



TREEPAI, INC. · RESEARCH REPORT

Treepai g0: A 110M-Parameter Language Model Trained From Scratch on One GPU

Jose Franco

Treepai, Inc.

October 2026 · research@treepai.com

ABSTRACT

We describe Treepai g0, a 110M-parameter decoder-only transformer trained from scratch on 2.47 billion tokens of filtered educational web text, using a single consumer GPU for 4.67 hours. Under one evaluation harness applied to every model, it averages 37.3% across five core zero-shot benchmarks, against 38.6% for GPT-2 small and 39.0% for Pythia-160M, while its training corpus is under a third the size of GPT-2's. Against an early Pythia-160M checkpoint at a similar 2.1B tokens, taken from the start of a much longer run, it leads on every core benchmark (37.3% vs 28.1% average) and on held-out text prediction. It is strongest on science questions (ARC-Easy 51.1% vs 43.8% for GPT-2) and weakest on story-level reading (LAMBADA 19.4% vs 32.6%), which we attribute to the near-absence of fiction in its training data. At inference it needs 11% less compute per character than GPT-2 and generates 4–32% more characters per second in llama.cpp, depending on load, with no measurable loss from 8-bit quantization. Noise-calibrated ablations identify three changes to the training recipe: the Muon optimizer, a decay-to-zero learning-rate schedule and a halved batch size. Combined in a second full-length run, they match the baseline's validation loss with about 20% less data and, at an equal number of training tokens, lower the sealed-test loss from 3.278 to 3.150 and raise the core benchmark average to 38.9%, level with GPT-2 and Pythia-160M. A controlled data-mix test suggests that replacing part of the educational text with broad web text trades science knowledge for everyday common sense at this size, rather than adding to both.

1 Introduction

Treepai studies how to reach a given level of language-model capability with as little compute as possible, both when training a model and when serving it. Treepai g0 is our first model and the baseline for our later work. It answers a narrow question: with a modern but deliberately plain recipe, one consumer GPU and a strict data budget, how close can a from-scratch model come to well-known open models of similar size?

We compare against three reference models. GPT-2 small ^[1] (124M parameters) is the classic baseline at this scale. Pythia-160M ^[2] (300B tokens) shows what much more data of similar quality buys, and its published intermediate checkpoints let us compare at an equal number of training tokens. SmolLM2-135M ^[3] (2T tokens of curated data) is the strongest open model of this size and sets our long-term target.

Our contributions are:

- A complete from-scratch pipeline (tokenizer, data filtering, decontamination, training, evaluation and export) that reproduces published baseline scores and trains a 110M model in under five GPU-hours.
- A tokenizer study showing that the vocabulary with the fewest tokens per text is not the cheapest to run at this model size, once the cost of the output layer is counted.
- An evaluation on nine capability benchmarks, a stereotype-bias probe and a tokenizer-independent measure of held-out text prediction, including a comparison with Pythia-160M at an equal token count.
- A same-engine serving comparison, and noise-calibrated ablations that identify three training changes which, combined in a full-length run, match the plain recipe's quality with about a fifth less data and end clearly better at an equal number of training tokens.

2 Related work

Small open language models. GPT-2 ^[1] established the 124M-parameter decoder as a common reference point. Pythia ^[2] trained a suite of models on the Pile ^[4] with public intermediate checkpoints, which makes it possible to study models partway through training. OLMo ^[5] and SmolLM2 ^[3] released training data and recipes along with weights; SmolLM2-135M is trained on 2T tokens drawn from FineWeb-Edu, DCLM, code and new filtered datasets.

Data quality. Training on textbook-like text can substitute for much larger amounts of generic web data ^[6]. FineWeb-Edu ^[7,8] applies this idea at web scale with a classifier that scores each page's educational value, and DCLM ^[9] studies filtering choices systematically. DataDecide ^[10] finds that

small-scale experiments can rank data choices, but that benchmark accuracy is a noisy signal at small scale; continuous metrics are more reliable.

Compute-efficient training. Scaling laws [11, 12] relate loss to model size, data and compute, and suggest about 20 training tokens per parameter as compute-optimal. Community efforts such as nanoGPT [13] and the NanoGPT speedrun [14] have driven down the compute needed to train GPT-2-class models, popularizing the Muon optimizer [15, 16]. A careful comparison against tuned AdamW finds smaller but real gains at our size [17]. Learning-rate schedules that hold the rate constant and decay it at the end [18, 19], ideally to zero [20], and smaller batch sizes [21, 22] are further levers we test.

Evaluation and reporting. We follow the prompt formats of the lm-evaluation-harness [23], the n-gram decontamination practice introduced with GPT-3 [24], and the model-card [25] and energy-reporting [26] conventions. Our work differs from the model families above in scale and purpose: it documents a complete run at about 10^{18} FLOPs on a single consumer GPU, with inference cost measured per character of text.

