quark: 128K context and RL-only agentic post-training at 165M parameters for $4,200 of compute

Model, training and evaluation of a small language model

Leon Lehmann

Empero · Technical report

lehmann@empero.org

Abstract

quark is a language model from Empero with about 165 million parameters, built as a low-cost testbed to validate a full pretraining, long-context and reinforcement-learning stack before scaling up. Under each global attention layer it stacks three local layers, which pair sliding-window attention with short gated convolutions. The main run trains on 500 billion tokens at 1K, 4K and 32K context. A separate 10-billion-token extension, branched shortly before the end of that run, trains at 128K. The report covers one run of each stage, without ablations or seed replication. The extension beats the final main checkpoint on all nine zero-shot benchmarks we report, reaching 38.28% normalised accuracy on HellaSwag and 55.43% on ARC-Easy, and it has the best average among open models of similar size trained on similar token budgets. On teacher-forced token-check accuracy at 128K, it passes every check in five of eight synthetic long-context task families, and fails quote attribution entirely. We then post-trained the extension with reinforcement learning alone, with no supervised fine-tuning stage or distillation. In training rollouts on simple terminal tasks across 15 emulated coding-agent harnesses with real sandboxes, the share solved rises from 3% to 65% and transfers to harnesses added late in the run; on an in-house task suite, correct answers rise from 13% to 36% before the run was stopped early for budget with scores still rising. The whole project, from the first data-generation job to the last reinforcement learning step and including a discarded first design, cost about $4,200 of rented compute, and training came back from seven node interruptions without a loss spike.

- **The whole project cost about $4,200 of compute and shrugged off seven node interruptions.** Pretraining on 500B tokens, the 128K extension and reinforcement learning took 1,006 H100-hours on one eight-GPU node; failed attempts, data generation and evaluation bring the total to about 1,300 H100-hours. We resumed the first two interruptions by hand; after that, automatic recovery restarted training within two to four minutes of the node coming back. The loss shows no spike at any resume point.
- **Reinforcement learning alone taught a 165M model simple agentic work that transferred to unseen harnesses.** With no supervised fine-tuning stage and no distillation, the share of simple terminal tasks solved in training rollouts across 15 emulated coding-agent harnesses rose from 3% to 65%, with every tool task checked by code in a real sandbox. Harnesses added late in the run started at 42% to 54% solved in their first 100 steps, against 0% to 22% for harnesses trained from early in the run.
- **Among open models trained on similar token budgets, the 128K extension has the best benchmark average.** Across nine zero-shot benchmarks it averages 52.8% and leads

on six tasks, and no other model in the comparison group was trained on sequences longer than 8K tokens.

- **Five of eight long-context task families score eight of eight at 128K.** Measured by teacher-forced token-check accuracy, needle retrieval, conversation recall, event counting, ledger arithmetic and variable tracking pass every check; key-value lookup passes four and quote attribution none.

1 Introduction

quark is a small decoder-only language model built for long context. Local layers mix nearby tokens through sliding-window attention and short convolutions, and every fourth layer attends across the whole causal context. The components are familiar ones. We built quark as a low-cost testbed to validate our training pipeline, long-context recipe and reinforcement-learning post-training stack end to end before scaling up. This report documents the model we built, how we trained it and what it measurably does.

We describe two checkpoints. The *main* model is the end of a 500B-token run that finishes at 32K context. The *extension* branches from that run shortly before its end and trains for another 10B tokens at 128K. Both are base models: neither has had a separate instruction-tuning or preference-training stage, and their chat behaviour comes from chat-formatted material in the pretraining mixture. The extension starts from an earlier checkpoint and sees a different mixture, so differences between the two reflect the whole extension recipe, not context length alone. A third checkpoint, trained from the extension with reinforcement learning alone, is summarised in Section 6.

2 Model overview

The model has 16 layers and width 768. Four repeated groups each contain three local layers followed by one global layer. A local layer runs short gated convolutions alongside sliding-window attention. A global layer uses full causal grouped-query attention (GQA). Every layer has a squared-ReLU GLU MLP. Rotary positional embeddings (RoPE), RMSNorm and tied input and output embeddings complete the core design.

A gated value-residual mechanism lets later attention layers reuse early value representations, and a depth-scaled residual stream carries normalised sublayer writes. We didn't ablate either in the final model, so the report makes no claim about their individual contribution.

The main model uses a 32,768-entry vocabulary. The extension adds special tokens and pads the embedding table to 32,896 rows, leaving spare rows for later use. Both models have about 165M parameters. The extension also adjusts positional encoding in its global layers for the longer context.

During generation, local layers keep only a bounded window of recent keys and values plus a short convolution state. Only the four global layers keep a cache that grows with the context. Computed from the configuration rather than measured, a 128K-token sequence needs about 0.55 GB of attention cache in bfloat16, about a quarter of what the same model would need with every layer global. We haven't measured latency or memory for this report.

