updates.

| | |
|---|---|
| **Optimize** | Minimize optimizer updates. |
| **Maintain quality** | Validation BPB $<$ 1.05 (fixed quality threshold) |
| **Stay within limits** | Training FLOPs $\leq$**239.1 M/token** |
| | Total parameters $\leq$**50.3 M** |
| | Peak training memory $\leq$**44.0 GiB** |
| | Available training-data pool **Fixed 631.2 M-token pool** |

**Live-state memory (KV cache)**

**Why it matters.** Retained state grows with the live context and limits either context length or concurrent requests during generation.

**How we measure it.** The increase in PyTorch-allocated GPU memory while retaining one request state after a 1,536-token prefill and 512 decoding steps, with capacity 2,048. Measurement follows a discarded warmup request and constructs the state with CUDA graph capture disabled. It measures retained allocation rather than transient peak memory or a logical tensor-size sum, and includes allocator rounding (approximately 1 MiB in the reference measurement).

| | |
|---|---|
| **Optimize** | Minimize long-prompt retained GPU allocation. |
| **Maintain quality** | Validation BPB $<$ 1.05 (fixed quality threshold) |
| **Stay within limits** | Request latency $\leq$**750 ms** |
| | Training FLOPs $\leq$**239.1 M/token** |
| | Total parameters $\leq$**50.3 M** |
| | Peak training memory $\leq$**44.0 GiB** |
| | Available training-data pool **Fixed 631.2 M-token pool** |

**Decode latency**

**Why it matters.** Lower decode latency produces generated tokens faster after the prompt has been processed.

**How we measure it.** Median latency for a one-token prefill followed by 512 decoding steps, measured over 30 timed passes after three warmups. The timer includes prefill and synchronized GPU completion, but excludes state allocation and reset.

| | |
|---|---|
| **Optimize** | Minimize latency. |
| **Maintain quality** | Validation BPB $<$ 1.05 (fixed quality threshold) |
| **Stay within limits** | Retained GPU state allocation at 513 tokens $\leq$**10 MiB** |
| | Training FLOPs $\leq$**239.1 M/token** |
| | Total parameters $\leq$**50.3 M** |
| | Peak training memory $\leq$**44.0 GiB** |
| | Available training-data pool **Fixed 631.2 M-token pool** |

**Long-prompt latency (request latency)**

**Why it matters.** Lower request latency reduces the full delay of processing a long context and generating a response.

**How we measure it.** Median latency for a 1,536-token prefill followed by 512 decoding steps, measured over 30 timed passes after three warmups. The timer includes prefill and synchronized GPU completion, but excludes state allocation and reset.

| | |
|---|---|
| **Optimize** | Minimize latency. |
| **Maintain quality** | Validation BPB $<$ 1.05 (fixed quality threshold) |
| **Stay within limits** | Retained GPU state allocation at 2,048 tokens $\leq$**40 MiB** |
| | Training FLOPs $\leq$**239.1 M/token** |
| | Total parameters $\leq$**50.3 M** |
| | Peak training memory $\leq$**44.0 GiB** |
| | Available training-data pool **Fixed 631.2 M-token pool** |

**Shared inference correctness requirements.** The three inference tasks supply ground-truth tokens at each decoding step and compare the resulting next-token distributions with the same model's full forward pass. For both the one-token and 1,536-token prefill settings, the maximum total-variation distance across all 513 scored positions must not exceed 0.05. The returned request-state object must own all retained KV data, including compressed representations or replacement token history, and these data must remain on the GPU throughout the request. Memory constraints use the same retained-allocation procedure as the KV-cache objective, with capacities of 513 and 2,048 for the short- and long-prompt settings, respectively.

# B Harness Integration and Reference Baselines

This section specifies the integration interface and the two reference search policies used in Section 4. Arena provides execution, measurement, and scoring; the baselines determine proposals and retention. Their selection rules are harness choices, not requirements imposed on other integrations.

## B.1 Interface and Recorded Artifacts

**Minimal interface.** A harness must inspect the provided workspace and task definition, modify only the permitted program, submit experiments through Arena, read standardized results, and choose subsequent candidates. Arena defines the research problem and evaluation environment while leaving the research strategy open: an integration may retain an incumbent, beam, population, hypothesis tree, or another search state.

**Recorded artifacts.** Each run records the reference program, proposed ideas, candidate implementations, execution outcomes, measured metrics, admissibility decisions, and search lineage. The terminal score is computed from measured admissible outcomes; the accompanying record supports audits and post-hoc trace analysis. Charged-experiment accounting follows Section B.4, independently of harness-side bookkeeping.