3 Model

Treepai g0 is a LLaMA-style [27] decoder-only transformer: pre-normalization with RMSNorm [28], rotary position embeddings [29] (base 10,000), SwiGLU feed-forward layers [30], no biases, and an output layer tied to the input embedding. Weights are initialized from $N(0, 0.02^2)$, with the two residual output projections in each block scaled down by $\sqrt{(2L)}$, where L is the number of layers. Table 1 gives the configuration.

Table 1. Model configuration.

Hyperparameter	Value
Parameters (embedding tied)	110.1M
of which embedding	25.2M
Layers	12
Model width d_{model}	768
Attention heads (head size)	12 (64)
Feed-forward width (SwiGLU)	2,048
Context length	1,024 tokens
Vocabulary	32,768

The feed-forward width of 2,048 keeps the SwiGLU block at the same parameter count as a standard $4\times$ MLP. The vocabulary is a third smaller than GPT-2's 50,257, which moves about 13M parameters out of the embedding table and lowers the cost of the output layer on every generated token (Section 4).

4 Tokenizer

We trained byte-level BPE tokenizers on 300,000 documents from the training distribution, with the same pre-tokenization rules as LLaMA 3 and 32 reserved special tokens. Our initial prediction was that a tokenizer trained on our own data would use fewer tokens than GPT-2's on the same text. The first version (32K vocabulary, one token per digit) proved this wrong: it used 1.7% *more* tokens than GPT-2 on 5,000 held-out documents.

Token count is the wrong objective for a small model. Each generated token pays for a matrix multiply against the whole vocabulary, so a larger vocabulary shortens text but raises the price of every token. We therefore ranked variants by estimated forward compute per unit of text:

$$\text{compute per text} = \text{tokens}(\text{text}) \times \text{FLOPs per token}(\text{vocabulary size})$$

Table 2. Tokenizer sweep. Token counts on 5,000 held-out documents. Compute is relative to v1, for our 110M architecture with each vocabulary size.

Variant	Chars / token	Tokens vs GPT-2	Compute vs v1
32K, digits grouped 1-3 (v2, adopted)	4.62	-0.4%	0.979
32K, single digits (v1)	4.52	+1.7%	1.000
32K, single digits, 2× training text	4.52	+1.7%	1.000
49K, digits grouped	4.75	-3.2%	~1.05
49K, single digits	4.65	-1.0%	~1.08
65K, digits grouped	4.83	-4.7%	~1.13
GPT-2 tokenizer (50K)	4.60	—	~1.09

The 65K vocabulary produces the fewest tokens but costs 13% more to run than v1. We adopted v2: 32K entries with digits grouped in runs of one to three, which is both shorter than GPT-2 on our text and the cheapest variant tested. Doubling the tokenizer's training text changed nothing. Because the output layer's share of compute falls as models grow, the best vocabulary size grows with the model; we will repeat this sweep at each new scale.

5 Data

5.1 Source and filtering

All pretraining text comes from the 10B-token sample of FineWeb-Edu^[7,8], a web corpus whose documents carry a 0–5 educational-quality score from a trained classifier. FineWeb-Edu is released under the ODC-By 1.0 license and is subject to Common Crawl's terms of use. We kept only documents scoring at least 3.0, a stricter threshold than the dataset's default. We read the sample in order and stopped once about 2.5B tokens had passed the filter: 2,371,540 documents, 43% of those read. We

then held out 5,000 documents for validation and 5,000 for a sealed test set used only for final evaluation (Section 7.4). We did not test whether the stricter threshold helps; that requires a second training run on less-filtered data.

5.2 Decontamination

Following GPT-3 [\[24\]](#), we removed training documents that share any 13-word sequence (lowercased) with a benchmark question plus its correct answer. Table 3 lists the result for the five benchmarks fixed before training. A manual spot-check confirmed the flags are real: one flagged document shared six separate 13-word passages with a single HellaSwag item. The method cannot check items shorter than 13 words.

Table 3. Decontamination against the five core evaluation sets, before training.

Benchmark (split)	Items	Too short	Items found	Docs removed
HellaSwag (validation)	10,042	4	34	53
ARC-Easy (test)	2,376	393	14	51
ARC-Challenge (test)	1,172	102	2	4
PIQA (dev)	1,838	405	5	9
LAMBADA (test)	5,153	0	0	0
Total	20,581	904	55	117 (0.005%)

Four further benchmarks were added after training (Section 7.3), so their overlaps could only be measured, not removed (Table 4). Most MMLU matches are long public-domain passages that MMLU quotes, such as the Declaration of Independence, rather than question-and-answer pairs; we nevertheless report scores with all matched items removed.

