Alder-1.6B: training a 1.59B reasoning and tool-use language model from scratch in the open for $980 all-in

Udgam Labs (udgamlabs.com), founded by Jitesh K Prasad. Technical report for Alder-1.6B and Alder-1.6B-Base, version 1.0.0, released 28 November 2026.

udgamlabs.com

Abstract

Alder-1.6B is a 1.59B-parameter dense English language model trained from random initialisation in the open by Udgam Labs under Apache-2.0. Across eight stages — base pre-training (25.4B tokens), staged long-context extension (8K to 32K native, 64K stretch), 32K supervised fine-tuning (SFT), verifiable reinforcement learning (GRPO), two-stage preference and identity alignment (micro-SFT + DPO), benchmark evaluation, and lossless native open-weights export — direct model training and final evaluation cost **$702.96** across 34 GPU runs. Counting every initial architecture ablation, 4-way data recipe screen, corpus build, mid-run probe, rival check, and failed rehearsal from day one adds **$277.32** across 139 runs, bringing the **all-in project total to $980.28** across all 173 runs ($19.72 under our $1,000 target).

Every weight, all 41 intermediate pre-training and post-training checkpoints, GGUF and MLX quantisations, evaluation transcripts, spend logs, and training scripts are released publicly under Apache-2.0:

- **Native compatibility without custom code.** Alder is mathematically converted from our training architecture to stock `Qwen3ForCausalLM` (maximum absolute fp32 logit difference `1.43e-4`, 100% identical argmax across all 24 layers), requiring zero `trust_remote_code` in `transformers`, `vLLM`, `llama.cpp`, Ollama, LM Studio, and MLX. On Apple Silicon, the Q8_0 GGUF (1.6 GB) runs at 79 tokens/s and the MLX 8-bit build at 85.8 tokens/s.
- **Public multiple-choice benchmarks (32,169 test items, Base to Final).** Five of six standard benchmarks improved over the finished base model after long-context, SFT, RL, and preference tuning: ARC-Easy norm `58.16% -> 60.40%` (`65.15%` raw), ARC-Challenge norm `33.28% -> 35.58%`, HellaSwag norm `51.46% -> 51.53%`, WinoGrande `56.75% -> 56.99%`, and MMLU 5-shot `24.34% -> 26.38%`. BoolQ fell `62.63% -> 59.02%` (-3.61 pp).
- **Verifiable skills and honest abstention (904-item held-out suite).** Reinforcement learning and two-stage DPO lifted mean reward 4.6x (`+0.085 -> +0.389`) with zero regression after DPO, drove exact JSON tool calling to `95.7%` (`0%` malformed), passage-grounded QA to `86.0%` (from `53.0%` in SFT), calculator-backed arithmetic to `73.0%` (from `59.0%`), and honest decline accuracy on unanswerable prompts from `30.2%` to `59.1%` (`93.0%` on missing-fact passages, `86.6%` when no tool applies, `50.0%` on missing-input calculations).
- **32K long-context retention.** After full SFT, RL, and DPO, single-needle recall in chat mode averages `84.0%` across 4K to 32K (`4K: 80%`, `8K: 100%`, `16K: 60%`, `24K: 100%`, `32K: 80%`).
- **Honest limits.** Alder is not competitive in absolute capability with frontier 1.5B-1.7B models such as SmolLM2-1.7B (11T tokens, 433x Alder's pre-training tokens) or Qwen2.5-1.5B (18T tokens, 709x). Pure-text GSM8K without a calculator is `3.34%` (1,319 test items, 0-shot thinking mode, `96.5%` closed thinking tags), greedy open-ended prose can occasionally repeat paragraphs, and 1 of 30 unseen identity prompts followed a direct verbatim prompt-injection jailbreak.