## B.2 Initialization

Arena prepares the environment and evaluates the reference program. The harness receives its source and measurements, the task's objective, BPB threshold, resource constraints, measurement procedures, experiment budget, and a workspace for candidate programs and research records.

Sequential Search starts from one incumbent and proposes one candidate per round. Beam Search retains up to four programs and proposes two modifications per retained program in each generation. A full generation therefore contains eight candidate experiments; selection occurs after all outcomes are available.

A persistent controller coordinates an ideator, executor, and reviewer. The implementation calls the reviewer an `evaluator`, distinct from Arena's evaluator. Agent sessions are fresh for each round or generation. Instructions, source code, task rules, and saved research history are supplied to each session, so continuation does not rely on conversational memory.

### B.3 Proposal Preparation

The ideator inspects the current code and prior experimental history, then records a proposal, rationale, timestamp, and position in that history. Edits may change architecture, attention, optimization, data processing, or training within the permitted boundary. A proposal can coordinate several edits implementing one mechanism; Sequential Search also requests follow-up experiments that could separate their contributions.

The executor implements each proposal in an isolated copy of its parent program and checks the edits before Arena's benchmark evaluation.

### B.4 Execution and Budget Accounting

Arena checks the submitted source, protected files, available launch budget, and hardware, then runs training and task-specific evaluation. It records source, measurements, execution status, and logs. Successful execution alone does not establish admissibility.

Sequential Search uses one worker; Beam Search uses four. Parallel submission does not alter accounting: each candidate is one experiment. A full eight-candidate generation needs eight available slots and consumes eight launches if all reach chargeable GPU work, although at most four run simultaneously. Local CPU checks do not replace candidate training through Arena.

The budget is 100 charged launches, including the reference evaluation. A recorded GPU-work-start marker determines charging. Once GPU work begins, crashes, out-of-memory errors, and task failures still consume budget. A resolved launch without the marker is not charged; an unresolved launch reserves a slot. A normally charged reference therefore leaves at most 99 candidate experiments. The harness reads Arena's remaining budget rather than counting its own rounds or generations. Failed and invalid experiments remain recorded.

### B.5 Review and Selection

The reviewer compares the proposal, parent and candidate code, measurements, and logs, then records feedback. Both baselines exclude failed or constraint-violating candidates from further development. Other harnesses may continue refining them, but only admissible results contribute to Arena's score.

**Unchanged-target attempts.** An efficiency improvement may increase BPB while remaining below the quality threshold. Both baselines also permit limited attempts to improve quality without changing the target. Target equality means exact equality of recorded values, regardless of the proposal's description. Such attempts consume the allowance even when rejected.

**Sequential Search.** An admissible candidate replaces the incumbent when it strictly improves the target. At exactly equal target value, it may replace the incumbent while the allowance remains if it improves either the task-defined tie-break metric or BPB. Otherwise the incumbent is retained. The allowance starts at four unchanged-target attempts and resets only after an accepted strict target improvement; accepting a quality improvement alone does not reset it.

**Beam Search.** Parents remain fixed throughout a generation. After every outcome is resolved, the reviewer pools parents and new candidates, removes exact program duplicates, and excludes failures, constraint violations, and missing required ranking measurements. Parents need not be replaced when their modifications fail.

Up to two unchanged-target attempts are permitted per generation only if the best retained target strictly improved during the preceding two completed generations. This allowance is determined before proposal generation. When permitted, a parent and modification with equal target are reduced to the admissible member with lower BPB. The remaining programs are ranked by target, ordered task tie-break metrics, and finally BPB, retaining up to four. Remaining ties favor lower complexity, lower resource use, and then an older program where evidence is available. A read-only helper computes identities, admissibility checks, and ranking values; the reviewer applies the selection policy.

### B.6 Persistence, Interruption, and Stopping

Saved state retains programs, measurements, decisions, feedback, and the remaining unchanged-target allowance, together with Beam generation membership and outcomes. Rejected and failed experiments remain in the history. Continuation reconciles outstanding requests with Arena and completes pending review before new proposals; recovery itself is not a new experiment.

Sequential Search normally stops when the budget is exhausted. Beam Search also stops when the remaining budget cannot cover two candidates per retained program; it does not shrink the generation or select from a partially evaluated one. Both record the stopping reason and unused budget.

Arena independently ranks all admissible recorded results and reports the best program, which need not be the final harness-selected endpoint. Computing this result requires no additional training.

