

Laghu: a 152M chat model trained from scratch for \$146, released fully open

UDGAM LABS VERSION 1.0.0

Abstract

Laghu is a 152M-parameter English chat model with a thinking on/off switch and a 16K-token context. Udgam Labs trained it from scratch, with our own tokenizer and data recipe, and releases everything: the weights of the chat model and both base models, every training checkpoint that survives, the training code, the full recipe including data shares and hyperparameters, data rebuild scripts, every evaluation output item by item, the failures and the cost.

- **Budget.** The main pretraining run cost \$146.26 (8× A100 80GB for 13.2 hours); the whole project, including data generation, test runs and the evaluation judge, cost about \$194.
- **Base model.** Laghu read about 71 billion training tokens in all stages, roughly 28 times fewer than SmoLM2-135M's published 2 trillion pretraining tokens. On 1,695 documents published in September and October 2026 (our new-text set), Laghu-Base scores 0.919 bits per byte, better than SmoLM2-135M (0.941) and Falcon-H1-Tiny-90M-Base (0.979) and 0.005 behind our 384M Gargi-Base. On five zero-shot commonsense benchmarks it averages 49.9, 5th of the 19 models up to 200M parameters we measured on 19 October (SmoLM2-135M: 51.8).
- **Chat model.** In one harness with the same items for every model, Laghu is level with SmoLM2-135M-Instruct on judged chat quality (51.0% win rate) and on multi-turn follow-ups (41.7% vs 37.5%), and ahead of it on maths with thinking (GSM8K 7.9% vs 1.8%). It is level with our own Gargi on chat quality (45.9% win rate, $p = 0.06$) at 40% of Gargi's size.
- **Where it is behind.** Falcon-H1-Tiny-90M-Instruct, trained on 800B tokens with a preference-tuning stage (about 12 times Laghu's compute), is ahead on all three of our pre-registered chat categories. Laghu's maths (GSM8K 7.9%) and instruction following (IFEval 26.1 vs Falcon's 60.8) are weak.
- **Long context.** After a 1B-token stretch at 32K, Laghu-Base-32K finds a hidden code 96-100% of the time up to 16K tokens and never at 24K or 32K.
- **What did not work.** One reinforcement-learning attempt made maths no better and loops worse; an SFT validation split leaked; the 32K stretch slightly overshot a self-set limit on reasoning text; and the intermediate pretraining checkpoints were lost in a storage clean-up. All are described in section 10.

1. Introduction

Small open language models are usually pretrained on trillions of tokens. Laghu asks how good a small chat model can be on a small, fixed budget: about \$150 of GPU time for pretraining, one consumer-cloud GPU for everything after it, and openly licensed data. We wanted the result to be easy to study, so we release every part of it, including the parts that went wrong.

Laghu was built under four constraints:

1. **From scratch and open:** random initialisation, our own 32,768-token tokenizer, openly licensed training data (or text written by openly licensed models and by API models whose terms allow it), Apache-2.0 weights and code.
2. **Small and cheap:** about 150M parameters and a pretraining budget of about \$150.
3. **Standard formats:** the Gemma 4 text architecture and chat format, which transformers, llama.cpp, Ollama and MLX already run.
4. **A general recipe:** data and post-training aimed at broad assistant behaviour, never weighted toward a benchmark, using only methods that also work at larger scale.

Before measuring Laghu we wrote down how we would evaluate it, which rivals we would compare against, and the bar for calling it best in its size class (section 9). Laghu did not clear that bar: one rival trained with about 12 times our compute is ahead. It does show how far a small budget goes, and we publish it in full as a research release.

Laghu (लघु) is Sanskrit for "small, light".

2. Models and links

MODEL	WHAT IT IS	HUGGING FACE
Laghu	chat model (SFT, 2 passes), thinking on/off, 16K context	udgamlabs/Laghu
Laghu-Base	end of pretraining (EMA weights at step 76,294), 2K context; raw weights in <code>raw-final/</code>	udgamlabs/Laghu-Base
Laghu-Base-32K	after the 32K stretch (step 1,900); the start of SFT	udgamlabs/Laghu-Base-32K
Laghu-Checkpoints	raw training states with optimiser (pretraining 76,000 and 76,294; 32K 1,750 and 1,900), SFT 1/2/3 passes, the failed RL run, logs	udgamlabs/Laghu-Checkpoints
Laghu-GGUF, Laghu-MLX	builds for llama.cpp / Ollama / LM Studio and for Apple MLX	udgamlabs/Laghu-GGUF, udgamlabs/Laghu-MLX
Code	data builders, training, SFT, RL, evaluation, dashboard	github.com/udgamlabs/laghu
Evaluation data	per-item outputs of every model, judge verdicts, validation sets	udgamlabs/laghu-evals
SFT data	41,467 training conversations with per-row source, generator and licence	udgamlabs/laghu-sft