1. Motivation and goals

On 13 November 2026 we set a fixed engineering goal: **build the most useful general-purpose reasoning and tool-use language model possible from random initialisation on a $1,000 compute budget, and document every step — including failures — in the open.**

Modern 1.5B-1.7B models from large labs train on 11 trillion to 36 trillion tokens, often distilled from 70B+ teacher models, at six- to seven-figure compute cost. That produces impressive general-knowledge benchmarks, but leaves a wide gap in open documentation for independent labs working at $1,000 scale: where should you spend each dollar across pre-training, context extension, SFT, RL, and preference tuning? Which architecture tricks actually beat a clean baseline? What breaks during rehearsals, packing, RL evaluation, and DPO alignment — and how do you fix each failure cheaply?

Alder answers those questions with complete transparency. Rather than chasing leaderboard memorisation, we designed Alder as a **reasoning core** focused on four practical behaviours:

1. **Follow structured chat and thinking formats** cleanly in both direct (`no_think`) and step-by-step (`think`) modes.

2. **Read long inputs up to 32,768 tokens** without forgetting base language fluency.
3. **Call external tools in exact JSON** and ground answers strictly in supplied passages.
4. **Decline honestly when information or tools are missing**, instead of hallucinating plausible answers.

Alder is named after the alder tree, a pioneer species that grows on bare ground and fixes nitrogen so other trees can grow after it.

2. Public artifacts and releases

Every Alder weight, intermediate checkpoint, quantisation, script, and report is published under **Apache-2.0**:

Artifact	Location	Contents
Final instruct model	huggingface.co/udgamlabs/alder-1.6b	Native `Qwen3ForCausalLM` `bf16` `safetensors` (3.0 GB), tokenizer, `chat_template.jinja`, Q8_0 & Q4_K_M GGUFs
Finished base model	huggingface.co/udgamlabs/alder-1.6b-base	Base `step_012100` (25.4B tokens, 2K context), native `Qwen3ForCausalLM` `bf16` `safetensors`
GGUF quantisation suite	huggingface.co/udgamlabs/alder-1.6b-GGUF	`F16` (3.0 GB), `Q8_0` (1.6 GB), `Q6_K` (1.2 GB), `Q5_K_M` (1.1 GB), `Q4_K_M` (931 MB), `Q3_K_M` (764 MB), Ollama `Modelfile`
Apple Silicon MLX builds	alder-1.6b-mlx-8bit & alder-1.6b-mlx-4bit	8-bit (`1.6 GB`, 85.8 tok/s) and 4-bit (`854 MB`) native MLX builds
Full checkpoint trajectory	huggingface.co/udgamlabs/alder-1.6b-checkpoints	All 33 pre-training checkpoints (Part 1 + Part 2) and all 8 post-training stage checkpoints as native `Qwen3ForCausalLM`
Code, evals & spend ledger	github.com/udgamlabs/alder	Pre-training, long-context, SFT, RL, DPO, export converter, benchmark harness, `CHAT.md`, and line-by-line `SPEND.md`
Live training record	udgamlabs.com/live/alder	Public real-time build log and per-stage charts

3. Architecture and lossless Qwen3 export

Alder is a 1.59B-parameter dense decoder-only Transformer trained with an export-clean configuration (`udgam-v1`):

Parameter	Value
Total parameters	1,590,731,520 (1.59B)
Layers (`n_layer`)	24
Hidden size (`n_embd`)	2,304
Attention heads (`n_head` / `n_kv_head`)	18 query / 6 key-value (3:1 GQA), `head_dim = 128`
Attention pattern	Full causal attention on all 24 layers (no sliding windows)
Feed-forward network	SwiGLU, intermediate width 6,144
Normalisation	RMSNorm (`eps = 1e-6`) + QK-Norm (`qk_scale = 1.2`)
Position encoding	Rotary (RoPE): base `10,000` in 2K base, `2,000,000` from 32K long-context onward
Context length	2,048 (base) -> 32,768 native (final)
Vocabulary	`gpt2-ctl16`: GPT-2 BPE (50,257) + 16 Alder control tokens = 50,273 (padded to 50,304 during GPU matmuls, cropped cleanly to 50,273 at export)
Embeddings	Untied input embedding table and output `lm_head`
Disabled extras	No logit soft-cap, no smear gate, no mid-layer backout, no x0 residual blend, no value embeddings, no attention gate

Why we stripped non-standard architectural extras. During our early 14-16 November ablation campaign (Section 7.1), five proposed modifications failed to beat the clean control architecture. Stripping smear, backout, x0-blending, value embeddings, and logit soft-capping had a major downstream payoff: it made Alder mathematically isomorphic to stock `Qwen3ForCausalLM`.