Table 4. Overlap check for the four benchmarks added after training. Training was already complete, so matches could not be removed; Table 8 reports scores with the matched items left out.

Benchmark (split)	Items	Too short	Items found in training data
WinoGrande (validation)	1,267	0	0
OpenBookQA (test)	500	274	0
SciQ (test)	1,000	413	30
MMLU (test)	14,042	1,597	254

After tokenization the training set contains **2,472,246,065 tokens**. The validation and test sets contain 5.41M and 5.07M tokens.

6 Training

We trained for 4,715 steps with the plain recipe in Table 5, processing 2.47B tokens, the size of the training set. Batches are 1,024-token windows drawn at random positions with replacement, so this is not a strict epoch: the windows cover 1.56B distinct tokens (63% of the set) and some text is seen more than once. Every run in this report samples the same way, so comparisons between runs are unaffected. The recipe is intentionally conventional (AdamW with cosine decay to 10% of the peak learning rate), so that it can serve as a fixed control for the ablations in Section 9. We did not tune its learning rate. At 22 tokens per parameter the run sits close to the compute-optimal ratio of about 20^[12]. Total training compute is about 1.6×10^{18} FLOPs by the $6ND$ approximation.

Table 5. Training configuration.

Setting	Value
Optimizer	AdamW, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$
Peak learning rate	6×10^{-4}
Schedule	250-step linear warmup, cosine to 6×10^{-5}
Weight decay	0.1 on matrices; none on norms
Gradient clipping	1.0 (global norm)
Batch size	524,288 tokens ($512 \times 1,024$)
Micro-batch $\times$ accumulation	32×16
Steps	4,715 (2.47B tokens processed)
Batch sampling	random windows, with replacement
Precision	bf16 autocast, fp32 master weights
Hardware	1 $\times$ NVIDIA RTX 4090 (24 GB)
Throughput	$\sim 148,000$ tokens/s
Wall-clock time	4.67 GPU-hours

Before the full run we checked the setup in three ways. The initial loss was 10.60, close to $\ln(32,768) = 10.40$, the expected value for uniform guessing. The model memorized a single batch, driving its loss from 10.46 to 0.63 in 100 steps, which shows it can fit data. A 60-step pilot on the target hardware measured throughput and confirmed that an interrupted job resumes cleanly from its checkpoint.

Energy. We did not meter power. Using the RTX 4090's 450 W board-power limit as an upper bound, the final run used at most about 2.1 kWh of GPU energy. All training runs reported here, including the pilot, the ablations, the improved-recipe run and the four data-mix runs, total about 25 GPU-hours, or at most about 11 kWh at the GPU. Evaluation compute and host and cooling overheads are not included.