3. Architecture

ITEM	VALUE
Layout	Gemma 4 text (<code>Gemma4ForCausalLM</code>), 28 layers, width 640, MLP width 1,792, GELU (tanh)
Parameters	151.8M, tied input and output embeddings counted once
Attention	10 query heads and 2 key-value heads; three local layers (head dimension 64, 512-token sliding window) then one global layer (head dimension 128), repeated
Positions	RoPE, base 10,000 on local layers and 1,000,000 on global layers
Output	final logit soft-cap 15
Tokenizer	BPE with byte fallback, 32,768 tokens, trained on 540M characters (60M from each of nine source groups: web, FinePhrase, maths, code, SYNTH, STEM, chat, tool use, Wikipedia); shared with Gargi
Context	2,048 in pretraining; 32,768 in the stretch; 16,384 in the chat model's config

The sliding-window layers keep the key-value cache small: only every fourth layer stores keys and values for the whole context.

4. Data

4.1 Pretraining pool

The pool was built by `udgam/build_mix.py --mix v3` on a 32-vCPU cloud machine in about 3.7 hours (about \$5.8 including one failed attempt). It has 52.6B tokens: phase A (the stable mix) 42.17B, phase B (the cool-down mix) 10.33B, validation 0.10B.

SOURCE	LICENCE	SHARE OF POOL TOKENS
FineWeb-Edu (HuggingFaceFW)	ODC-By 1.0	17.5%
DCLM-Edu (HuggingFaceTB)	CC-BY-4.0	17.4%
FinePhrase: FAQ, table, tutorial and maths rewrites (HuggingFaceFW; SmoLLM2-1.7B-Instruct, Apache-2.0)	ODC-By 1.0	23.0%
Ultra-FineWeb, English (OpenBMB)	Apache-2.0 (FineWeb, ODC-By)	14.5%
FineMath 4+ (HuggingFaceTB)	ODC-By 1.0	8.9%
SYNTH (PleIAs; Qwen3-8B fine-tunes, Apache-2.0)	CC-BY-4.0	8.6%
Wikipedia, English (2023-11-01)	CC-BY-SA-3.0 / GFDL	3.4%
Nemotron Post-Training v2, chat (NVIDIA; Qwen3, Apache-2.0)	CC-BY-4.0	3.3%
Nemotron Pretraining Specialized v1, STEM and reasoning Q&A (NVIDIA; DeepSeek-R1-0528, Qwen, gpt-oss)	CC-BY-4.0	1.8%
github-code-clean, permissively licensed files only	per file: MIT, Apache-2.0, BSD, ISC, CC0, Unlicense	1.7%

Shares are the builder's own token estimates per source (they sum to 48.8B rather than the 52.6B tokenized total). The licence policy excludes any data generated by models whose licence reaches models trained on their outputs (for example Llama, Qwen-licence and DeepSeek-licence models) and any non-commercial or unclear source; the builder enforces it with a generator allow-list. Every document was checked against ten evaluation sets (BFCL, IFEval, MMLU, HellaSwag, ARC, MATH-500, PIQA, OpenBookQA, GSM8K, CommonsenseQA) with a 13-gram index; 21,427 documents (0.027%) were removed. Most removals were in FineMath (9,241) and the web sets.

The pool is published as a rebuild script plus the source list, versions, seeds, decontamination index and per-shard manifest, not re-hosted: its shards keep only the text, so per-row credit for the share-alike and per-file-licensed rows would be lost.

4.2 Long-context data

About 1.0B tokens: 42% long documents (books, scientific papers, government text and repository-level code, from Common Pile sources such as Project Gutenberg, pre-1929 books, arXiv, DOAB and UK Hansard, prepared earlier for Gargi), about 8% long modern documents (web, code and Wikipedia), and 50% replay of cool-down text the main run had not read. The same 13-gram decontamination applies. This data is also published as a rebuild script.

4.3 Supervised fine-tuning data

41,467 training conversations (853 more for validation), 34.5% of them multi-turn, 28.2M tokens per pass, of which 9.57M are trained on (assistant turns only; tool results are masked).