## C Discovery Evidence and Supplementary Analysis

### C.1 Discovery Catalogue

This catalogue supports Section 4.2. Across the analyzed searches, it records 32 findings, including 20 accepted findings or variants, consolidated into nine representative idea families.

Table 7 gives before/after mechanisms, objectives, and comparisons with prior work. Its scope notes distinguish the main quality requirement from BPB $< 2.0$ and $< 1.02$ settings. Correspondences identify related research questions and technical differences, not necessarily exact rediscoveries. Endpoints combine multiple changes and do not isolate individual mechanisms' contributions.

**Table 7:** Nine findings from Auto-Research Arena: **six research findings** (a–f) and **three infrastructure engineering findings** (g–i). The latter address low-precision updates, activation memory, and decode execution in large-model infrastructure. Blue highlights searched changes.

**RESEARCH FINDINGS** Model and optimizer design (a–f)

### (a) Looped Transformers

Depth-specific residual gains

Parameters

**BEFORE**

**for** $\ell = 0, \ldots, 7$ :
$u \leftarrow h + \mathrm{Attn}_\ell(h)$
$h \leftarrow u + \mathrm{MLP}_\ell(u)$

**AFTER**

**for** $\ell = 0, \ldots, 7$ :
$u \leftarrow h + \alpha_\ell \, \mathrm{Attn}_{\ell \bmod B}(h)$
$h \leftarrow u + \beta_\ell \, \mathrm{MLP}_{\ell \bmod B}(u)$

**RELATION TO PRIOR WORK**

One block ($B = 1$), eight uses; each depth learns residual gains, unlike loop-count scaling (Wang et al., 2026).

### (b) Lookup-augmented FFNs

Token lookup before activation

Training FLOPs $\mathrm{MLP}_\ell(h)$

**BEFORE**

$z \leftarrow W_\ell^{\mathrm{up}} h$
**return** $W_\ell^{\mathrm{down}} \phi(z)$

**AFTER**

$e \leftarrow \mathrm{Gate}_\ell(h) \odot E_\ell[\tau]$
$z \leftarrow G_\ell^{\mathrm{up}} h + e$
**return** $G_\ell^{\mathrm{down}} \phi(z)$

**RELATION TO PRIOR WORK**

Grouped FFNs receive context-gated token-ID features before activation; Engram (Cheng et al., 2026) instead retrieves hashed $N$-grams.

### (c) Cross-layer K/V sharing

Layer-specific queries

Live-state memory

**BEFORE**

$\mathbf{K}, \mathbf{V} \leftarrow \mathrm{KV}_\ell(h)$
$y \leftarrow \mathrm{Attn}_\ell(h; \mathbf{K}, \mathbf{V})$

**AFTER**

**if** $\ell \in \mathcal{P}$ :
$\mathbf{K}, \mathbf{V} \leftarrow \mathrm{KV}_\ell(h)$
$y \leftarrow \mathrm{Attn}_\ell(h; \mathbf{K}, \mathbf{V})$

**RELATION TO PRIOR WORK**

Unequal CLA-style groups (Brandon et al., 2024) share K/V with private queries; LISA (Mu et al., 2024) instead shares and corrects attention weights.

### (d) Low-memory Adam

Token-row second moments

Training memory

**BEFORE**

$m_{ij} \leftarrow \mathrm{EMA}_1(g_{ij})$
$v_{ij} \leftarrow \mathrm{EMA}_2(g_{ij}^2)$

**AFTER**

$m_{ij} \leftarrow \mathrm{EMA}_1(g_{ij})$
$v_i \leftarrow \mathrm{EMA}_2\big(d^{-1} \sum_j g_{ij}^2\big)$

**RELATION TO PRIOR WORK**

Adam-mini-style (Zhang et al., 2025d) token-row second moments apply to input/value embeddings; first moments and output-head moments remain full.

### (e) Shared low-rank bases

Shared reads; private projections

Parameters

**BEFORE**

**for** $p \in \{q, k, v\}$ :
$y_p \leftarrow L_p R_p h$

**AFTER**

**for** $p \in \{q, k, v\}$ :
$y_p \leftarrow L_p R h$

**RELATION TO PRIOR WORK**

Learned Q/K/V read basis, private output factors; ASVD (Yuan et al., 2023) instead factorizes pretrained weights using activation statistics.