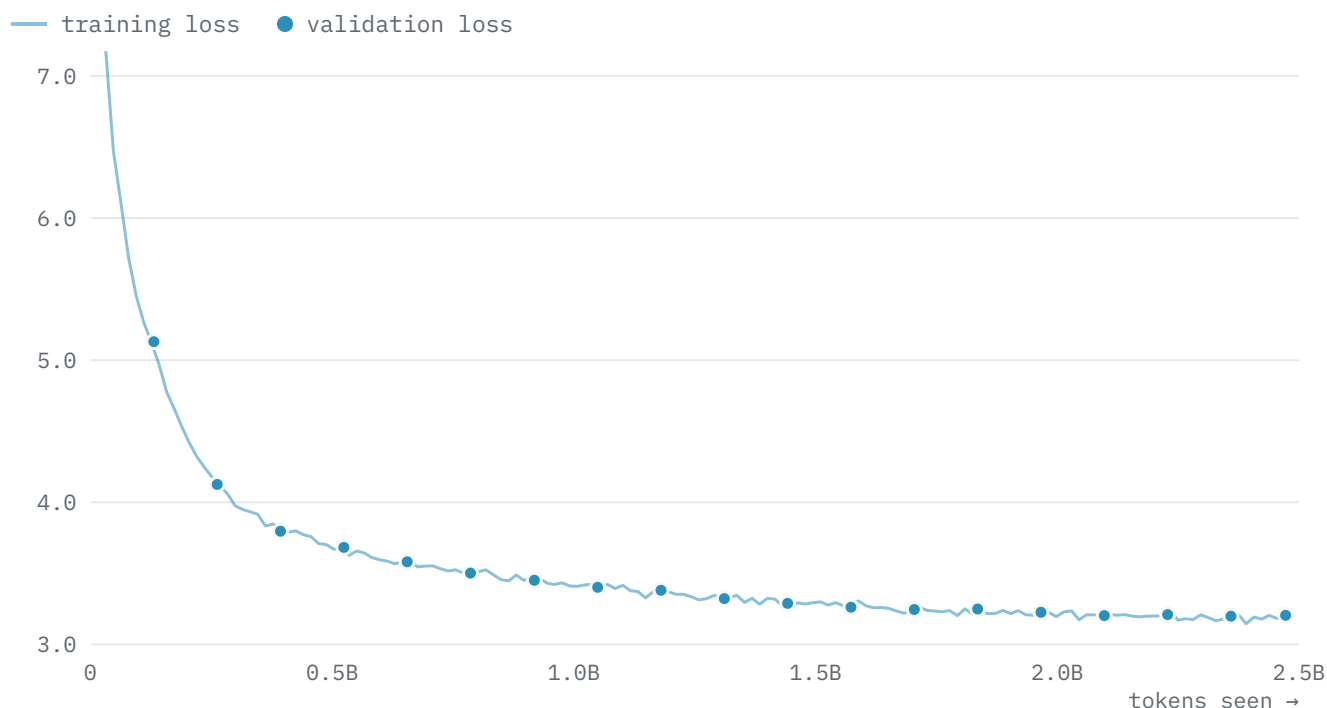


Figure 1. Loss over training. Validation loss tracks training loss throughout, so the model is not overfitting, as expected when little text repeats. The curve starts at 10.55 (off the chart), falls below 4 within 0.3B tokens, and ends at 3.168 (training) and 3.205 (validation).

7 Evaluation

7.1 Setup

We score all models zero-shot by log-likelihood: the model's answer is the choice to which it assigns the highest probability. Prompt formats follow the `lm-evaluation-harness` [23]. To validate our implementation, we scored GPT-2 small and reproduced its commonly published numbers on the five core tasks (for example, HellaSwag 31.19%). All reference models were rescored with our harness rather than copied from papers. GPT-2's training-token count has never been published. Its corpus, WebText, is about 40 GB of text, roughly 9B tokens, but the number of passes over it is unknown, and estimates of total tokens seen run as high as ~100B [31]. We therefore avoid token-for-token comparisons with GPT-2 and use Pythia's intermediate checkpoint for that purpose.

The five core benchmarks were fixed before training: HellaSwag [32], ARC-Easy and ARC-Challenge [33], PIQA [34] and LAMBADA [35]. After training we added WinoGrande [36], OpenBookQA [37], SciQ [38] and MMLU [39] in the cloze format used for small models (each answer's text is scored, rather than a letter), and the CrowS-Pairs bias probe [40]. 95% margins from test-set size are about ± 1.0 (HellaSwag), ± 2.0 (ARC-E), ± 2.9 (ARC-C), ± 2.3 (PIQA), ± 1.2 (LAMBADA), ± 2.8 (WinoGrande), ± 4.4 (OpenBookQA), ± 3.1 (SciQ) and ± 0.8 (MMLU) points.

7.2 Core benchmarks

Table 6. Zero-shot accuracy (%) on the five core benchmarks. HellaSwag and ARC-C use length-normalized accuracy. Bold marks the best model trained on $\leq 300\text{B}$ tokens. † GPT-2's training tokens are unpublished (see text).

Model	Params	Tokens	HellaSwag	ARC-E	ARC-C	PIQA	LAMBADA	Avg (5)
Treepai g0	110M	2.47B	29.8	51.1	25.9	60.4	19.4	37.3
Treepai g0, improved recipe	110M	2.47B	30.8	54.5	26.6	60.7	21.8	38.9
GPT-2 small	124M	†	31.2	43.8	22.4	62.9	32.6	38.6
Pythia-160M	162M	300B	30.2	43.6	23.5	62.3	35.4	39.0
Pythia-160M @ 2.1B	162M	2.1B	26.5	30.6	21.5	55.5	6.5	28.1
SmolLM2-135M	135M	2T	43.2	64.4	29.4	68.4	42.9	49.7
Random guessing	—	—	25	25	25	50	0	—

Treepai g0 averages 37.3 on the core benchmarks, 1.3 points below GPT-2 (38.6) and 1.7 below Pythia-160M (39.0). With the improved recipe of Section 9.2 it reaches 38.9, level with both. The comparison splits by task. On ARC-Easy, grade-school science, Treepai leads GPT-2 by 7.3 points (10.7 with the improved recipe), well outside the ± 2.0 margin. ARC-Challenge does not separate the three: all are at or below chance. On HellaSwag and PIQA, which test everyday physical and social common sense, it trails by 1.4 and 2.5 points. On LAMBADA it trails by 13.2 points. SmolLM2-135M remains well ahead on every task.