SLICE	CONVERSATIONS	HOW IT WAS MADE
Multi-turn follow-ups	10,388	gpt-oss-120b (Apache-2.0) conversations from our earlier work
Chat, thinking off	8,232	real-user-style prompts (WildChat, Chatbot Arena, oasst2) answered by DeepSeek-V4.1-Flash
Maths and logic (thinking on and off)	8,768	verified problems, logic puzzles and GSM8K and MATH training problems, answered with short reasoning; answers checked
Instruction following	3,743	prompts with checkable constraints; answers verified by code
Python tool use	2,946	real execution results inside the conversation
JSON and function calls	1,473	schema-checked
System prompts	1,469	
Chat with thinking	1,471	
Long documents	1,404	questions and summaries over documents up to 16K tokens
Safety	1,173	refusals and safe redirections
Identity	400	"I'm Laghu, a small language model made by Udgam Labs"

New answers came from DeepSeek-V4.1-Flash (MIT open weights) through its API, for \$9.49. Every row records its source, generator and licence. Conversations that overlap our held-out evaluation prompts were removed (8-gram check, plus exact checks on the multi-turn evaluation pairs).

5. Training

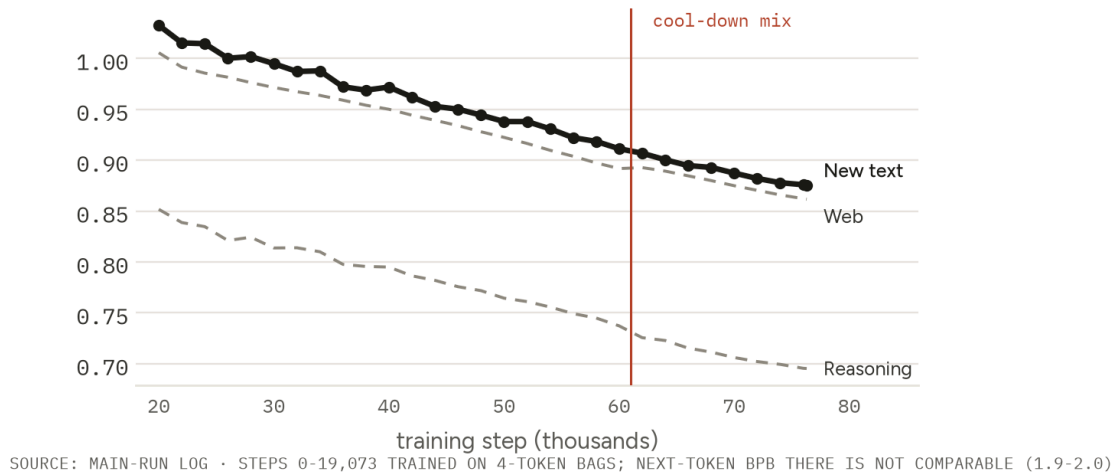
5.1 Pretraining

SETTING	VALUE
Steps × batch	76,294 × 524,288 tokens = 40.0B token positions, sequence 2,048
Optimiser	Muon for matrices (learning rate 0.02, cautious weight decay 0.28); Adam for embeddings (0.3), output (0.008) and scalars (0.5)
Schedule	40 warm-up steps; constant; linear warm-down over the last 65% of steps to 5% of the peak
Token bags	for the first 25% of steps each position reads a bag of 4 tokens, so about 70B tokens were read in all and the stable mix was read about 1.47 times
Data phases	stable mix (phase A) until 80% of steps, then the cool-down mix (phase B: less web, more SYNTH, maths, STEM solutions and chat)
Weight averaging	EMA from 90% of steps, β 0.998; the released base is the EMA
Precision	bf16 (FP8 was 25-27% slower at this width in a test run)
Hardware	8× A100 80GB SXM, 885-898K tokens per second after the token-bag phase; 13.2 hours; \$146.26

Training compute is about 3.6×10^{19} FLOPs ($6 \times \text{parameters} \times \text{token positions}$). The run had no loss spikes. Before it, three short dry runs (on an H100 and an L40S) checked speed, exact resume and the export path, and two data-mix gates compared candidate mixes on small models; the v3 mix used here was the pre-registered default.

Held-out bits per byte during pretraining

Lower is better · in-run scorer, every 2,000 steps from step 20,000



5.2 Long-context stretch

From the EMA weights at step 76,294: 1,900 steps × 524,288 tokens = 996M tokens at a 32,768-token context, on one RTX PRO 6000 (96 GB) in about 4 hours (75.9K tokens per second, 56 GB peak memory), using Colab compute units.

5.3 Supervised fine-tuning and the pick

From the 32K weights: sequences of 16,384 tokens (best-fit packing), batch 262,144 tokens, learning rates matrix 0.0071, embedding 0.106, output 0.0028, 5% warm-up and 50% warm-down to zero. Before fine-tuning we found that the 12 chat-format tokens, never seen in plain-text pretraining, had collapsed to identical embeddings; we re-drew them from the distribution of trained embeddings.