### (f) Unified vocabulary

Input / value / output

Parameters

**BEFORE**

$h_0 \leftarrow E^{\mathrm{in}}[\tau]$
$v_\ell^{\mathrm{tok}} \leftarrow E_\ell^{\mathrm{val}}[\tau]$
$o \leftarrow E^{\mathrm{out}} h$

**AFTER**

$h_0 \leftarrow E[\tau]$
$v_\ell^{\mathrm{tok}} \leftarrow E[\tau]$
$o \leftarrow E h$

**RELATION TO PRIOR WORK**

Input–output tying (Press & Wolf, 2016) extends to token-indexed value residuals: one table for all three roles.

**INFRASTRUCTURE ENGINEERING** Training and inference execution (g–i)

### (g) Stochastic rounding

BF16 parameter write-back

Update precision

**BEFORE**

$w \leftarrow \theta + \Delta\theta$
$\theta \leftarrow \mathrm{RN}_{\mathrm{BF16}}(w)$

**AFTER**

$w \leftarrow \theta + \Delta\theta$
$\theta \leftarrow \mathrm{SR}_{\mathrm{BF16}}(w)$

**RELATION TO PRIOR WORK**

BF16 storage is retained; stochastic write-back preserves small updates in expectation (Zamirai et al., 2020).

### (h) Activation reconstruction

Backward from retained outputs

Activation memory

**BEFORE**

$r \leftarrow \mathrm{ReLU}(z)$
$s \leftarrow r^2$; save$(r)$
$g_z \leftarrow 2r \odot g_s$

**AFTER**

$s \leftarrow \phi(z)$; save$(s)$
$g_z \leftarrow 2\sqrt{s} \odot g_s$

**RELATION TO PRIOR WORK**

Reuse retained outputs for backward; selective recomputation (Korthikanti et al., 2022) instead reruns operations to save memory.

### (i) Decode compilation regions

Gathers inside Q/K/V preparation

Decode latency

**BEFORE**

$e \leftarrow E[\tau]$
$c, s \leftarrow T[t]$
$q, k, v \leftarrow \mathrm{In}(h, e, c, s)$

**AFTER**

$F \leftarrow \mathrm{Compile}(\mathrm{Prep})$
$q, k, v \leftarrow F(h, \tau, t)$

**RELATION TO PRIOR WORK**

Execution-overhead optimization, as in DeepSpeed Inference (Aminabadi et al., 2022); the baseline already has CUDA graphs.

**Scope.** (a,e) Beam Search and (f) Sequential Search, BPB $< 2.0$; (b–d,g,h) Sequential Search and (i) Beam Search , BPB $< 1.02$

**Notation.** $h$: hidden state; $\ell$: depth; $B$: stored blocks; $\tau$: token ID; $\phi(z) = \mathrm{ReLU}(z)^2$; $\mathcal{P}$: K/V producer layers. $\mathrm{Attn}_\ell$ and $\mathrm{MLP}_\ell$ are full modules; supplied K/V retain layer-specific query/output projections. $G$: grouped projections; $E_\ell$: table assigned to depth $\ell$; $\mathrm{Gate}_\ell$: broadcast gate. In (d), $g_{ij}$ is an embedding gradient with row width $d$; $\mathrm{EMA}_{1,2}$ use Adam decay rates. In (f), $E$ stores token vectors by row; $o$: logits. In (g), $\Delta\theta$: computed update; RN/SR: nearest/stochastic rounding. In (h), $g_x = \partial\mathcal{L}/\partial x$. In (i), $T[t]$: rotary factors at position $t$; Prep includes both gathers and Q/K/V preparation In. Normalization, other auxiliary paths, and decode-state reconstruction are omitted.

### C.2 Supplementary Protocols for Research-Process Studies

#### C.2.1 Budget Allocation

Four parameter-minimization searches use BPB $< 2.0$ and candidate training caps of 1,200, 2,400, 4,800, and 6,000 seconds. Obtaining 60, 30, 15, and 12 completed candidates, respectively, gives 20 nominal GPU-hours per setting (Table 8). The searches share starting code, a 379,260,514-token data pool, seed 42, A100 80GB hardware, tokenizer, and held-out evaluator. Beam width four, two proposals per parent, two retuning slots, and four GPU workers are fixed.