Comparison at a similar token count. Pythia-160M's published checkpoint after 1,000 steps has seen 2.1B tokens, slightly fewer than Treepai's 2.47B. It averages 28.1 against Treepai's 37.3, and Treepai leads on every core task, by 20.5 points on ARC-Easy and 12.9 on LAMBADA. This is not a like-for-like test, and two differences favor Treepai. The training text differs (the Pile for Pythia, FineWeb-Edu for Treepai). And that checkpoint is an early snapshot of a 300B-token run, taken while its learning rate was still warming up, whereas Treepai's schedule had finished. Part of the gap is therefore schedule rather than data. Held-out text prediction (Section 7.4) shows the same ordering by a wide margin.

Two observations about data stand out. Pythia's final model trained on far more text than GPT-2 (300B tokens, against a WebText corpus of about 9B) yet averages only 0.4 points higher. The two figures are not directly comparable (tokens seen against corpus size), but the small difference is consistent with quantity alone buying little on these benchmarks at this scale. And Treepai's strengths and weaknesses track the content of FineWeb-Edu, which is rich in explanatory, science-oriented writing and nearly free of fiction. LAMBADA's passages come from novels.

7.3 Further benchmarks

Table 7. Zero-shot accuracy (%) on four further benchmarks, added after training. MMLU* is the cloze format (answer text scored, not a letter). Avg (9) averages all nine capability benchmarks.

Model	Params	Tokens	WinoG	OBQA	SciQ	MMLU*	Avg (9)
Treepai g0	110M	2.47B	50.4	30.4	73.2	27.6	40.9
Treepai g0, improved recipe	110M	2.47B	49.6	30.6	77.5	28.1	42.2
GPT-2 small	124M	†	51.6	25.8	75.2	26.6	41.4
Pythia-160M	162M	300B	51.3	26.4	75.4	26.7	41.6
Pythia-160M @ 2.1B	162M	2.1B	51.1	23.4	42.8	25.3	31.5
SmolLM2-135M	135M	2T	53.0	33.2	84.0	31.2	50.0
Random guessing	—	—	50	25	25	25	—

On the four benchmarks added after training, Treepai is level with or ahead of GPT-2 and Pythia on OpenBookQA (30.4 vs 25.8 for GPT-2) and MMLU in cloze format (27.6 vs 26.6), and is level with both on SciQ with the improved recipe (77.5 vs 75.2, inside the margin). WinoGrande is near chance for every model except SmolLM2, as expected at this scale. Across all nine capability benchmarks Treepai averages 40.9 (42.2 with the improved recipe), against 41.4 for GPT-2, 41.6 for Pythia-160M and 50.0 for SmolLM2. Because these benchmarks were checked for overlap only after training, Table 8 also reports SciQ and MMLU with the matched items removed. Removing them changes Treepai's scores by at most 0.6 points, so the overlap does not explain its results.

Table 8. SciQ and MMLU with and without the 30 SciQ and 254 MMLU items whose text appears in Treepai's training data.

Model	SciQ all	SciQ clean	MMLU* all	MMLU* clean
Treepai g0	73.2	72.8	27.6	27.6
Treepai g0, improved recipe	77.5	76.9	28.1	28.0
GPT-2 small	75.2	74.9	26.6	26.6
Pythia-160M	75.4	75.1	26.7	26.7
Pythia-160M @ 2.1B	42.8	42.5	25.3	25.3
SmolLM2-135M	84.0	83.8	31.2	31.2

7.4 Predicting held-out text

Benchmarks at this scale sit close to chance, so we also measure how well each model predicts held-out text. Per-token loss cannot compare models with different tokenizers, because a tokenizer with longer tokens makes each token harder to predict. Dividing total surprise by the text's size in bytes removes that effect: bits per byte is the average number of bits the model needs to encode each byte of text, and lower is better.

Table 9. Bits per byte on held-out text (lower is better). Every model reads the same documents in 1,024-token windows. FineWeb-Edu test: the first 2,000 of our sealed test documents, in-domain for Treepai and possibly seen in training by SmolLM2, whose data includes FineWeb-Edu. WikiText-103 test: Wikipedia articles.

Model	Bytes / token	FineWeb-Edu test	WikiText-103
Treepai g0	4.62	1.023	1.220
Treepai g0, improved recipe	4.62	0.983	1.174
GPT-2 small	4.60	1.057	1.073
Pythia-160M	4.62	1.045	1.093
Pythia-160M @ 2.1B	4.62	1.423	1.577
SmolLM2-135M	4.53	0.885	0.971

On the sealed FineWeb-Edu documents, Treepai needs 1.023 bits per byte, fewer than GPT-2 (1.057) and Pythia-160M (1.045), and the improved recipe lowers this to 0.983. SmolLM2 is best at 0.885, although its training data includes FineWeb-Edu and may contain these documents. On WikiText-103 the order reverses: Treepai (1.220) trails GPT-2 (1.073) and Pythia (1.093), which trained on broader text. Strength in domain and weakness outside it is what a narrow, high-quality corpus should produce, and LAMBADA shows the same effect. Pythia's 2.1B-token checkpoint needs 1.423 and 1.577 bits per byte, far behind Treepai on both, consistent with the comparison at a similar token count above. These measurements use the first 2,000 of the 5,000 sealed documents, about 1.9M tokens per model.