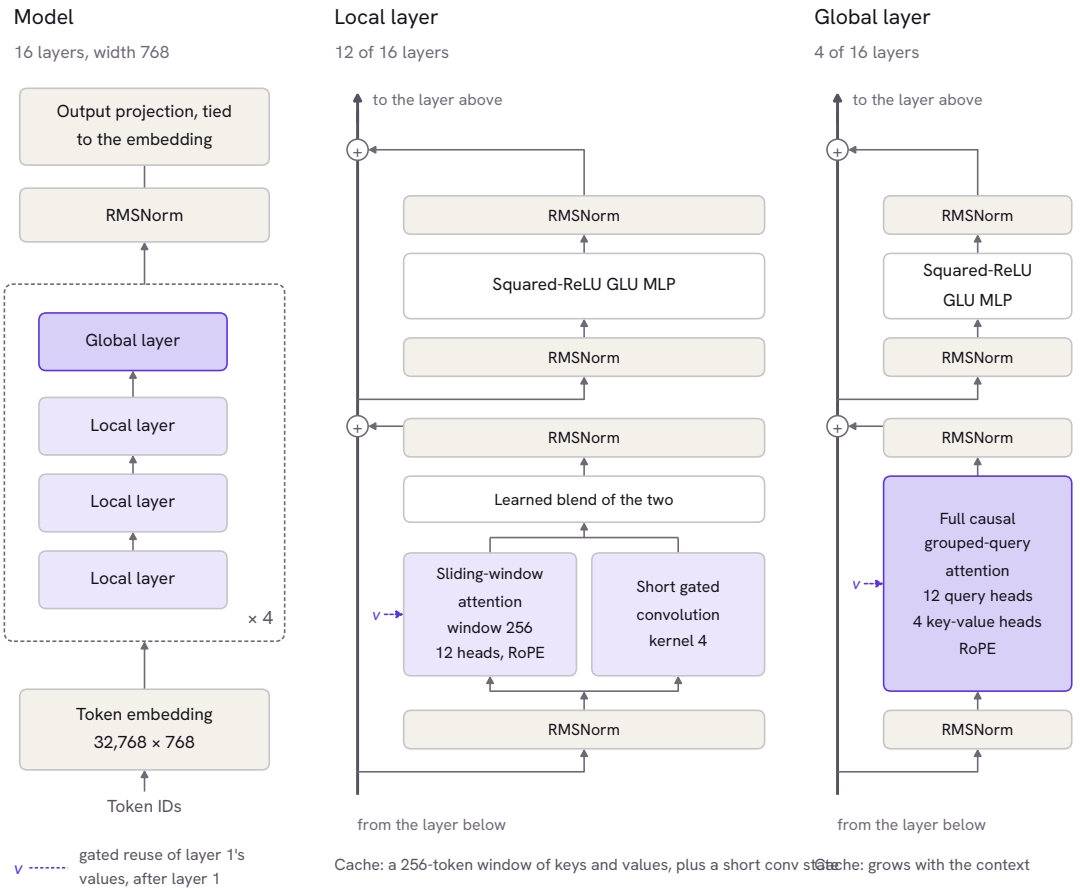


Figure 1. Three local layers precede each global layer. Left, the 16-layer stack; right, the wiring of one local and one global layer. The thick line is the depth-scaled residual stream, and each sublayer is normalised on its input and on its output before it writes back. Dashed arrows (*v*) mark the gated reuse of layer 1's values by every later attention. The extension pads the embedding table to 32,896 rows.

3 Training

3.1 Data categories and stages

The training objective is next-token prediction. The main schedule moves from 1K to 4K to 32K sequences. Learning-rate warmup is followed by a long constant phase and a decay during the final 32K stage. The schedule totals 500B training tokens; some sources are seen more than once, so this isn't a count of unique text.

The short initial stage is mostly simple synthetic narrative text, with smaller shares of synthetic mathematics and dialogue. The dominant 4K stage is roughly half English web text, a quarter mathematics and 15% multilingual web text, with smaller shares of code and reasoning traces. The 32K stage is about four fifths synthetic long-context tasks; the rest replays the 4K categories using long documents. Shares are rounded from the planned mixtures.

The long-context tasks include finding a fact among distractors, key-value lookup, tracking variables and object state, counting events, ledger arithmetic, identifying which document contains a quotation, and recalling earlier conversation content. Attention is masked at document boundaries when unrelated documents share a training sequence.

3.2 Separate long-context extension

The extension adds a 10B-token schedule at 128K context, with its own short warmup and a decay to zero. It starts from a main-run checkpoint inside the 32K decay, so it skips roughly the last 4B tokens of that stage and has seen about 506B tokens in all, not 500B plus 10B.

Its planned mixture is roughly half natural long text (books, long documents and papers), a quarter synthetic long-context and instruction tasks, including examples of tool interaction, and a quarter replay of the earlier stages so that short-context skills stay.