We trained 1, 2 and 3 passes over the data. Held-out loss favoured 1 pass (1.724 vs 1.784 for 2 and 2.036 for 3), but a pick rule written before generation favoured 2: a pairwise judge preferred 2 passes in 60.0% of 200 dev conversations (95% CI 54.9-65.0), and GSM8K-train accuracy with thinking was 7.0% vs 4.0%. The 2-pass replies were also shorter and looped less. Laghu is the 2-pass model.

5.4 Reinforcement learning (not used)

We tried GRPO on maths with on-policy distillation from Gargi (40 steps × 256 samples, learning rate 5e-6, KL 0.02), with a learnability gate written in advance. Dev GSM8K-train accuracy went from 8.6% to 8.2% against a bar of 11.6%, and the MATH loop rate rose from 10.5% to 19.5%. We stopped there; the released model has no RL.

6. Inference

6.1 Chat format and thinking

Laghu uses the Gemma 4 chat template. With `enable_thinking=True` it writes a thought block between `<|channel>thought` and `<channel|>`, then the answer, then `<turn|>`. Decode with special tokens kept and split on `<channel|>` to separate the two. Sampling defaults: temperature 0.7, top-p 0.95, top-k 64.

6.2 Runtimes, speed and memory

Laghu runs unchanged in transformers, llama.cpp (with Jinja templates), Ollama and MLX, and fully inside a browser tab through llama.cpp compiled to WebAssembly. We checked the chat format in each: thinking off and on, multi-turn conversations and tool calls. llama.cpp, Ollama and the MLX server turn thinking on by default; their switches are `--reasoning off`, `--think=false` and `--chat-template-args '{"enable_thinking":false}'`. llama-server without Jinja templates rejects requests, and it caps the context at 16,384 tokens.

ENGINE AND FILE (MAC MINI, APPLE M6, 24 GB)	PROMPT TOKENS/S	GENERATION TOKENS/S
llama.cpp, GGUF Q8_0 (Metal)	about 21,300	about 330
MLX 8-bit	about 23,800	about 456
MLX bf16	about 21,900	about 319

Memory in llama-server (Q8_0, one slot): 154 MiB at a full 4K context and 264 MiB at a full 16K context. The KV cache of the 21 sliding-window layers stays at 10.5 MiB at any length, which confirms the runtime applies the 512-token window; the 7 global layers take 28 MiB at 4K and 112 MiB at 16K.

In the browser, the page builds Laghu's prompt itself; it matches the transformers template on all 8 test conversations by text and token count. The 163 MB model downloads once; messages never leave the device.

7. Evaluation

7.1 Setup

Every comparison was measured by us, on the same hardware class (CUDA, RTX PRO 6000 on Colab), with the same scripts, items and decoding for every model, on 2026-10-19. Base models: lm-evaluation-harness 0.4.13, zero-shot (GSM8K 5-shot), bf16, `<bos>` added for our models. Chat models: vLLM, bf16, greedy, each model's own chat template, an output cap of 4,096 tokens (1,280 for IFEval, 1,024 for multi-turn and loops). Pairwise chat quality: DeepSeek-V4.1-Flash as judge, thinking off, temperature 0, both orders, length-controlled; 20 verdicts read by

hand. Confidence intervals are 95%; paired comparisons use McNemar or a paired bootstrap, with Holm correction across the nine pre-registered comparisons.

7.2 Base models

MODEL	PARAMS	ARC - E	ARC - C	HELLASWAG	PIQA	WINOGRANDE	5 - TASK AVG	MMLU	GSM8K 5 - SHOT
Laghu-Base	152M	60.6	30.1	40.9	65.7	52.2	49.9	25.3	4.2
GPT-X3-150M	149M	63.0	31.4	45.9	71.3	53.6	53.0	24.6	4.2
Boris-2-Preview-1	144M	63.3	29.8	45.2	69.2	52.6	52.0	23.8	4.8
SmoLLM2-135M	135M	64.5	30.0	43.0	68.3	53.0	51.8	24.5	2.0
cagliostro-v3.5	146M	58.8	29.2	43.4	67.7	51.5	50.1	24.7	6.9
Falcon-H1-Tiny-90M-Base	91M	54.3	28.5	37.7	64.8	50.8	47.2	32.4	2.7
MobileLLM-R1-140M-base	140M	52.7	24.3	34.1	63.1	50.4	44.9	24.3	19.7
Gargi-Base (ours)	384M	63.0	32.5	41.5	66.9	54.5	51.7	25.3	5.7

ARC-C, HellaSwag and PIQA use length-normalised accuracy. The full table of 19 models up to 200M and four near-size references is in the repository ([udgam3/evals/phase0/BASE_RESULTS_CUDA.md](#)). These benchmarks sit close to chance at this size, so we also measure held-out loss.