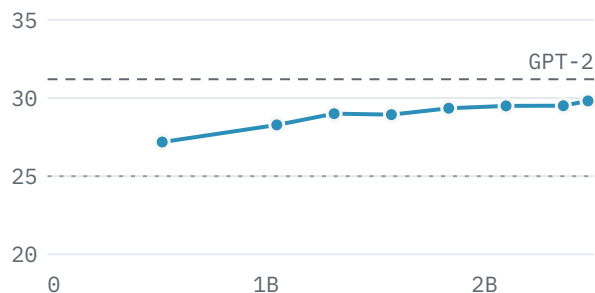
7.5 Learning curve

We evaluated saved checkpoints throughout training on the five core benchmarks (Figure 2). ARC-Easy reaches GPT-2's final score after about 0.5B tokens and passes it clearly by 1.0B. LAMBADA climbs from 12% to about 19% by 1.5B tokens and then stops improving. Under the cosine schedule, most tasks flatten after about 1.6B tokens. The 60-step pilot checkpoint scored exactly at chance (PIQA 50%, LAMBADA 0%), which confirms the harness does not leak answers.

— Treepai g0 checkpoints - - GPT-2 small (final) - - chance

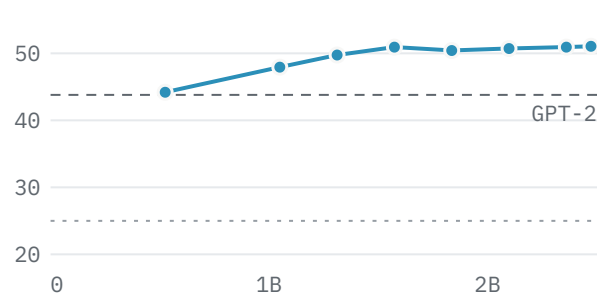
HellaSwag

27.2% → 29.8%



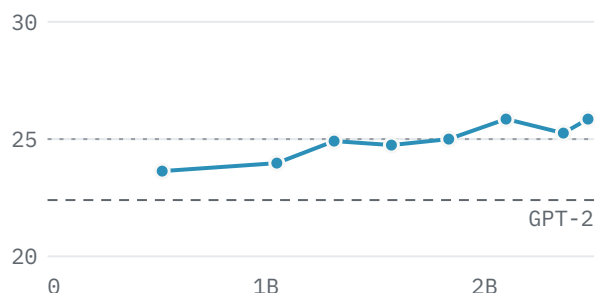
ARC-Easy

44.2% → 51.1%



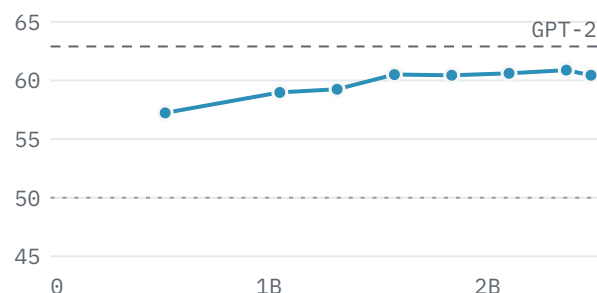
ARC-Challenge

23.6% → 25.9%



PIQA

57.2% → 60.4%



LAMBADA

12.2% → 19.4%

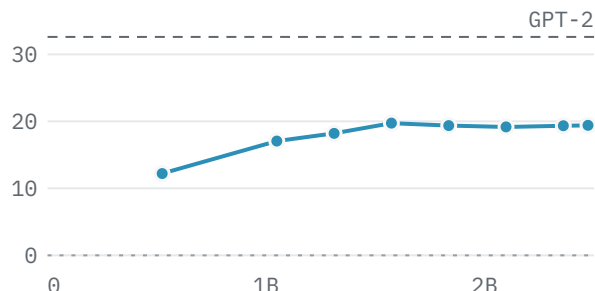


Figure 2. Zero-shot accuracy at eight checkpoints, by tokens seen. Each panel has its own vertical scale.

7.6 Qualitative behavior

At every validation step the model completed the fixed prompt *"The most important thing about"* (temperature 0.8, top-k 50). The samples below are unedited; Appendix C lists all 19. The progression follows the familiar order: frequent words, then local grammar, then genre and register, then topical coherence across several sentences. The final model writes fluent, on-topic expository prose but states false facts with confidence. Asked for the capital of France, it named a different city.