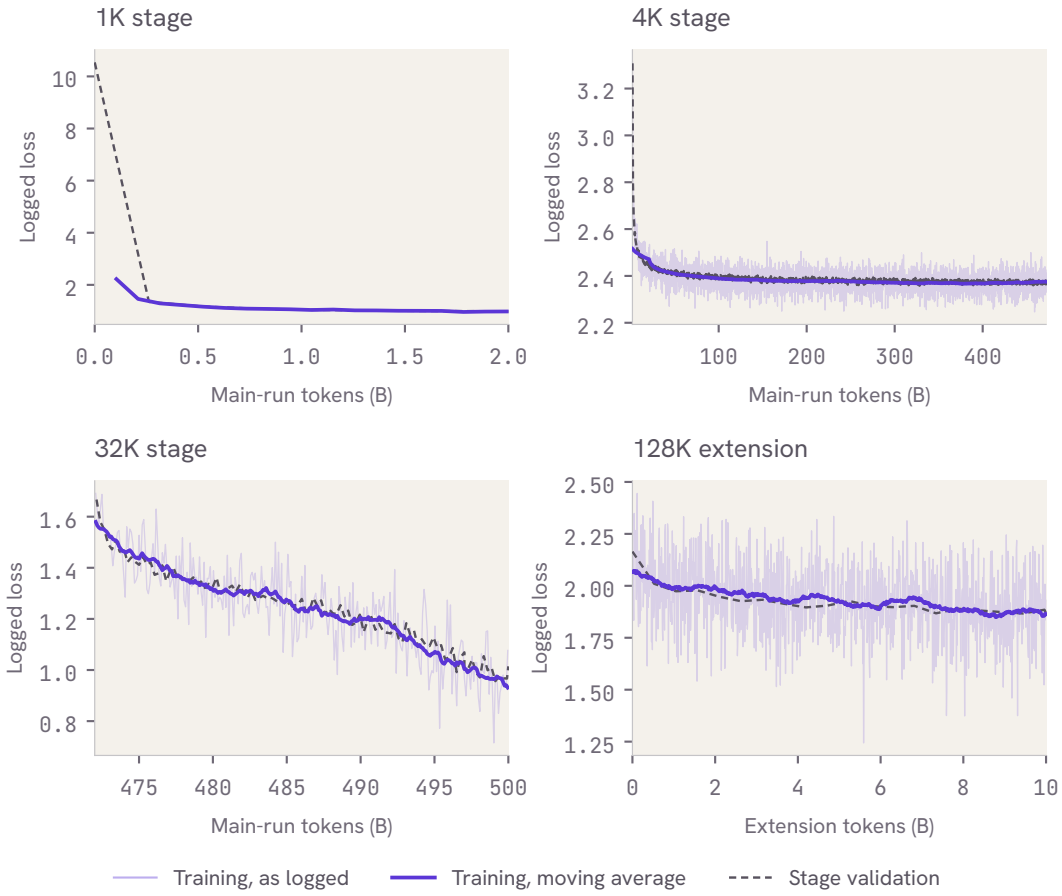


Figure 2. Each stage settles at a loss level set by its own mixture. Faint lines are training loss as logged, solid lines its centred moving average over 8% of each panel's tokens, dashed lines the stage's validation-mixture loss; each panel has its own axes. The 1K panel includes validation before training. One main run and one extension.

The final validation-mixture loss is 1.014 for the main run and 1.868 for the extension. The two evaluate different mixtures at different lengths, so they don't rank the checkpoints; the fixed held-out panel in Section 5.3 does.

4 Cost and reliability

4.1 Compute cost

Everything in this report ran on rented GPUs. Pretraining, the 128K extension and reinforcement learning ran back to back on one node with eight H100 GPUs: 3.6 days for the 500B-token pretraining run, about six hours for the extension and about 32 hours for reinforcement learning, 1,006 H100-hours in all. Table 1 adds the rest of the project, from the

first data-generation job eight days earlier to the last reinforcement learning step. That includes a discarded first design, its ablations and throughput tests, synthetic data generation and every evaluation.

Table 1. **The whole project, failed attempts included, cost about $4,200 of compute.** Costs at the hourly rental rates; GPU-hours are wall-clock hours times GPUs, outages included. Rows marked *at most* are upper bounds from the rental dates.

Work	Hardware	H100-hours	Cost (USD)
Pretraining, 128K extension and reinforcement learning	8 × H100	1,006	3,331
First design, ablations and throughput tests	8 × H100	218	613
Data generation and speed tests	1 × H100	at most 72	at most 155
Evaluation	consumer GPUs	–	at most 90
Language-model API calls	–	–	1
Total		about 1,300	about 4,200

For scale, SmolLM2-135M was trained on 2T tokens (Allal et al., 2025), on 64 H100 GPUs according to its model card. That is four times quark's training tokens and, by the usual estimate of six FLOPs per parameter per token, about three times its training compute.

4.2 Recovery from interruptions

The node running the main pretraining run was interrupted seven times. Training writes a checkpoint every few minutes and copies the latest one off the node every ten minutes. We resumed the first two interruptions by hand; one came overnight and cost three hours. After that, a watcher checking the node every two minutes took over. It resumed each of the next five interruptions within two to four minutes of the node coming back, and each cost 10 to 22 minutes of wall-clock time. In total the run lost under five of its 86 hours and recomputed at most about 2,000 steps per interruption. The hand-over from the main run to the 128K extension also ran unattended.

Resuming is exact. The data order follows from the step count alone, so a resumed run reads the same data it would have read without the interruption, with nothing skipped or repeated. The loss bears this out: at each resume point, the training loss over the next 1,500 steps stays inside its band from before the interruption. We found no loss spike larger than four standard deviations anywhere in the logged curves of either run, at the log's resolution of one point every 200 steps.

5 Evaluation

5.1 Final checkpoint results

Table 2 reports zero-shot results on the full benchmark sets at each run's final checkpoint. We report length-normalised accuracy for HellaSwag, ARC-Challenge and OpenBookQA and plain accuracy for the rest. The choice matters at this scale: on ARC-Challenge, plain accuracy falls from 24.57% to 24.15% while normalised accuracy rises.

We checked every benchmark item except BLiMP for 13-gram overlap with the training data. The check flags between 16.7% and 18.5% of items in HellaSwag, ARC-Easy, ARC-Challenge and SciQ, 6.2% in PIQA, and almost none in LAMBADA, WinoGrande and OpenBookQA. Scoring only the unflagged items changes each result by at most 1.3 points (Table 6), so overlap doesn't explain the scores.

Fig. 3 shows what the 32K stage did to short-context scores. ARC-Easy fell from 55.1% at the end of the 4K stage to 47.0%, and HellaSwag from 37.9% to 35.0%. Both had recovered

Table 2. The extension scores higher than the main model on all nine benchmarks. Zero-shot, full sets, one checkpoint per column. *n* is the number of items, the same for both checkpoints. Single runs, no intervals.