Table 8: **Searching results with varying training time.** We compare Beam Search for parameter minimization with different training times per candidate under a fixed 20 GPU-hour budget. Shorter evaluations allow more candidates to be explored. The shortest setting finds the smallest model satisfying BPB $< 2.0$, while longer evaluations achieve better BPB with larger models.

| Training time/candidate (s) | Completed candidates | Parameters | BPB |
|---|---|---|---|
| 1,200 | 60 | 123,272 | 1.986241 |
| 2,400 | 30 | 3,194,960 | 1.201839 |
| 4,800 | 15 | 11,542,672 | 1.060628 |
| 6,000 | 12 | 7,864,464 | 1.151816 |

These retrospective prefixes vary duration, candidate count, explored programs, and achieved quality together; they establish neither an optimal allocation nor a causal or matched-quality effect.

#### C.2.2 Cross-Target Transfer Protocol

Table 6 reports parameter-to-FLOP, FLOP-to-uncached-decode-latency, and parameter-to-training-time continuations. Program code is transferred; weights are freshly initialized. Each receiving evaluator remeasures the transferred program under its own objective and constraints. Earlier gains are not imposed as hard constraints.

There is one search per receiving objective. These selected examples demonstrate compatible improvements and permitted trade-offs; without matched restarts from the reference program, they establish neither a causal benefit of transfer nor a transfer-success rate.

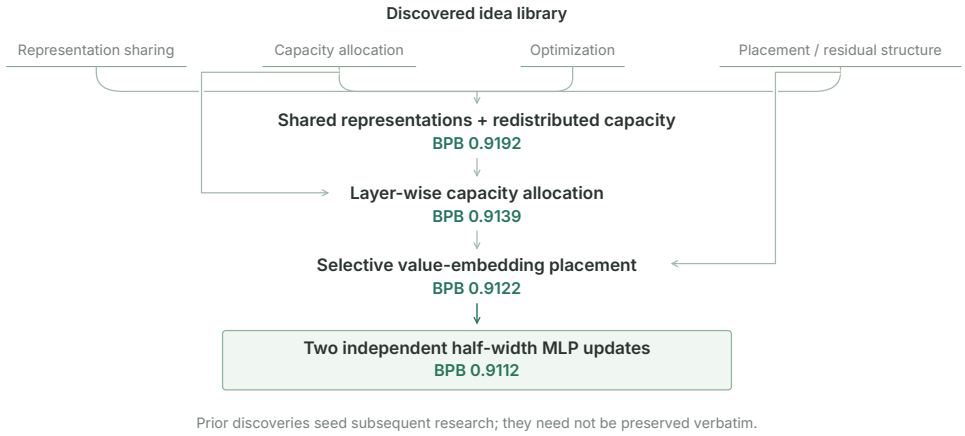



Figure 5: **The selected design emerges through reuse and reinterpretation of prior discoveries.** The quality-driven search builds on mechanisms for representation sharing and capacity allocation, subsequently changing where these mechanisms act and ultimately reinterpreting the MLP update as two independent half-width sublayers. Rather than simply accumulating prior discoveries, the search selects, recombines, and modifies them into new designs.

### C.3 How Discovered Ideas Are Reused

Section 4.3 shows that ideas discovered in Arena's short, target-specific searches can be reused to improve model quality at a larger training scale. Here we examine how this reuse occurs. Rather than transferring a single winning program or mechanically combining all previous discoveries, the quality-driven search receives an idea library and can select, combine, modify, or discard its contents. We analyze the provenance of the selected design, representative interactions encountered during search, and how the candidate distribution evolves across search rounds.

**The search draws on discoveries across objectives and harnesses.** The recorded sources represented in the selected design span four research harnesses and four classes of Arena objectives: parameter efficiency, training tokens, training time, and training updates. They include mechanisms for sharing representations, allocating MLP capacity, changing optimization schedules, gating attention and value residuals, restricting context, and changing where computations are applied across depth. The resulting search is therefore not a continuation of one original objective or one harness trajectory. Instead, discoveries from heterogeneous searches form a common library of candidate research ingredients.

**Reuse progressively transforms prior mechanisms.** Figure 5 summarizes the main architectural progression leading to the selected design. An early improvement shares the value-embedding table and reallocates MLP capacity, reducing BPB from the reference value of 0.9295 to 0.9192. Subsequent search changes how capacity is distributed across depth and where value embeddings are injected, eventually reaching 0.9122. The final step changes a different aspect of the architecture: instead of one MLP update per block, the model applies two independent half-width MLP sublayers, reaching 0.9112 BPB. Thus, the selected design is not a direct union of library entries. The search progressively reinterprets how previously discovered mechanisms are implemented and combined.