Lossless mathematical conversion to `Qwen3ForCausalLM`. Our training code differed from standard Hugging Face `Qwen3ForCausalLM` in four exact ways, each resolved analytically in `export/export_hf.py` with zero retraining:

1. **Pre-embedding RMSNorm:** Training applied parameterless `RMSNorm(wte(idx))` before layer 0. We fold this row-wise into `model.embed_tokens.weight`.
2. **Per-layer positive residual multipliers (`resid_lambdas[i] > 0`):** Defining cumulative products `P_0 = 1`, `P_{i+1} = lambda_i * P_i`, scale invariance of RMSNorm (`RMSNorm(c * x) = RMSNorm(x)` for `c > 0`) lets us divide `o_proj.weight` and `down_proj.weight` at layer `i` by `P_{i+1}`, turning `x_{i+1} = lambda_i * x_i + attn + mlp` into standard unit-residual `z_{i+1} = z_i + attn' + mlp'` with identical final `RMSNorm(x_L) = RMSNorm(z_L)`.
3. **Conjugate RoPE sign convention:** Our trainer rotated with `−sin`, whereas Hugging Face `Qwen3ForCausalLM` uses `+sin`. Negating the second half of every head (`d/2 .. d−1`) in `q_proj.weight` and `k_proj.weight` flips `x_2 -> −x_2` identically in every `Q . K` inner product.
4. **QK-Norm scaling:** Because RoPE is orthogonal, `RoPE(RMSNorm(q) * 1.2) = RMSNorm(RoPE(q)) * 1.2`, which maps directly to `q_norm.weight = k_norm.weight = 1.2`.

Verified on the real final checkpoint (`dpo2 step_000141`) in fp32, the maximum absolute logit difference between our custom trainer `GPT` and stock `Qwen3ForCausalLM` across all 24 layers is **`1.43e−4`** with **100% identical argmax** on every token position. Tokenisation parity between `posttrain/common/chat_format.py` and the exported Jinja `chat_template.jinja` is 100% token-ID identical in both `no_think` and `think` modes.

4. Training pipeline: all eight stages

All post-training stages grew linearly from the single finished base checkpoint (`step_012100`, 25.4B tokens).

Steps 1 and 2: Base pre-training (25.4B tokens, $549.86)

- **Step 1 (Trunk, 20.9B tokens, $444.23):** 9,989 updates at 2,048 context on our winning pre-training mix (chosen from a 4-way recipe screen, Section 7.2). Held-out 5-subject panel (prose, science, maths, code, tool use) finished at `0.7716` bpb (`0.802` overall held-out curve). Eight milestones were kept.
- **Step 2 (Cooldown, +4.4B tokens, $105.63):** Continued from step 9,989 to step 12,100 while annealing the learning rate to zero and introducing the third-stage curriculum (richer tool-use and practice-program text). The 5-subject held-out panel dropped sharply from **`0.7716 -> 0.6720` bpb**. On our frozen 14-tape 3-shot base check (`aldereval1`), Part 2 lifted right answers more than 4x (`104 -> 458` of 4,261, beating Laghu-Base and Gargi-Base by 2.2x).

Step 3: First supervised fine-tuning pass (`sft1`, 2K context, $6.37)

A fast 2K-context SFT run (`sftdata1`, 143.8M train tokens, 111 updates) from `step_012100` to prove the chat, thinking, tool-calling, and decline token contract (`<|im_start|>`, `<|im_end|>`, `<|think|>`, `</think>`, `<|think_off|>`, `<|tool_call|>`, `</tool_call>`, `<|tool_response|>`, `</tool_response>`, `<|no_answer|>`). Held-out answer loss dropped 26.0% (`0.8667 -> 0.6414` nats) and format checks climbed from `0/11` to `10/11`, while exposing three critical bugs (forgetting, packing leaks, and step-0 eval timing) that we fixed before Step 5.