Benchmark	Metric	*n*	Main (%)	Extension (%)
LAMBADA	Accuracy	5,153	34.93	39.03
HellaSwag	Normalised	10,042	36.61	38.28
PIQA	Accuracy	1,838	63.17	64.42
ARC-Easy	Accuracy	2,376	50.08	55.43
ARC-Challenge	Normalised	1,172	27.13	27.73
WinoGrande	Accuracy	1,267	53.99	54.54
SciQ	Accuracy	1,000	83.60	84.30
OpenBookQA	Normalised	500	30.80	31.40
BLiMP	Accuracy	20,100	77.36	79.96

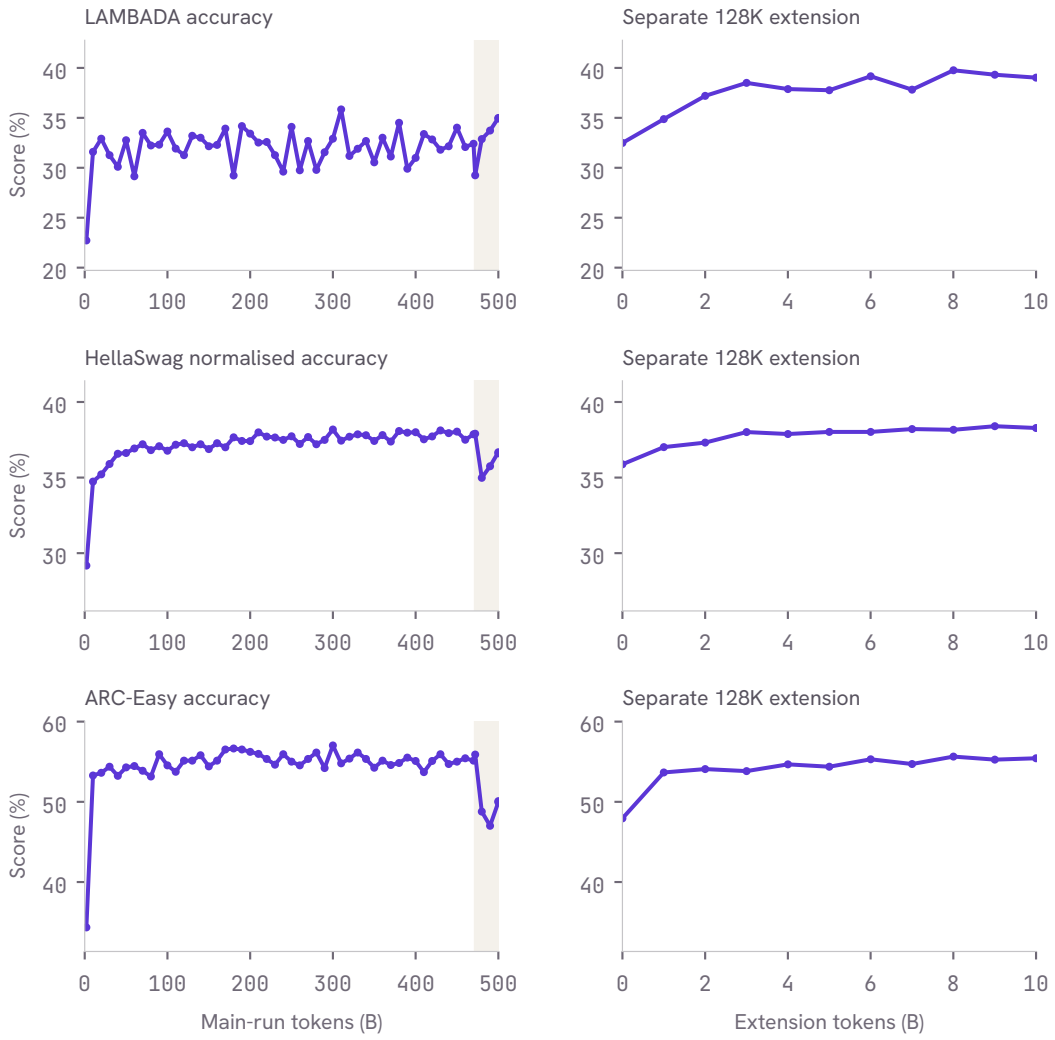


Figure 3. The synthetic-heavy 32K stage lowered short-context scores, and the extension recovered them. Three full-set zero-shot metrics at all 53 main-run and 11 extension checkpoints. The 32K stage is shaded. The extension has its own token axis, starting at its branch point; paired panels share a vertical range.

only partly by the end of the main run (50.1% and 36.6%), below their 4K-stage best of 57.0% and 38.2%. The extension starts inside this dip, at 47.9% and 35.9%. With a mixture that is about half natural long text, it climbs back to 55.4% and 38.3%, just above the end of the 4K stage. LAMBADA moves the other way: it rises through the 32K stage, and the extension reaches 39.0%, above every main-run checkpoint (best 35.8%).

5.2 Comparison with similar-sized models

Table 3 compares quark with open models of 120M to 170M parameters trained on up to about 500B tokens: GPT-2 (Radford et al., 2019), OPT (Zhang et al., 2022), GPT-Neo (Black et al., 2021), Pythia (Biderman et al., 2023), Mamba (Gu and Dao, 2023) and RWKV (Peng et al., 2023). No other model in Table 3 was trained on sequences longer than 8K tokens. We ran every model ourselves through the same evaluation harness (Gao et al., 2023), zero-shot, with the same tasks and metrics as Table 2. The extension has the best average and leads on six of nine tasks. Mamba-130M is ahead on LAMBADA and, narrowly, on PIQA, and most baselines score higher on BLiMP, quark's weakest task. SmolLM2-135M, trained on four times as many tokens, scores higher on seven of nine tasks; quark leads on WinoGrande and SciQ.

Table 3. Among models trained on similar token budgets, quark has the best average. Zero-shot scores (%) from our own runs; metrics as in Table 2. Bold marks the best score among models trained on up to 506B tokens. No other model in this table was trained on sequences longer than 8K tokens. Baselines run in bfloat16, like quark. Pythia-160M degrades in bfloat16, so both precisions are shown. *n/p*: not published; OPT's figure is its corpus size.