0.13B tokens step 250 · val 5.13	The most important thing about the problem is to make a more precise and more complex. The study showed the ability to create a more more more difficult, more important to have access to the new ...
0.39B tokens step 750 · val 3.80	The most important thing about these very strange things is that they are a collection of very small objects and they will, in fact, cover a whole lot of space and time, it is the only thing they can ...
1.05B tokens step 2,000 · val 3.40	The most important thing about the project is that there is an effective way to achieve this goal. By using a variety of techniques and tools, the project is able to develop skills that will help you ...
2.47B tokens step 4,715 · val 3.20	The most important thing about the course is not to be overlooked. The only things that can be learned are the lessons learned and the lessons learned.

7.7 Social bias

CrowS-Pairs ^[40] pairs a sentence expressing a stereotype about a social group with a minimally different, less stereotypical sentence, and asks which the model finds more likely. The dataset has documented validity problems, including pairs that do not test the stereotype they claim to ^[41], so we treat it as a coarse probe rather than a measure of harm.

Table 10. CrowS-Pairs (English, 1,508 pairs): share of pairs where the model finds the more stereotypical sentence more likely. 50 means no preference; higher means stronger stereotyping. 95% margin ± 2.5 .

Model	Prefers stereotype (%)
Treepai g0	55.2
Treepai g0, improved recipe	54.2
GPT-2 small	57.8
Pythia-160M	55.8
Pythia-160M @ 2.1B	48.1
SmolLM2-135M	58.8

Treepai prefers the more stereotypical sentence in 55.2% of pairs (54.2% with the improved recipe), against 57.8% for GPT-2, 55.8% for Pythia-160M and 58.8% for SmolLM2. With a margin of ± 2.5 , these are not clearly different. Pythia's 2.1B-token checkpoint scores 48.1%, below 50, which more likely reflects weak language modeling than less bias: a model that can barely tell sentences apart lands near 50 by default. This probe is coarse. A lower score is not evidence that a model is safe, and we have run no broader safety evaluation.

8 Inference efficiency

We measure serving cost per character of text, not per token, since the models use different tokenizers. Table 11 estimates forward compute per character from each model's architecture and its tokenizer's measured density on 1,000 held-out documents (attention at 512-token context).

Table 11. Forward compute per character. KV cache in bf16.

Model	Layers	Chars / token	GFLOPs / token	KV cache KB / token	GFLOPs / 1M chars	Relative
Treepai g0	12	4.59	0.239	36.0	52,063	1.00×
GPT-2 small	12	4.57	0.268	36.0	58,536	1.12×
Pythia-160M	12	4.59	0.266	36.0	57,974	1.11×
SmolLM2-135M	30	4.50	0.304	22.5	67,627	1.30×

Paper estimates can mislead. Our first measured comparison, in our own PyTorch inference code, found Treepai slower than GPT-2 despite needing fewer FLOPs. The cause was the inference code, which grew the cache by repeated concatenation and ran normalization and rotary embeddings as many small kernels. For a fair race we converted Treepai to the standard LLaMA format, which required permuting the query and key weights from interleaved to half-split rotary layout. The converted model's logits match the original to within 1.7×10^{-5} , with 100% top-1 agreement. We then ran all three models in the same engine, llama.cpp^[42], at 8-bit precision (Table 12).

Table 12. Generation speed in llama.cpp, 8-bit (q8_0), on consumer Apple-silicon hardware. Characters per second for a 128-token prompt and 128-token reply, by number of concurrent requests.

Model	File size	1 request	8 requests	32 requests
Treepai g0	117 MB	1,976	8,100	19,694
GPT-2 small	135 MB	1,897	6,133	17,168
SmolLM2-135M	143 MB	934	3,259	13,905

On an equal engine the estimate holds: Treepai is the fastest of the three, by 4–32% over GPT-2 and 1.4–2.5× over SmolLM2, whose 30 sequential layers hurt latency beyond what FLOP counts capture. Quantization is free at this scale. Perplexity on the same $40 \times 1,024$ -token chunks is 37.81 in PyTorch fp32, 37.80 in 16-bit GGUF and 37.80 at 8 bits, at half the file size. These measurements are on consumer hardware with small batches; datacenter serving at high concurrency remains to be measured.

9 Recipe ablations

9.1 Single changes at 500M tokens

We tested five changes to the plain recipe, each against the same control, on the first 500M training tokens (Appendix B gives every setting). To separate real effects from run-to-run variation, we trained the control twice with different seeds and took their gap (0.0096 in validation loss) as the noise floor. We decided in advance that a change counts as a win only if it beats the control by more than twice that gap ($\Delta < -0.0192$) and is not slower by more than it saves in tokens. All runs used the full validation set. All arms except C2 share one sampling seed; A5's smaller batches give it a different sequence of samples.