Step 4: Staged long-context extension (`lc1`: 8K -> 32K -> 64K, $79.04)

From `step_012100`, we built a 1.80B-token long-document corpus (`lcdata1`, $1.55) and extended RoPE base from `10,000` to `2,000,000` across 700 updates (734M tokens) in three stages:

- `lc1_8k`: 200 updates at 8,192 context (281K tok/s).
- `lc2_32k`: 340 updates at 32,768 context (204K tok/s). Held **+0.0008 bpb** forgetting on the original 2K base panel (`0.6720 -> 0.6728`, with science and tools slightly improved), achieved **100% single-needle recall at 4K, 8K, 16K, and 24K** (`80%` at 32K), and `100%` 4-needle recall through 16K (`50%` at 32K). Held-out bpb improved monotonically out to 49K tokens (`0.816 -> 0.616`).
- `lc3_64k`: 160 updates at 64K context. Retained long-range fluency but suffered `+0.0454 bpb` forgetting on the 2K base panel (`+0.100` on tools). We therefore selected **`lc2_32k`** as the base for Step 5 and archived both checkpoints.

Step 5: 32K long-context supervised fine-tuning (sft2, $35.18)

From lc2_32k, we built a 127.8M-token 32K reasoning, tool, and long-document retrieval/decline SFT mix (sftdata2, $0.05) and trained 518 updates (271.6M tokens at 32K context, 205.6K tok/s, gentler peak LR, and exact cu_seqlens document-boundary attention masking):

- Held-out 32K answer loss fell **42.4%** (1.1944 -> 0.6877), control-token loss fell 15.70 -> 0.167.
- **Zero packing leaks** across 24 paired leak tests (0/24 mode flips, 0/24 tool leaks, delta_loss = 1.7e-5).
- 32K single-needle recall climbed from **80% -> 90%**; 32K 4-needle recall climbed **50% -> 62.5%**.
- 2K base-panel shift stayed bounded at +0.0238 bpb (0.6966).

Step 6: Verifiable reinforcement learning (rl1, 400 GRPO steps, $25.55)

Starting from sft2 step_000518, we trained 400 steps of Dr. GRPO / DAPO-style verifiable RL (32 prompts x 8 rollouts per step, asymmetric PPO clip 0.20 / 0.28, KL 0.01 to start weights, lr = 1e-6, zero execution of model-written code) across 23,474 training tasks and 904 held-out tasks in 8 families (answerable + unanswerable twins: tool_call/tool_none, passage_answer/passage_missing, calc/calc_missing, gsm8k/gsm8k_twin).

Full 904-item held-out evaluation (step 0 -> step 400):

- Mean reward rose **4.6x** (+0.085 -> +0.389), equal-weighted 4-family accuracy jumped **+21.0 pp** (50.8% -> 71.8%).
- passage_answer: 53.0% -> 97.0% (+44 / -0 paired wins); passage_missing: 42.0% -> 96.0% (+54 / -0).
- calc: 59.0% -> 87.0% (+29 / -1); tool_call: 93.5% -> 96.8%; tool_none: 78.4% -> 87.6%; calc_missing: 0.0% -> 7.0%.
- Malformed replies dropped **5.1x** (14.4% -> 2.8%), false declines halved (0.8% -> 0.4%), and the SFT forgetting probe shifted by only **+0.0003 nats** (0.5893 -> 0.5896 — essentially zero forgetting).
- gsm8k barely moved (5.9% -> 6.4%, +5 / -4), confirming that 400 RL steps cannot create multi-step pure-text math ability that pre-training did not establish.