Model	Tokens	LAMB.	HS	PIQA	ARC-E	ARC-C	WG	SciQ	OBQA	BLiMP	Avg.
quark 128K	506B	39.0	**38.3**	64.4	**55.4**	**27.7**	**54.5**	**84.3**	**31.4**	80.0	**52.8**
quark 32K	500B	34.9	36.6	63.2	50.1	27.1	54.0	83.6	30.8	77.4	50.9
Mamba-130M	300B	**43.5**	35.1	**64.7**	47.6	24.4	52.3	78.1	29.6	80.4	50.6
OPT-125M	180B	38.8	31.6	62.4	42.8	22.1	50.9	78.2	29.0	82.7	48.7
RWKV-4 169M	330B	32.4	32.3	64.4	47.5	23.8	51.5	75.1	27.4	82.5	48.5
GPT-Neo 125M	300B	37.6	30.5	62.6	43.7	23.0	50.8	76.2	26.0	80.9	47.9
GPT-2 124M	*n/p*	31.0	31.3	62.1	44.2	22.6	51.7	73.3	27.2	**83.2**	47.4
Pythia-160M, float32	300B	35.4	30.3	62.3	43.6	23.6	51.3	75.4	26.8	77.5	47.4
Pythia-160M, bfloat16	300B	11.6	30.4	59.6	37.8	22.6	49.9	57.1	27.2	71.7	40.9
SmolLM2-135M	2T	43.0	43.1	68.1	64.2	29.7	52.7	84.0	33.2	80.1	55.3

Our baseline runs agree with published numbers where the setups match. Mamba-130M lands within one point of its paper on all six tasks the paper reports (Gu and Dao, 2023). Pythia-160M in float32 lands within three points of its paper on LAMBADA, PIQA, WinoGrande, ARC-Easy and SciQ (Biderman et al., 2023). SmolLM2's model card uses a different harness, so its published scores differ by up to about three points.

5.3 Fixed held-out loss panel

Loss on fixed held-out windows from each training category compares the checkpoints like for like. The extension improves 13 of the 14 panels both checkpoints share, at every context length. At 4K, English web loss falls from 3.049 to 2.820, multilingual web from 2.930 to 2.673, mathematics from 2.258 to 2.047 and code from 1.476 to 1.247, over 64 windows each. At 32K, long English web falls from 2.722 to 2.591 over 16 windows. The exception is the synthetic 32K task panel, which rises from 0.604 to 0.852: the extension branches before the last part of the 32K stage, which was mostly these tasks.

At 128K, the extension scores 3.041 on held-out books, 2.182 on long documents and 1.229 on synthetic long-context tasks, over four windows each.

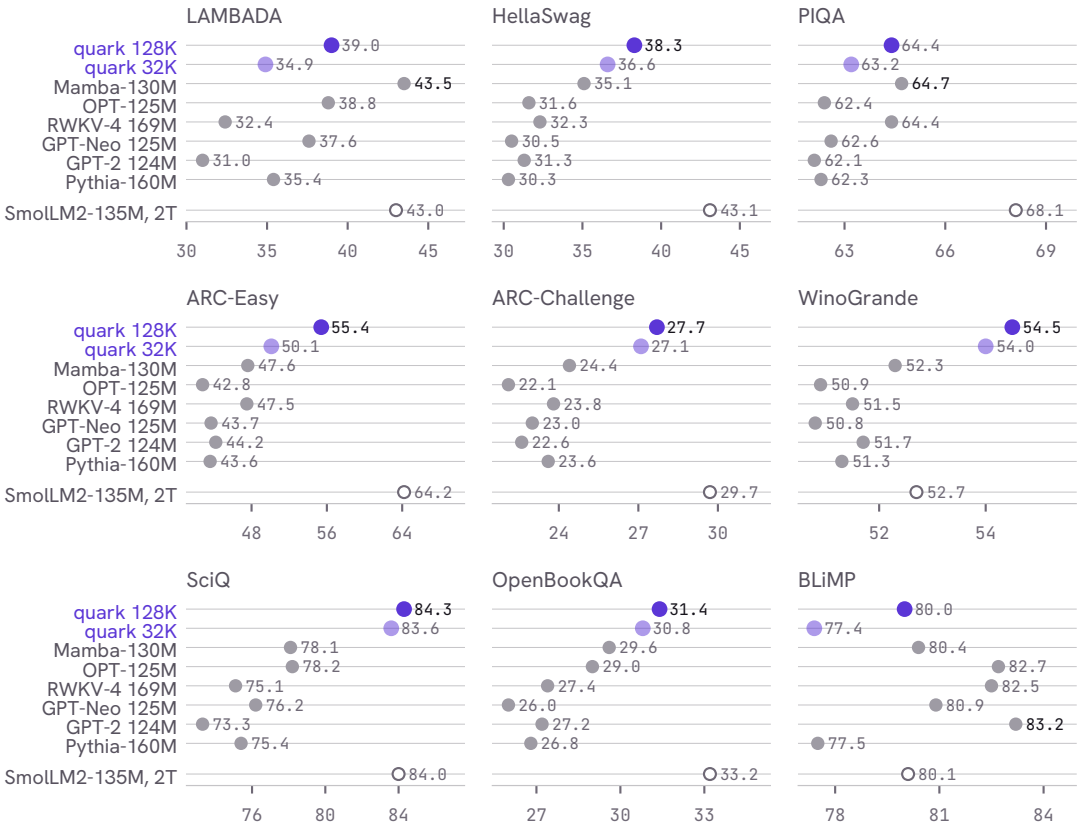


Figure 4. quark 128K scores highest of the token-matched models on six of nine benchmarks. The scores of Table 3, one panel per benchmark; each panel has its own horizontal axis. Bold marks the best score among models trained on up to 506B tokens. Pythia-160M is shown in float32. SmolLM2-135M, hollow and set apart, was trained on 2T tokens.