**Idea provenance spans multiple research histories.** Table 9 lists the ten recorded idea sources represented in the selected search lineage. They originate from Sequential Search, GEAR, Auto-Scientists, and Heuresis, and from searches targeting parameters, training tokens, training time, and training updates. This diversity supports the intended role of the idea library: mechanisms discovered under one research objective can become hypotheses for subsequent research under another. At the same time, provenance only records where an idea entered the search; it does not establish that every listed mechanism independently contributes to the final improvement.

**Useful ideas do not combine monotonically.** Representative outcomes in Table 10 reveal both positive and negative interactions. For example, simply reapplying an existing MLP adds another transformation without introducing independent parameters and performs worse than the reference, whereas

Table 9: **Recorded idea sources represented in the selected search lineage.** The quality-driven search draws on mechanisms originally discovered by four research harnesses under four classes of Arena objectives. The provenance records indicate where an idea entered the library, not an independent causal contribution to the final model.

| Recorded idea | Source harness | Objective | Use in the quality-driven search |
|---|---|---|---|
| Shared value-embedding table | Sequential Search | Parameters | Shares token-dependent value representations rather than maintaining separate lookup tables. |
| MLP width as a search variable | GEAR | Parameters | Reallocates model capacity through MLP width rather than uniformly shrinking the architecture. |
| Reduced weight decay | GEAR | Training tokens | Changes optimization while retaining the architectural backbone. |
| Square-root cooldown | Heuresis | Training time | Changes the learning-rate schedule during later training. |
| Per-head attention output gate | Sequential Search | Training time | Introduces learned control over individual attention-head outputs. |
| Sparse value-residual gate channels | AutoScientists | Parameters | Restricts value-residual gating to a subset of channels. |
| Shorter attention window | GEAR | Training updates | Changes the amount of context processed by attention. |
| Layer-specific MLP widths | AutoScientists | Parameters | Redistributes MLP capacity across depth while approximately preserving total width. |
| Value embeddings at selected stack depths | Sequential Search | Training time | Moves value-residual injection from every layer to selected depths. |
| Two independent half-width MLP sublayers | Heuresis | Training tokens | Motivates a final design with two independent MLP updates within each block. |

Table 10: **Idea reuse is selective rather than additive.** Representative outcomes from the quality-driven search show that plausible combinations can help or hurt. These comparisons arise from the search process and are not controlled ablations.

| Representative design | BPB ↓ |
|---|---|
| Fixed reference | 0.929497 |
| Shared value embedding + redistributed MLP capacity | 0.919165 |
| Reapply the same MLP without introducing independent capacity | 0.934888 |
| Redistribute approximately the same total MLP width across depth | 0.913908 |
| Inject value embeddings only at selected stack depths | 0.912195 |
| Further tune the selective-injection design | 0.911221 |
| Two independent half-width MLP sublayers | **0.911213** |

introducing a genuinely independent second MLP appears in the strongest final design. Other search outcomes show that individually strong modifications can lose their advantage when combined and that different allocations of approximately the same total MLP width can yield different quality. These observations are not controlled ablations, but they show why the idea library should not be viewed as a collection of independently additive improvements. Its value instead lies in providing structured hypotheses that subsequent research can test, reject, and reinterpret.

**Search progressively moves into a stronger design region.** Figure 6 shows the distribution of candidate quality across the six search rounds. None of the 56 valid candidates in the first round outperforms the fixed reference, compared with 44/56, 29/56, 50/56, 55/56, and 25/25 in subsequent rounds. The temporary broadening in the third round shows that the search does not simply accumulate accepted improvements, while later rounds increasingly concentrate around substantially lower BPB. Across all rounds, 203 of 305 valid candidates outperform the reference, indicating that the improvement reflects a broad region of stronger designs rather than a single isolated candidate.