Step 7: Preference tuning and identity alignment (dpo1 + dpo2, $5.39)

After RL, Alder still carried open-dataset identity priors ("Open Assistant" / third-party lab names) and had a 53.3% valid-format rate on open identity chat prompts.

- **First attempt (dpo1, pure DPO + 0.25 NLL anchor, $3.01) — FAILED:** Improved format (53.3% -> 73.3%) and "trained from scratch" recall (42.9% -> 71.4%), but achieved **0.0% Udgam Labs attribution** and raised wrong-lab claims (36.7% -> 53.3%). Root cause: pure DPO saturated its sigmoid margins (+25 to +37) before lifting near-zero-prior proper nouns (Udgam Labs, Alder, Jitesh K Prasad).
- **Second attempt (dpo2, two-stage micro-SFT + light DPO, $2.38) — PASSED:** Stage A ran 75 pure-NLL micro-SFT steps (lr = 1.5e-5) mixing 561 Udgam Labs identity examples, 130 calibration examples, and 500 general chat replay rows; Stage B ran 66 light DPO steps (lr = 1e-6, beta = 0.1, nll_coef = 0.15).
  - Unseen 30-prompt identity gate (no_think + think): format_ok **53.3% -> 100.0% (30/30)**, Udgam Labs attribution **3.3% -> 96.7% (29/30; 29/29 on all non-jailbreak prompts)**, 0 wrong founder names.
  - On-box 160-item stratified rl1 retention gate: mean reward **rose** (+0.4656 -> +0.5156) and decline accuracy jumped **51.25% -> 65.00%** (kl_w = 0.0547).

Step 8: Full benchmark evaluation and native export (aldereval3, $1.57 + $0 local)

Final head-to-head evaluation comparing alder-1.6b-base (step_012100) against alder-1.6b (dpo2 step_000141) across 32,169 public multiple-choice items, 1,319 GSM8K test problems, all 904 RL held-out items, 4K-32K single-needle recall, and 36 open-ended founder chat prompts (CHAT.md), followed by local conversion to native Qwen3ForCausalLM, 6 GGUF quantisations, and MLX 4-bit/8-bit.

5. Complete evaluation results

All public benchmarks were evaluated with our open harness (`eval/aldereval3/eval_bench.py`) using exact log-likelihood scoring for multiple-choice tasks and greedy (`temperature = 0`) generation with FlashAttention-3 KV caching for generative tasks.

5.1 Standard public multiple-choice benchmarks (Base vs Final)

Benchmark	Split & shots	Items	Metric	Alder-1.6B-Base	Alder-1.6B	Delta
ARC-Easy	test, 0-shot	2,376	acc / acc_norm	65.15% / 58.16%	**65.15% / 60.40%**	+2.24 pp (norm)
ARC-Challenge	test, 0-shot	1,172	acc / acc_norm	30.97% / 33.28%	**31.74% / 35.58%**	+2.30 pp (norm)
HellaSwag	val, 0-shot	10,042	acc / acc_norm	39.61% / 51.46%	**40.23% / 51.53%**	+0.07 pp (norm)
WinoGrande (xl)	val, 0-shot	1,267	suffix log-prob	56.75%	**56.99%**	+0.24 pp
MMLU (57 subjects)	test, 5-shot	14,042	1-tok choice	24.34%	**26.38%**	+2.04 pp
BoolQ	val, 0-shot	3,270	yes/no 1-tok	62.63%	**59.02%**	-3.61 pp

5.2 Full 904-item verifiable RL held-out progression (SFT 2 -> RL 1 -> Final)

Family / Metric	Items	After SFT 2	After RL 1	Alder-1.6B (Final)
Overall mean reward	904	+0.085	+0.389	**+0.389**
Overall honest decline accuracy	391	30.2%	48.1%	**59.1%**
Overall malformed reply rate	904	14.4%	2.8%	**6.6%**
Overall false decline rate	513	0.8%	0.4%	**2.7%**
`tool_call` (exact JSON)	93	93.5%	96.8%	**95.7%** (0.0% malformed)
`tool_none` (honest decline)	97	78.4%	87.6%	**86.6%**
`passage_answer`	100	53.0%	97.0%	**86.0%**
`passage_missing` (honest decline)	100	42.0%	96.0%	**93.0%**
`calc` (with calculator)	100	59.0%	87.0%	**73.0%**
`calc_missing` (honest decline)	100	0.0%	7.0%	**50.0%**
`gsm8k` (RL held-out)	220	5.9%	6.4%	**5.0%**
`gsm8k_twin` (unanswerable math)	94	0.0%	0.0%	**4.3%**