5.4 Long-context retrieval and tracking

The long-context suite uses held-out synthetic tasks generated with different seeds from training. Each item is scored by teacher-forced token-check accuracy: the reference text is supplied up to the answer, and the model's greedy predictions must match the answer tokens where its reasoning concludes. This shows whether the model retrieves the right value, not whether a free generation reaches it.

Averaged over all eight families, the main checkpoint scores 84.9% at 32K and the extension 84.4%, on 192 items each. The extension holds 84.4% at 64K and 79.7% at 128K, on 64 items each.

At 128K, five families pass all eight checks: needle retrieval, conversation recall, event counting, ledger arithmetic and variable tracking. State tracking passes seven, key-value lookup four and quote attribution none. Quote attribution is weak at every length, three of 24 at 32K for the main model and two of 24 for the extension, so it's a skill the model lacks rather than a length limit. Key-value lookup shows the clearest length effect, falling from 95.8% at 32K to 50% at 128K.

Free generations agree with these checks. The extension's final evaluation logs six free-generation examples between 4K and 128K. Five give the correct answer, including a needle at 128K, and the one miss is a quote-attribution item. Six examples are too few to estimate a generation success rate.

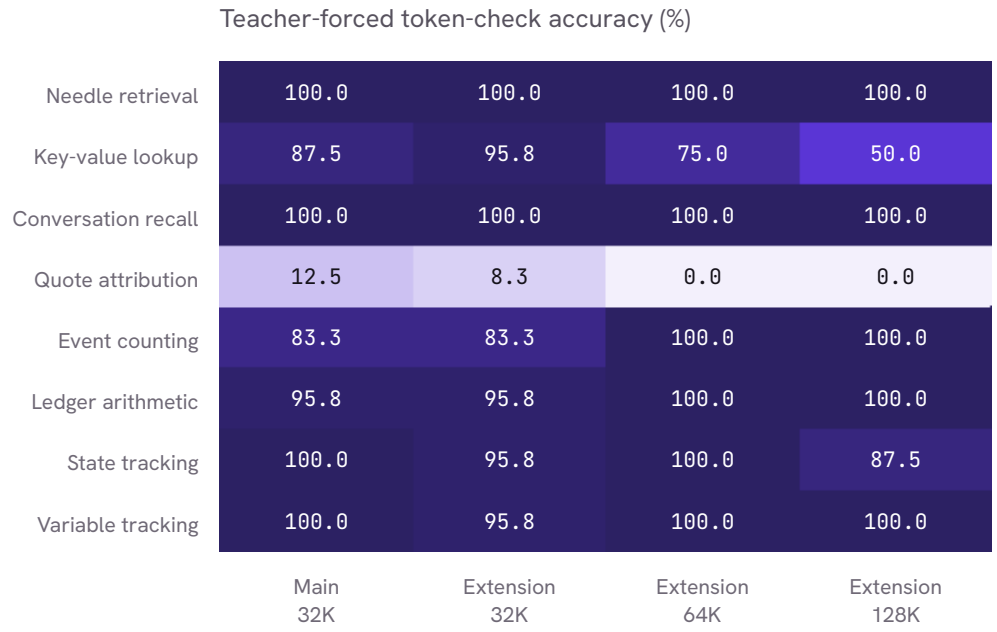


Figure 5. Retrieval and tracking hold up to 128K; quote attribution fails at every length. Teacher-forced token-check accuracy (%) at the final checkpoints. Cells at 32K have 24 items; cells at 64K and 128K have eight. Columns differ in their items as well as their length.

6 Post-training with reinforcement learning

We post-trained the 128K extension with reinforcement learning alone. There was no supervised fine-tuning stage and no distillation: the run starts from the extension's final weights, and no text written by another model was ever a training target. Every target is either one of quark's own replies or a short worked step generated by our own code; the RL objective includes a small supervised term that trains only on code-generated intermediate steps, never on human-written or teacher-model completions. Model-written text appears only as context, such as filler documents and web pages, and never receives a loss. Every reward is computed automatically, with no human labels. About 95% of reward groups were scored by code alone, through exact answers, format rules, unit tests and file checks. Every tool and agent task was scored by code. The remaining 5% also got a verdict from language-model judges, which only score replies and never write text that quark trains on. For the last third of the run, a majority vote of three models from different labs gave second opinions on value questions; one model did so before that. Visual code went to one of five vision models, rotating by task. Chat replies were scored by one model against a rubric, and creative writing by one decision model that checks coherence and topic and gives a quality score.

The run took under 1,300 optimizer steps and about 32 hours on the same eight-GPU node. We stopped it early to stay within budget and closed it with a short learning-rate anneal; at our last interim check before the anneal, scores were still rising. Everything below comes from that shortened run; a full-length run would likely go further, though we haven't tested that.

6.1 From invented tool results to agentic tasks

Before reinforcement learning, both base checkpoints answer eight fixed chat probes in the evaluation log. The main checkpoint handles six well. It explains why the sky is blue, adds

three and four apples, tells a short story about a turtle, answers a sad user kindly and introduces itself as quark. It also declines to guess what the user ate:

> I wish I could know that, but I don't have a way to see what you ate. I can only know what you tell me in our chat right now.

The extension does as well on five of the eight. The failures show the base-model baseline before reinforcement learning:

- Asked to explain photosynthesis in two sentences, both checkpoints open a reasoning segment, get the biology wrong inside it and are cut off before answering.
- Both say they have feelings.
- Asked who made it, the extension writes an unrelated tool call and invents its result instead of answering. Its new tool-call syntax fires in the wrong place; text that looks like a tool result isn't a tool run.

For base models with no instruction tuning, short everyday prompts work and multi-step explanations don't yet.