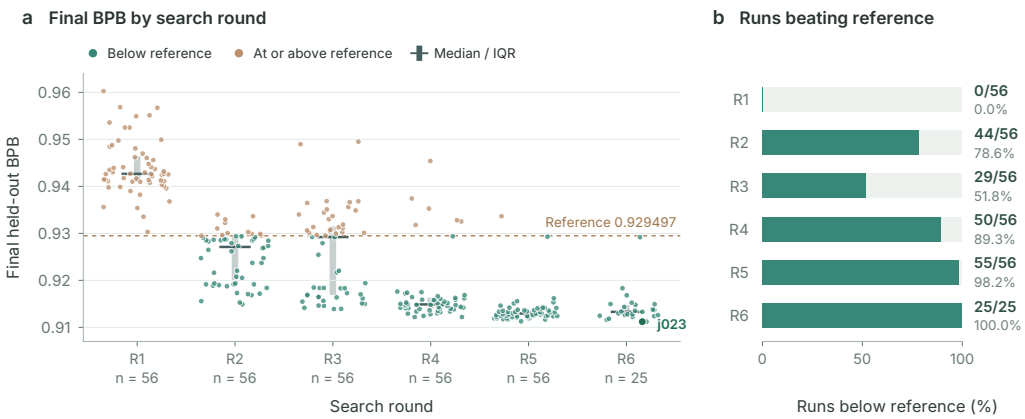



Figure 6: **The quality-driven search progressively moves into a stronger region.** Each point shows the final held-out BPB of one valid candidate, grouped by search round; horizontal jitter is used only to avoid overlap. Green points outperform the fixed reference (dashed line), while gray bars show the median and interquartile range within each round. The distribution shifts substantially below the reference after the first round and becomes increasingly concentrated at lower BPB in later rounds. Overall, 203 of 305 valid candidates outperform the reference.

**The exact top-ranked combination should not be overinterpreted.** Although the search moves toward stronger designs overall, its best candidates are extremely close. The two best recorded candidates achieve BPB values of 0.911213 and 0.911221, differing by only $8 \times 10^{-6}$ in their one-hour evaluations. With one measurement per candidate, differences at this scale do not establish that either design is uniquely superior to nearby alternatives. We therefore interpret the experiment as evidence that the discovered idea library guides subsequent research toward a productive region of model designs, rather than that one particular combination is uniquely optimal.

## D Elo Scores

Table 2 reports the best recorded admissible target values across nine objectives for Opus 5 and GPT-6. We aggregate these values using the same margin-aware Elo procedure, with equal weight assigned to each objective.

Training times are compared after rounding to the nearest 30-second checkpoint; matching checkpoints are ties. Ratings are calculated from these stored target values before display rounding or unit conversion. The two agent-model groups use separate method pools, each with a mean rating of 1500, so Elo comparisons are made within each group.

All harnesses begin each calculation with the initial rating $R = 1500$. Lower values are better for every included objective.

**Turn best admissible results into pairwise outcomes**

$$y_{a,b,j} = \begin{cases} 1, & x_{a,j} < x_{b,j} \\ \frac{1}{2}, & x_{a,j} = x_{b,j} \\ 0, & x_{a,j} > x_{b,j} \end{cases}$$

$x_{a,j}$ is harness $a$'s best recorded admissible target value on objective $j$. $y_{a,b,j}$ is the outcome when harness $a$ is compared with harness $b$: 1 for a win, $\frac{1}{2}$ for a tie, and 0 for a loss.

**Estimate the expected outcome from the current ratings**

$$E_{a,b} = \frac{1}{1 + 10^{(R_b - R_a)/400}}$$

$R_a$ and $R_b$ are the harnesses' current ratings. $E_{a,b}$ is the outcome expected for harness $a$, between 0 and 1. A higher-rated harness is expected to win more often.

**Measure how large the win is**

$$d_{a,b,j} = \log\left(\frac{x_{b,j}}{x_{a,j}}\right), \qquad z_{a,b,j} = \frac{|d_{a,b,j}|}{s_j}, \qquad m_{a,b,j} = \min\left(3, \frac{\log(1 + z_{a,b,j})}{\log 2}\right)$$

$d_{a,b,j}$ is the logarithm of the ratio between the two target values. Using a ratio makes proportional improvements comparable within an objective. $s_j$ is the median nonzero absolute log margin between all harness pairs on objective $j$, bounded below by $10^{-9}$; it defaults to 1 when every pair is tied. It defines a typical margin for that objective. $z_{a,b,j}$ expresses the observed margin in units of that typical margin. The margin multiplier $m_{a,b,j}$ grows gradually with the margin and is capped at 3 so that one unusually large win cannot dominate the ranking. Exact ties have a zero multiplier and cause no rating update.

**Weight objectives and update both ratings**

$$\Delta R_a = K \, \frac{w_j}{\bar{w}} \, m_{a,b,j}\big(y_{a,b,j} - E_{a,b}\big), \qquad R_a \leftarrow R_a + \Delta R_a, \qquad R_b \leftarrow R_b - \Delta R_a$$

$K$ is 32, which controls the size of each update. $w_j$ is objective $j$'s weight, and $w^-$ is the average objective weight. Dividing by the average keeps an ordinary comparison near the standard Elo update size. The term $y - E$ rewards results that exceed the current expectation and penalizes results that fall short.