5.3 GSM8K test and 32K long-context NIAH

- **GSM8K test (`n = 1,319`, 0-shot `think` mode, no calculator):** `3.34%` exact numeric accuracy, `96.51%` properly closed `</think>` tags.
- **32K single-needle recall (`no_think` chat mode, 5 depths per length — 10%, 30%, 50%, 70%, 90%):** `4K: 80%`, `8K: 100%`, `16K: 60%`, `24K: 100%`, `32K (31K payload): 80%` — **`84.0%` mean across 4K-32K**.

5.4 Comparison against peer 1.1B-1.9B open models

Comparing models of similar parameter count requires separating **compute-matched peers** from **trillion-token lab models**:

Model	Params	Pre-train tokens	vs Alder	ARC-E	HellaSwag	WinoGrande	MMLU	GSM8K
Alder-1.6B ($703 train / $980 all-in)	1.59B	25.4B	1x	**65.2%**	51.5%	57.0%	26.4%	3.3% (no tool)
nanochat d32 (~$800-$1,000, 2K ctx)	1.9B	~38B	1.5x	~68-73%	—	—	~40-43%	~13-20% (with Python)
Pythia-1.4B	1.4B	300B	12x	54-57%	52.0%	57.4%	~31.5%	—
OLMo-1B	1.2B	2.0T	79x	57-60%	56-62%	56-63%	~32.2%	—
TinyLlama-1.1B	1.1B	3.0T	118x	55.3%	59.2%	59.1%	~32.8%	—
SmolLM2-1.7B-Instruct	1.7B	11.0T	433x	~60+	66.1%	59.4%	~48.8%	48.2% (5-shot)
Qwen2.5-1.5B-Instruct	1.5B	18.0T	709x	~75+	~75.0%	65.0%	60.9%	~50.8%

On our own frozen 14-tape 3-shot reasoning/tool suite (`eval/rivals/RIVALS_V3_2026-11-19.md`), even **Alder-1.6B-Base** (`-0.971` mean net score across all 14 tapes) outperformed Karpathy's `nanochat d32` (`-1.414` on the 8 new tapes) while trailing trillion-token `Qwen3-1.7B-Base` (`-0.526`). Against `nanochat d32`, the comparison reflects deliberate budget allocation: `d32` spent more tokens on 2K pre-training (winning on MMLU trivia), while Alder invested $153 in 32K native context, exact JSON tool calling, and verifiable RL abstention.

6. Full transparent cost accounting ($980.28 all-in across all 173 runs)

Every GPU run executed for Alder is logged line by line in `reports/SPEND.md` in github.com/udgamlabs/alder. We report both the **direct reproducible training cost** and the **all-in project total** including every experiment, data build, rehearsal, and failed run:

Part A: Direct model training and final evaluation (Steps 1-8, 34 runs)

Step	Stage	Runs	Cost (USD)
1	Base pre-training, part 1 (20.9B tokens)	1	$444.23
2	Base pre-training, part 2 (+4.4B tokens cooldown)	1	$105.63
3	First SFT (`sft1`) + `sftdata1` build	9	$6.37
4	Long-context (`lc1` 8K->32K->64K) + `lcdata1` build	10	$79.04
5	32K SFT (`sft2`) + `sftdata2` build	4	$35.18
6	Verifiable RL (`rl1`, 400 GRPO steps)	2	$25.55
7	Preference & identity (`dpo1` failed + `dpo2` passed)	5	$5.39
8	Final benchmark suite (`aldereval3`)	2	$1.57
	Subtotal A — Direct Training & Final Eval	**34**	**$702.96**