Held-out bits per byte. One script ([bpb_hf.py](#)), the same documents, a 2,048-token window, every token scored once; lower is better. "New text" is 1,695 English Wikipedia pages created between 1 September and 15 October 2026 and arXiv abstracts submitted 1-15 October 2026 (published as [udgamlabs/fresh-text-2026-10](#)).

MODEL	NEW TEXT	WEB (580)	REASONING (1,179)
Laghu-Base	0.919	0.858	0.693
SmoLLM2-135M	0.941	0.858	0.752
Falcon-H1-Tiny-90M-Base	0.979	0.910	0.769
Udgam-384M-Base	0.937	0.863	0.699
Gargi-Base (384M)	0.914	0.850	0.686
SmoLLM2-360M	0.850	0.788	0.678

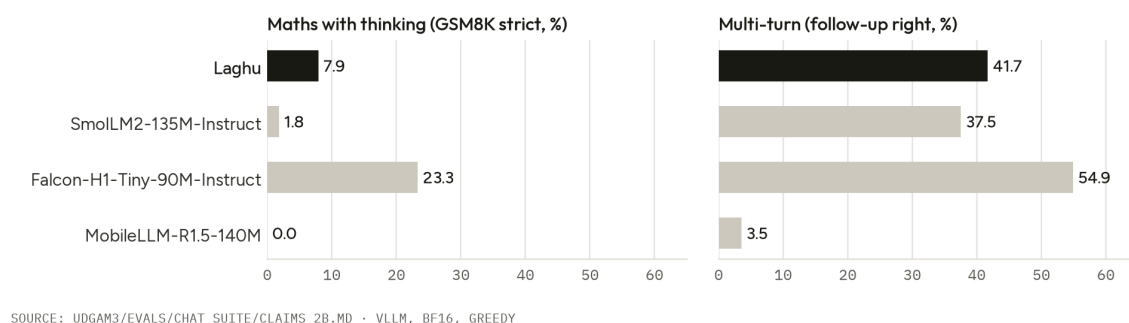
Paired bootstrap over documents (4,000 resamples): on new text SmoLLM2-135M is worse than Laghu-Base by 0.022 (95% CI 0.020-0.024), Falcon-H1-Tiny-90M-Base by 0.060 (0.057-0.062),

and Gargi-Base is better by 0.005 (0.004-0.006). On reasoning text SmoLLM2-135M is worse by 0.059 (0.056-0.062); on web text the two are level (-0.0001 , CI -0.0035 to $+0.0037$). The web and reasoning sets are held-out slices of our own data lineage and favour our models; the new-text set is the cleanest comparison.

7.3 Chat model: the pre-registered categories

Laghu against size-class rivals

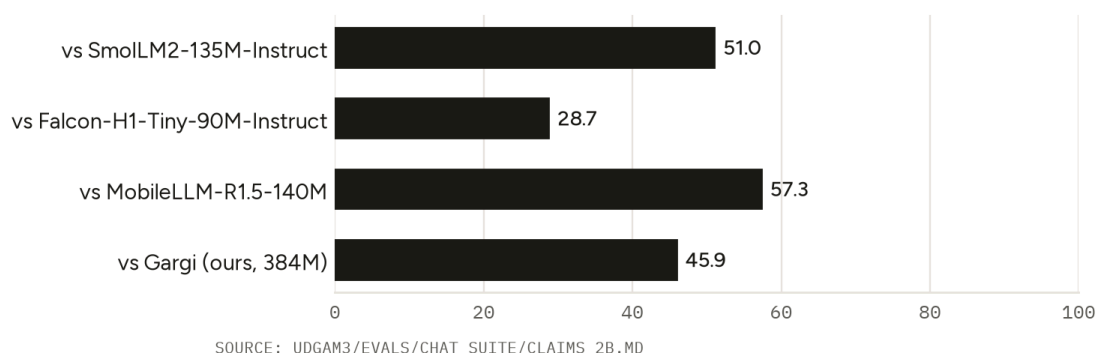
Higher is better · same harness, items and decoding for every model · 2026-10-19