**Reduce sensitivity to comparison order**

Because Elo updates happen one comparison at a time, their order can slightly affect the result. The calculator therefore generates 4,000 permutations using NumPy's default random generator with seed 20260902. Each pass starts all harnesses at 1500 and processes every pair once per objective; the reported score is the mean final rating across passes. The zero-sum updates preserve a cohort mean of 1500 for both agent-model groups, without a final shift or fixed maximum. The published score is a relative rating of the best recorded admissible results, not a percentage or a measurement in any resource unit.

# E Extended Related Work

## E.1 Autonomous-Research Benchmarks

**Implementation, reproduction, and research tasks.** Benchmarks cover repository execution, method implementation, reproduction, and expert-specified extensions (Bogin et al., 2024; Siegel et al., 2025; Hua et al., 2025; Yan et al., 2025; Starace et al., 2025; Kon et al., 2026; Edwards et al., 2026). Related capability suites assess software and command-line tasks (Jimenez et al., 2024; Merrill et al., 2026), retrieval (Wei et al., 2025; Xiong et al., 2026), scientific discovery and programming (Majumder et al., 2025; Chen et al., 2025b; Tian et al., 2024), and theorem proving (Tsoukalas et al., 2024). Arena instead leaves the improvement mechanism open and evaluates a measured optimization target under quality and resource constraints.

**Open-ended research and research artifacts.** Machine-learning engineering, research competitions, and paper-derived research environments support open-ended experimentation under task-specific evaluation protocols (Chan et al., 2025; Miller et al., 2025; Wijk et al., 2025; Nathani et al., 2025; Garikaparthi et al., 2026; Lupidi et al., 2026; Wu et al., 2026; Zhang et al., 2025c). Other evaluations examine proposals, experiments, reports, alignment, and artifact support across research stages (Tang et al., 2025; Liu et al., 2026; Chen et al., 2025a; Xu et al., 2026; Cui et al., 2026; Zhang et al., 2026; Bragg et al., 2026). Arena focuses on how harness performance changes across objectives in one language-modeling environment, ranking measured endpoints and using traces as explanatory evidence.

**Language-model and computational optimization.** Language-model benchmarks evaluate validation loss, training speed, reproduction of historical improvements, or post-training under compute constraints (Karpathy, 2026; Intology, 2026; Zhao et al., 2025; Rank et al., 2026); systems benchmarks target kernels, numerical routines, and software performance under correctness requirements (Ouyang et al., 2025; Press et al., 2025; Shetty et al., 2025). Arena spans nine model, training, and inference objectives with a common quality requirement and objective-specific resource limits.

**Evaluation validity and reliability.** Studies examine efficiency and scaffold contributions after benchmark saturation (Nadgir et al., 2026), specification and evaluator inconsistencies (Tu et al., 2026), and hardware sensitivity and score aggregation (Chen et al., 2026). These concerns motivate distinguishing valid resource improvements from apparent gains caused by evaluation artifacts.

### E.2 Autonomous-Research Harnesses

**End-to-end workflows.** Research systems connect literature review, idea generation, implementation, experiments, analysis, and reporting (Lu et al., 2026; Tang et al., 2025; Trirat et al., 2025; Schmidgall et al., 2025). The research harness orchestrates these activities around a foundation model.

**Tree, archive, and evolutionary search.** Harnesses allocate experiments through code trees, hypothesis trees, archives, and alternative selection policies (Jiang et al., 2025; Toledo et al., 2025; Jin et al., 2026; Antoniades et al., 2026). Population-based methods retain competing programs or heuristics and generate candidates using executable feedback, mutation, or crossover (Romera-Paredes et al., 2024; Liu et al., 2024; Novikov et al., 2025; Jeddi et al., 2026). These search procedures organize exploration; performance gains alone do not establish scientific novelty.

**Multi-agent coordination and persistent memory.** Teams, proposal critique, execution recovery, and shared successful or failed experiments support sustained research (Gao et al., 2026; Liu et al., 2026). Semantic, episodic, and structured scientific memory provide complementary mechanisms for retaining knowledge across sessions (Fang et al., 2025; Gottweis et al., 2026; Mitchener et al., 2025).

**Harness design and self-improvement.** Other methods optimize agent architectures, workflows, or implementations through meta-search and recursive improvement (Hu et al., 2025; Zhang et al., 2025b;a; Srikanth et al., 2026). Their search object is the research procedure itself, rather than only the candidate experimental program.

Arena evaluates harness–model combinations across multiple optimization objectives in a shared language-modeling environment, leaving the search strategy open and retaining standardized experimental traces.