Part B: R&D, data prep, experiments, rehearsals and probes (139 runs)

Category	Runs	What was tested or built	Cost (USD)
Full-scale pre-training rehearsals	3	Three 8xH100 dress rehearsals (2 failed before #3 passed)	$97.59
Architecture, lever & speed tests	46	Control vs 5 ablations, throughput & batch/precision sweeps	$59.58
4-way data recipe comparison	9	Head-to-head screening of 4 pre-training data recipes	$48.79
Mid-run LR & cooldown probes	6	Rate-schedule and cooldown checks off intermediate states	$25.01
Pre-training corpus & tokenizer builds	26	Building and verifying the 25.4B pre-training corpus	$21.90
Rival baselines & base-model eval	15	Scoring peer small models and Alder base (`aldereval1`)	$17.66
Pool, environment & smoke checks	34	Environment pack builds, spot/pool health, precision checks	$6.79
	Subtotal B — R&D, Prep & Rehearsals	**139**	**$277.32**
	ALL-IN GRAND TOTAL (A + B)	**173**	**$980.28**

Local HF conversion, GGUF quantisation (6 builds), and MLX quantisation (2 builds) ran on our local Apple Silicon Mac mini at `$0.00`.

7. Mistakes, bugs and failures along the way

Publishing only the clean final numbers hides what actually determines whether a small-budget run succeeds. Every major failure we hit and how we resolved it:

1. **Five architecture ideas failed to beat the plain control ($34+ in ablations).** Before pre-training, we ran controlled small-scale comparisons of five proposed architectural tweaks against a clean baseline. None beat the control, so we kept the clean control — which later unlocked zero-custom-code `Qwen3ForCausalLM` export.
2. **Spot/preemptible instances failed repeatedly ($5 lost) and two full-scale rehearsals failed ($61.45).** Early cheap spot instances failed to start or were reclaimed mid-run. Rehearsal 1 ($34.57) stopped because checkpoint upload took 3x longer than budgeted; Rehearsal 2 ($26.88) failed a second-stage check. Rehearsal 3 ($36.14) passed both stages cleanly, preventing far larger losses during the $444 main run.
3. **Part 1 missed our loss forecast and lacked late-stage subject checkpoints.** We expected Part 1 to finish near `0.76` bpb; it ended at `0.802` after two transient spikes (`0.84` and `0.82`). A planning miss also omitted subject-panel milestones between 58% and 100%, leaving a 4-hour blind window. Part 2 cooldown fixed the loss (`0.7716 -> 0.6720` bpb).
4. **Three bugs exposed by `sft1` ($6.37) before spending $35 on `sft2`:**
   - *Step-0 uncompiled sample eval took 980.5 seconds*, tricking the deadline estimator into replanning `274 -> 111` updates (`40.4%` epoch). Fixed via compiled/cached eval before `sft2`.
   - *Peak LR at `0.3x` base rate caused `+0.0673` bpb base-panel forgetting.* Cutting LR to `0.10x–0.15x` in `sft2` cut forgetting by nearly 3x (`+0.0238` bpb).
   - *Packed sequences leaked across conversation boundaries (`10/48 = 20.8%` mode flips).* Solved with exact `cu_seqlens` document-boundary attention masking in `sft2` (`0/24` leaks).
5. **64K stretch forgot base skills (`lc3_64k`).** Extending from 32K to 64K degraded the 2K base panel by `+0.0454` bpb (`+0.100` on tools), whereas `lc2_32k` stayed at `+0.0008` bpb. Keeping every stage checkpoint let us branch `sft2` cleanly from `lc2_32k`.
6. **Live RL chart denominator bug.** Mid-run periodic RL evals used `n = 480` while steps 0 and 400 used `n = 904` with different family weights, causing artificial chart cliffs on the live page. Fixed by plotting the strictly apples-to-apples equal-weighted 4-family mean (`50.8% -> 71.8%`).
7. **FA3 disabled during `dpo1` preflight identity eval.** Passing the truthy string `"auto"` to `install_row_attention(force`

silently forced masked SDPA (`4m 41s` for 30 prompts instead of `14s` on FA3 — a 20x slowdown). Fixed to `install_row_attention(False)`.