CATEGORY (METRIC)	LAGHU	SMOLLM2-135M-INSTRUCT	FALCON-H1-TINY-90M-INSTRUCT	MOBILELLM-R1.5-140M
Maths with thinking (GSM8K test, strict: cut-off replies count as wrong)	7.9%	1.8%	23.3%	0.0%
Multi-turn (follow-up right, 144 checkable items)	41.7%	37.5%	54.9%	3.5%
Chat quality (Laghu's length-controlled win rate vs the rival)	—	51.0% [46.5, 55.5]	28.7% [24.0, 33.6]	57.3% [27.0, 76.2]

Chat quality: Laghu's win rate in pairwise judging

Length-controlled, 324 held-out prompts, both orders · 50% = level · judge DeepSeek-V4.1-Flash



After Holm correction Laghu is ahead of SmoLLM2-135M-Instruct and MobileLLM-R1.5 on maths, ahead of MobileLLM-R1.5 on multi-turn, and level with SmoLLM2-135M-Instruct on multi-turn and

chat quality. Falcon-H1-Tiny-90M-Instruct is ahead in all three. MobileLLM-R1.5 is a reasoning model that usually runs past the 4,096-token cap, which explains its low strict scores; its raw chat win rate of 91% is length-confounded, and the length-controlled 57.3% has a wide interval.

MobileLLM-R1.5-140M and MobileLLM-R1-140M-base results are reported for noncommercial research under the FAIR Noncommercial Research License; we acknowledge Meta's materials.

Against our own models: chat quality vs Gargi 45.9% ($p = 0.06$); multi-turn 41.7% vs Gargi 49.3% ($p = 0.12$) and vs Udgam-384M-Chat 40.3% ($p = 0.89$); maths 7.9% vs Gargi 35.6%.

7.4 Other chat measurements

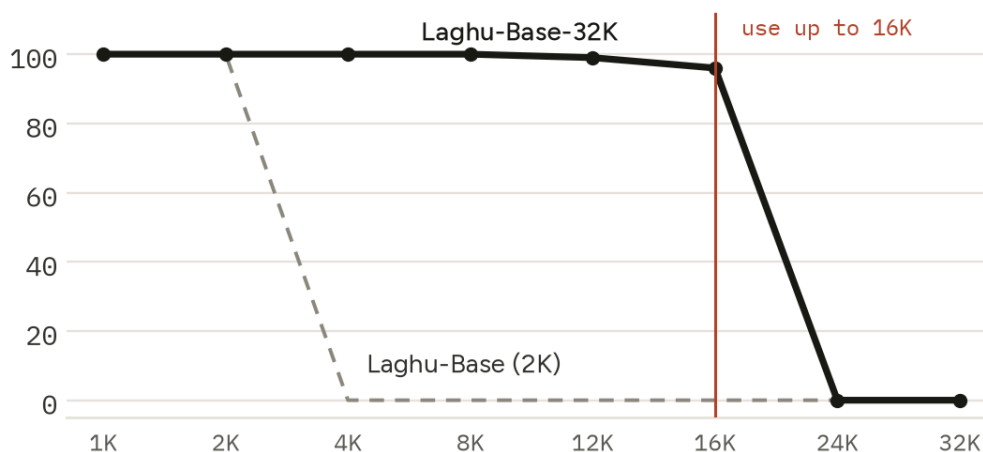
CHECK	LAGHU, THINKING OFF	LAGHU, THINKING ON	FALCON-H1-TINY- 90M-INSTRUCT	SMOLLM2-135M- INSTRUCT	GARGI (BEST MODE)
IFEval prompt-level strict	26.1	17.7	60.8	23.5	26.4
GSM8K (last number)	4.8	8.0	23.3	2.1	35.6
MATH-500 levels 1-3 (238 items)	5.5	6.3	23.9	1.7	19.3
GSM8K as the second turn	—	8.0 (8.0 alone)	27.0	1.0	37.0
Greedy loop rate (324 prompts)	8.3%	12.0%	0.3%	17.3%	11.7%
Loop rate with shipped sampling	1.2%	1.9%	0.0%	17.9%	2.5%

bf16 greedy decoding is not batch-invariant at this size: for Laghu, 32 of 40 check items gave identical first 512 tokens batched and one at a time. Treat differences of a point or two between models as noise.

7.5 Long context

Finding a hidden code in a long input

Recall %, 100 trials per length



SOURCE: UD GAM3/RUNS/CTX32K/RESULTS.MD

The stretch keeps bits per byte flat from 2K to 32K tokens on 400 long documents (0.92-0.94), while the 2K base rises to 1.39 by 16K. Its cost on short text was small: -0.76 points on the commonsense average (validation splits) and +0.40% (web), +0.82% (chat-formatted text), +1.10% (reasoning) and -0.04% (new text) in bits per byte at 2K.