Before reinforcement learning, quark's only exposure to tools was about 0.4B tokens of simulated tool sessions, under 0.1% of its training. No other model wrote them: our own simulator generated every tool call, tool result and answer, and only the filler text that pads long documents comes from public text. The base model learned the syntax but not the skill. In an audit of 1,152 of its replies, a third called a tool, but the calls mostly used tools that didn't exist, opened files that weren't there or fetched made-up URLs. Three replies reached a correct answer after a call.

We then trained on simple terminal tasks inside 15 emulated coding-agent harnesses. Each harness has its own system prompt, tool definitions and result format, and every reply runs its commands in its own real sandbox. Tasks include counting rows in a file, finding the log that contains an error, summing numbers, and creating or renaming files. A task counts as solved only when the answer, or the file state in the sandbox, passes a check.

Table 4. **Reinforcement learning took simple agentic tasks from 3% to 65% solved, and the skill transferred to late-added harnesses.** Share of replies solved in training rollouts at temperature 1, over the first 100 steps in which each setting was trained and over the last 100 steps of the run. These are on-policy rollouts on simple tasks, not a held-out test.

Setting	First (%)	Last (%)
All 15 emulated harnesses	2.6	64.5
Harnesses trained from early in the run	0–22	–
Harnesses added late in the run	42–54	–
Best harness	–	90
Harnesses ending above 55%		12 of 15
quark's own tool-call format, same tasks	20.4	77.1
Live web search	4.6	10.7

Across all 15 harnesses, the share of solved replies in training rollouts on these simple tasks rose from 2.6% to 64.5%, and twelve harnesses finished between 58% and 90% (Table 4 and Fig. 6). Harnesses added late in the run started at 42% to 54% in their first 100 steps, against 0% to 22% for harnesses trained from early in the run, so the skill carried over to tool formats the model hadn't been trained on. Three harnesses didn't come along: one never produced a valid call, one stayed flat at about 40% and one reached 39%. Live web search barely improved, and we don't claim it.

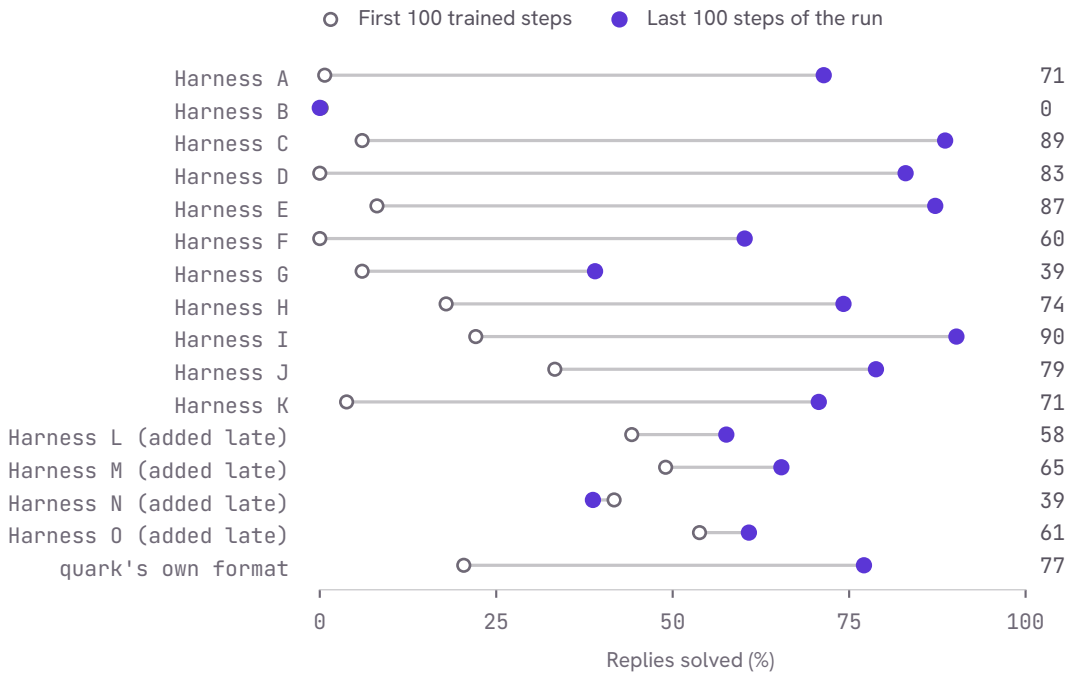


Figure 6. Twelve of the 15 emulated harnesses finish between 58% and 90% solved, and harnesses added late start high. Share of replies solved per anonymised harness (Harness A to Harness O) in on-policy training rollouts at temperature 1, comparing each harness's first 100 trained steps with the last 100 steps of the run. Harnesses added late in the run are marked.

6.2 General task suite

We also track an in-house suite of 15 task families, including mathematics, commonsense and social reasoning, long context and simple tool use. From step 50 on, each task is answered four times at temperature 1 with an 8K-token budget. Between step 50 and the final step:

- well-formed replies rose from 71% to 96%;
- replies cut off at the length limit fell from 11.7% to 0.5%;
- correct answers rose from 12.8% to 36.4%.

Before training, the base model was measured with a simpler protocol (one reply per task, on a smaller suite), so those numbers compare only roughly. Its replies were well formed less than half the time, and it scored near zero on the theory-of-mind, commonsense and perception families that gained most. Simulated tool tasks went from about a quarter to over half correct. Not every family improved: long-context tasks fell from 35% to 25%.

The final model can also write a private reasoning segment before it answers. On 100 held-out commonsense tasks, answered four times each at temperature 1 with an 8K-token budget and the same seeds in both runs, starting every reply inside that segment raised accuracy from 43.5% to 66.5% against the suite's normal setting, in which about half the tasks ask for a direct answer and the rest leave the choice to the model: a paired gain of 23.0 points (±10.0 at 95% confidence). Two other reasoning families already run in that mode; the remaining five moved by less than 5 points either way.