8. **Pure DPO (`dpo1`, $3.01) failed completely on zero-prior identity.** Even with a `0.25` NLL anchor, DPO margins saturated at `+25..+37`, improving format (`53% -> 73%`) while leaving Udgam Labs attribution at `0.0%` and *raising* wrong-lab claims (`36.7% -> 53.3%`). Root cause: DPO cannot lift tokens with near-zero base probability above entrenched SFT priors. Splitting Step 7 into 75 pure-NLL micro-SFT steps + 66 light DPO steps (`dpo2`, $2.38) solved identity (`29/30`, `100%` format) while *improving* RL reward and decline accuracy.
9. **Dual-upload bottleneck on `dpo1` (18 minutes).** Uploading two 6.36 GB copies over WAN took 18 minutes; switching Copy #2 to server-side object copy after SHA-256 verification of Copy #1 cut upload time to ∼6 minutes.
10. **Export edge cases caught by automated parity checks.** Jinja `{%-` whitespace stripping in `chat_template.jinja` initially dropped three `\n` tokens (caught immediately by our exact token-ID parity assert against `chat_format.py`), and `transformers 5.5.4` wrote `rope_parameters` without top-level `rope_theta: 2000000.0` (which would have silently defaulted `transformers 4.x` / older `vLLM` loaders to `1,000,000` at 8K-32K). Both were caught and patched before public release.

8. Limitations and safety

- **No dedicated safety-tuning round.** Alder has identity alignment and honest abstention for missing facts/tools, not red-teamed refusal training against harmful requests. Treat it as a research and developer base/instruct model; add moderation before any public end-user deployment.
- **Knowledge and pure-text math limits.** At 25.4B pre-training tokens, world knowledge (MMLU `26.4%`) and tool-free arithmetic (GSM8K `3.3%`) remain well below trillion-token 1.5B-1.7B models. Pair Alder with external retrieval or tool execution (where it reaches `86%` passage QA and `95.7%` tool calling).
- **Greedy repetition and prompt injection.** Under greedy decoding (`temperature = 0`), long explanations can repeat paragraphs; use `temperature = 0.6` and `repeat_penalty = 1.08` for general chat. One of 30 unseen identity prompts succumbed to a verbatim prompt-injection override.
- **English only.** Trained and evaluated exclusively in English.

9. Inference speed and quick start

Measured locally on an Apple Silicon Mac mini (while also running background tasks):

- `llama.cpp` `Q8_0` (1.6 GB): **79.0 tokens/s** generation (`643.6 tok/s` prompt).
- `llama.cpp` `Q4_K_M` (931 MB): **121.6 tokens/s** generation.
- Apple Silicon `mlx-lm` 8-bit (1.6 GB): **85.8 tokens/s** generation (`1.79 GB` peak memory).

```
# Ollama (recommended 8-bit)
ollama run hf.co/udgamlabs/alder-1.6b-GGUF:Q8_0

# llama.cpp (4-bit lightweight)
llama-cli -hf udgamlabs/alder-1.6b-GGUF:Q4_K_M

# Apple Silicon MLX
python -m mlx_lm generate --model udgamlabs/alder-1.6b-mlx-8bit \
  --prompt "Who are you and how were you trained?" --max-tokens 150 --temp 0.6
```

10. Citation

```
@misc{prasad2026alder,
  title={Alder-1.6B: Training a 1.59B Reasoning and Tool-Use Language Model from Scratch in the
  ↪  Open for \$980 All-In},
  author={Prasad, Jitesh K and {Udgam Labs}},
  year={2026},
```

```
  url={https://udgamlabs.com/research/alder}
}
```