7.6 Quantisation

Measured against the F16 GGUF on 32 windows of 2,048 tokens of held-out text (32,736 scored tokens), plus short task checks run through each runtime:

FILE	SIZE	PERPLEXITY	MEAN KL VS F16	TOP-1 AGREEMENT
GGUF F16	305.1 MB	52.24	0	100%
GGUF Q8_0	162.8 MB	52.11	0.0016	97.7%
GGUF Q6_K	153.7 MB	52.16	0.0054	96.0%
GGUF Q5_K_M	122.2 MB	53.87	0.041	88.6%
GGUF Q4_K_M	113.7 MB	55.70	0.080	84.7%
MLX bf16	303.6 MB	52.25	0.0009	98.0%
MLX 8-bit	161.4 MB	52.37	0.0040	96.2%
MLX 4-bit	85.6 MB	62.56	0.337	69.9%

We publish GGUF F16 and Q8_0 and MLX bf16 and 8-bit. Q8_0 and MLX 8-bit stay within noise of full precision (IFEval on 300 items: 25.7 and 21.7 vs 23.3 for F16, neither significantly different). The 4- and 5-bit files lose clearly. Because Laghu's width (640) is not a multiple of 256, llama.cpp's K-quants fall back to simpler types, so "Q4_K_M" is really about 5.9 bits per weight and barely smaller than Q8_0. The llama.cpp F16 file matches transformers almost exactly (mean KL 0.0000018).

8. Analysis

8.1 What a small budget buys

Laghu-Base read about 28 times fewer tokens than SmolLM2-135M (about 71B in all stages vs 2T in pretraining alone, as published) and predicts new text better than it (0.919 vs 0.941 bits per byte on our new-text set). On the commonsense benchmarks it is 1.9 points behind on average. After fine-tuning, the chat model is level with SmolLM2-135M-Instruct on judged quality and multi-turn and ahead on maths with thinking. We do not present the token ratio as unusual: other small models have also matched SmolLM2-135M on far fewer tokens (GPT-X3-150M, for example, reports 27× fewer). Token counts use each model's own tokenizer, so the ratios are approximate.

8.2 What limits Laghu

	FALCON-H1-TINY-90M	LAGHU (152M)
Pretraining tokens	800B (as reported by TII)	40.0B positions (about 70B tokens read)
Training compute (6·N·D)	$\approx 4.4 \times 10^{20}$	$\approx 3.6 \times 10^{19}$
Instruction data	about 10B tokens of SFT, or SFT data mixed into pretraining (as reported)	about 19M trained tokens (2 passes over 41K conversations)
Preference tuning	DPO (as reported)	none
Architecture	hybrid Mamba and attention	attention (Gemma 4 layout)

The model that sets every bar in our chat suite differs from Laghu mostly in post-training scale and a preference stage, and in about 12 times the pretraining compute. Our own 384M Gargi, with about 17B training tokens, scores far higher on maths (35.6%) because it was trained on verified maths traces at scale. We read this as: at 150M, the gap to the best small chat models is post-training and data, not architecture.

8.3 What thinking buys

Thinking on raises GSM8K from 4.8% to 8.0% but lowers IFEval (26.1 → 17.7) and multi-turn (41.7% → 32.6%), and loops more. The thought blocks use the right format but the arithmetic inside is often wrong even when the final answer is right. We recommend thinking off except for short arithmetic.

9. Pre-registration and the release decision

We wrote the evaluation protocol before training (17 October 2026) and the claims in two stages before the matching results were measured: base-model claims (none; all reported) before Laghu's base suite ran, and chat claims at 18:59 IST on 19 October before any Laghu chat result. The chat claims named three categories (maths with thinking, multi-turn, chat quality), three size-class rivals, the statistics (Holm over nine comparisons), a loop-rate limit ($\leq 10\%$) and an own-model bar, and set the release rule: beat every rival in all three categories. At 19:04 IST, still before measuring, we changed the chat-quality judge from gpt-oss-120b to DeepSeek-V4.1-Flash because our account could not reach the pre-registered judge. The rival measurements were not finished before pretraining started, which the amendment log records.

Laghu passed the loop limit (8.3%) and the own-model bar, and won 0 of 3 categories, because Falcon-H1-Tiny-90M-Instruct is ahead in each. On 21 October we decided to release it anyway as a fully open research release with no best-in-class claim, without changing any result. The pre-registration and its amendment log are in the repository (`udgam3/PREREGISTRATION.md`).