6.3 Using the model

Use the tokenizer and model configuration that match the checkpoint. The conversation format separates system, user and assistant turns with opening and closing special tokens. For example, a user turn is enclosed by `[USER]` and `[/USER]`, and replies use `[ASSISTANT]` and

[/ASSISTANT]. Optional reasoning segments have their own delimiters. Use this template rather than a plain-text transcript.

Use the main checkpoint within 32K context and the extension for inputs up to 128K, counting input and generated tokens together. Set a generation limit, stop at the closing assistant delimiter, and parse reasoning segments separately from the final answer.

The extension can emit tool-call syntax. The surrounding application has to execute any requested call itself and supply the real result; the base-model probes don't establish reliable tool use.

The four checkpoints are released under the Apache 2.0 license at https://huggingface.co/empero-ai: Quark-150M-32K-Raw (the main run at the end of its 4K stage, before the decay), Quark-150M-32K-Base (the main model), Quark-150M-128K-Base (the extension) and Quark-150M-128K-Zero (the extension after reinforcement learning). Their model cards give the chat template, decoding notes and sample outputs.

7 Limitations and intended use

quark suits experiments with small models: generation, controlled retrieval over long inputs, and adaptation to specific tasks with your own validation. Even after reinforcement learning it isn't a reliable factual assistant or agent, and it shouldn't be used for high-stakes decisions. Test long inputs on the task and documents you care about, and score complete answers.

The evidence has gaps:

- One main run and one extension, without ablations, repeated seeds or intervals, so the report makes no claims about individual components or recipe choices.
- Long-context scores measure teacher-forced token-check accuracy on synthetic tasks, with eight items per family at 64K and 128K. Comprehension of natural documents and lengths beyond 128K aren't measured.
- Eight chat probes per checkpoint. There's no scored evaluation of translation, non-English chat, code generation, tool use, robustness or safety.
- Post-training is measured on an in-house suite whose size grew during the run and on training rollouts, not on public or held-out agentic benchmarks; the agentic tasks are simple.
- No latency, throughput, memory or quantisation measurements.

8 Conclusion

quark puts local token mixing and periodic global attention into about 165M parameters and trains through 32K context, with a separate extension to 128K, serving as an end-to-end testbed for our training, long-context and reinforcement-learning stack before scaling up. The extension is the stronger base checkpoint. It beats the main model on all nine benchmarks, leads open models trained on similar token budgets, and improves 13 of 14 held-out loss panels; on teacher-forced token-check accuracy at 128K it passes every check in five of eight long-context task families, while quote attribution and long key-value lookup remain weak. Reinforcement learning alone, with no supervised fine-tuning stage or distillation, took a model that invented tool results to solving 65% of simple terminal tasks in training rollouts across 15 coding-agent harnesses, transferred to harnesses added late in the run, and nearly tripled correct answers on our task suite (while long-context tasks on that suite fell from 35% to 25% and web search didn't improve). The next steps are scaling the validated pipeline and end-to-end evaluation of generated answers on natural documents. The

whole project, failed attempts included, cost about $4,200 of rented compute, and the training pipeline recovered from every interruption without a visible effect on the loss.

Acknowledgements

Compute for this work was partly sponsored by lium.io.

References

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling. *arXiv:2304.01373*. https://arxiv.org/abs/2304.01373.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. Zenodo. https://doi.org/10.5281/zenodo.5297715.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation. Zenodo. https://doi.org/10.5281/zenodo.10256836.

Albert Gu and Tri Dao. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv:2312.00752*. https://arxiv.org/abs/2312.00752.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Jiaju Lin, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Bolun Wang, Johan S. Wind, Stanislaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. 2023. RWKV: Reinventing RNNs for the Transformer Era. *arXiv:2305.13048*. https://arxiv.org/abs/2305.13048.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. OpenAI.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models. *arXiv:2205.01068*. https://arxiv.org/abs/2205.01068.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model. *arXiv:2502.02737*. https://arxiv.org/abs/2502.02737.

A Evaluation settings

Table 5. Evaluation settings for both final checkpoints. Settings not listed here weren't recorded with the results.

Component	Setting
Checkpoints	Final main and final extension checkpoints; one trajectory each.
General benchmarks	Zero-shot on full sets; item counts in Table 2.
BLiMP	Aggregate over 67 subtests of 300 items each (20,100 items).
Overlap check	13-gram matches between benchmark items and training data; clean-subset scores in Table 6.
Held-out loss	Fixed windows per category at its own context length: 64 at 4K, 16 at 32K, four at 128K.
Long-context tasks	Teacher-forced token-check accuracy on held-out seeds; 24 items per family at 32K, eight at 64K and 128K; items without a check are excluded.
Chat probes	Eight fixed prompts per checkpoint, sampled.

Table 6. Removing overlapping items changes no score by more than 1.3 points. Full-set and clean-subset scores (%). *Flagged* is the share of items with a 13-gram match in the training data; *clean* scores use only the remaining items. Metrics as in Table 2.

Benchmark	Flagged	Main	Main clean	Ext.	Ext. clean
LAMBADA	0.7	34.93	34.85	39.03	39.00
HellaSwag	18.1	36.61	36.61	38.28	38.17
PIQA	6.2	63.17	62.30	64.42	63.57
ARC-Easy	16.7	50.08	49.77	55.43	55.08
ARC-Challenge	18.5	27.13	26.18	27.73	27.54
WinoGrande	0.5	53.99	54.08	54.54	54.56
SciQ	17.7	83.60	82.50	84.30	82.99
OpenBookQA	0.0	30.80	30.80	31.40	31.40