10. What did not work

1. **Reinforcement learning** (section 5.4) did not improve maths and doubled loops on MATH. At single-digit accuracy most GRPO groups have no correct sample, and distillation copied the teacher's longer style rather than its reasoning. This was our third RL failure below 400M parameters.
2. **A leaky SFT validation split.** Splitting rows by id put related rows (thinking on/off twins, several questions about one document, multi-turn rows from one seed) on both sides; about 37% of eligible validation conversations had a near-copy in training. The SFT pick used 200 separate leak-free conversations instead. Future splits key on content.
3. **Held-out loss chose the wrong SFT length** (1 pass), as InstructGPT reported for larger models; a generation-based rule written in advance chose 2.
4. **The 32K stretch** raised reasoning-text bits per byte by 1.10%, just over the 1% limit we had set; we accepted it.
5. **Lost checkpoints.** The pretraining checkpoints saved every 2,000 steps (steps 2,000-74,000) and about six earlier 32K checkpoints existed only as older versions of files that

our uploader overwrote. A storage clean-up on 21 October 2026 that squashed file histories deleted them. Only steps 76,000 and 76,294 (pretraining) and 1,750 and 1,900 (32K) survive. Checkpoints are now saved under step-numbered names in a folder nothing prunes.

- 6. **The main run's first start failed** at step 0 (a validation file had one row group for eight GPUs; about \$1.90), and a monitor on the Mac crashed on a network stall, which hung the experiment-tracking sync for 5.5 hours (backfilled from the run log).
- 7. **Below projection.** Our scaling projection expected 0.900-0.912 bits per byte on new text after the 32K stretch; the result was 0.921.

11. Cost

ITEM	CASH
Main pretraining run (Shadeform, 8× A100 80GB, 13.2 h at \$11.04/h, including the failed start)	\$146.26
Data build, data-mix gates, dry runs (Runpod, 13 machines)	\$36.01
Cloud smoke tests (Shadeform, 2)	\$1.43
DeepSeek API: SFT data \$9.49, SFT pick judge \$0.07, chat-quality judge \$0.45	\$10.01
X search for research	\$0.03
Total	about \$194

The 32K stretch, SFT, RL and evaluations ran on Colab compute units included in a subscription (about 60 units). Money that produced no kept result (failed starts, stopped runs) was about \$5, 2.6% of the total.

12. Limitations and safety

Laghu is a small research model. It knows little and states wrong facts confidently; its maths and instruction following are weak; it can loop with greedy decoding; and it works with up to 16K tokens of input. It was trained on English only. Its safety training is light (1,173 conversations), so it can produce wrong, biased or harmful text, and it is not suitable for advice or decisions that matter. The base models have no safety tuning.

13. Release details

Weights and our code are Apache-2.0 (copyright Udgam Labs LLP); files from nanochat keep their MIT notice; the chat template is Google's Gemma 4 template (Apache-2.0). Training data keeps each source's licence; the repository's `docs/DATA_LICENSES.md` lists every source, its licence and the credit it asks for. The SFT data is re-hosted with per-row source, generator and licence; the pretraining and 32K data are rebuild scripts. Evaluation outputs are published per item for every model whose licence allows it, and as scores for the rest.

14. Conclusion

For \$146 of pretraining, a 152M model trained from scratch on about 71B tokens can predict new text better than models trained on 800B to 2T tokens, and be level in chat with SmoLLM2-135M-Instruct. It is not the best small chat model; a model trained with 12 times the compute and a preference stage is ahead. We release every part of Laghu so that others can study, reproduce and improve it.

15. References

1. Google. Gemma 4 model family and chat template (google/gemma-4-E2B-it), 2026.
2. A. Karpathy. nanochat. github.com/karpathy/nanochat, 2025.
3. Hugging Face. SmoLLM2 model cards (HuggingFaceTB/SmoLLM2-135M, -Instruct), 2024.
4. Technology Innovation Institute. Falcon-H1-Tiny technical blog (huggingface.co/spaces/tiiuae/tiny-h1-blogpost), January 2026.
5. Meta FAIR. MobileLLM-R1.5-140M and MobileLLM-R1-140M-base model cards. Released under the FAIR Noncommercial Research License; we report their results for noncommercial research and acknowledge Meta's materials.
6. L. Ouyang et al. Training language models to follow instructions with human feedback (InstructGPT), 2022.
7. N. Muennighoff et al. Scaling data-constrained language models, 2023.
8. EleutherAI. lm-evaluation-harness, version 0.4.13.

About Udgam Labs. Udgam Labs is an independent AI research lab. We work from first principles: before we build, we understand why things work, whether we are training a model, fine-tuning one or applying AI to a new problem. Udgam (उद्गम) is Sanskrit for origin. udgamlabs.com