Table 13. Ablations at 500M tokens. "Data to match" is the share of the control's 500M tokens each arm needed to reach the control's final validation loss.

Arm	Val loss	Δ vs control	Data to match	Tokens / s	Verdict
C1 control (plain recipe)	3.7165	–	100%	148,971	control
C2 control, second seed	3.7261	+0.0096	–	148,707	noise floor
A1 warmup-stable-decay to zero [18,20]	3.5981	−0.1184	90%*	148,913	win
A2 Muon optimizer [15,16]	3.5001	−0.2164	66%	146,233	win
A3 QK-norm [43] + zero-init + logit softcap	3.6941	−0.0224	96%	134,349	better loss, slower
A4 grouped-query attention [44] , 4 KV heads	3.7487	+0.0322	not reached	158,279	worse
A5 half batch (262K tokens) [21,22]	3.5413	−0.1752	61%	148,951	win

* Decay-to-zero holds the learning rate high until the final 25% of training, so its loss drops late and "data to match" understates its gain. Its final Δ is the better measure.

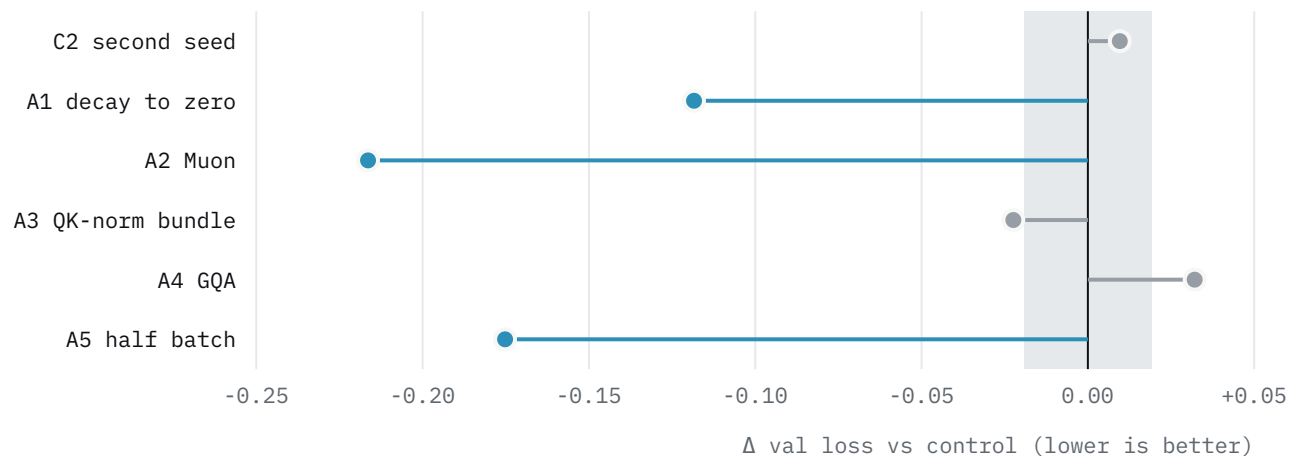


Figure 3. Change in final validation loss against the control, at 500M tokens. Lower is better. The shaded band is the pre-specified no-decision zone (± 0.0192).

Muon gave the largest gain: it needed about two thirds of the data, roughly a $1.5\times$ saving. A careful study at a similar size reports $1.3\text{--}1.4\times$ against a well-tuned AdamW^[17]; our AdamW baseline was not tuned, so part of our measured gain may reflect that. Halving the batch size helped almost as much, which suggests a 524K-token batch is above this model's critical batch size^[21]. The QK-norm bundle improved loss slightly but cost 10% throughput, so it is not cheaper overall. Grouped-query attention made training slightly faster and shrinks the KV cache threefold, but cost quality at this size, partly because it also removes 9% of the parameters. Neither architectural change is part of our current recipe. Each arm was run once, and at one-fifth of the full length.

9.2 The three winners combined, at full length

We then retrained the full 2.47B-token run with the three winning changes combined: Muon, warmup-stable-decay to zero with a 25% cool-down, and 262K-token batches. The success criterion, set in advance, was to reach the baseline's final validation loss with at least 20% fewer tokens and no loss on benchmarks.

Table 14. Full-length runs (2.47B tokens each) with the plain and the improved recipe. Test loss is per token on the sealed test set (same tokenizer, so directly comparable).

Recipe	Test loss	Bits/byte FW-Edu	Bits/byte Wiki	Avg (5)	Avg (9)	GPU-hours
Plain recipe (baseline)	3.278	1.023	1.220	37.3	40.9	4.67
Improved recipe	3.150	0.983	1.174	38.9	42.2	4.81

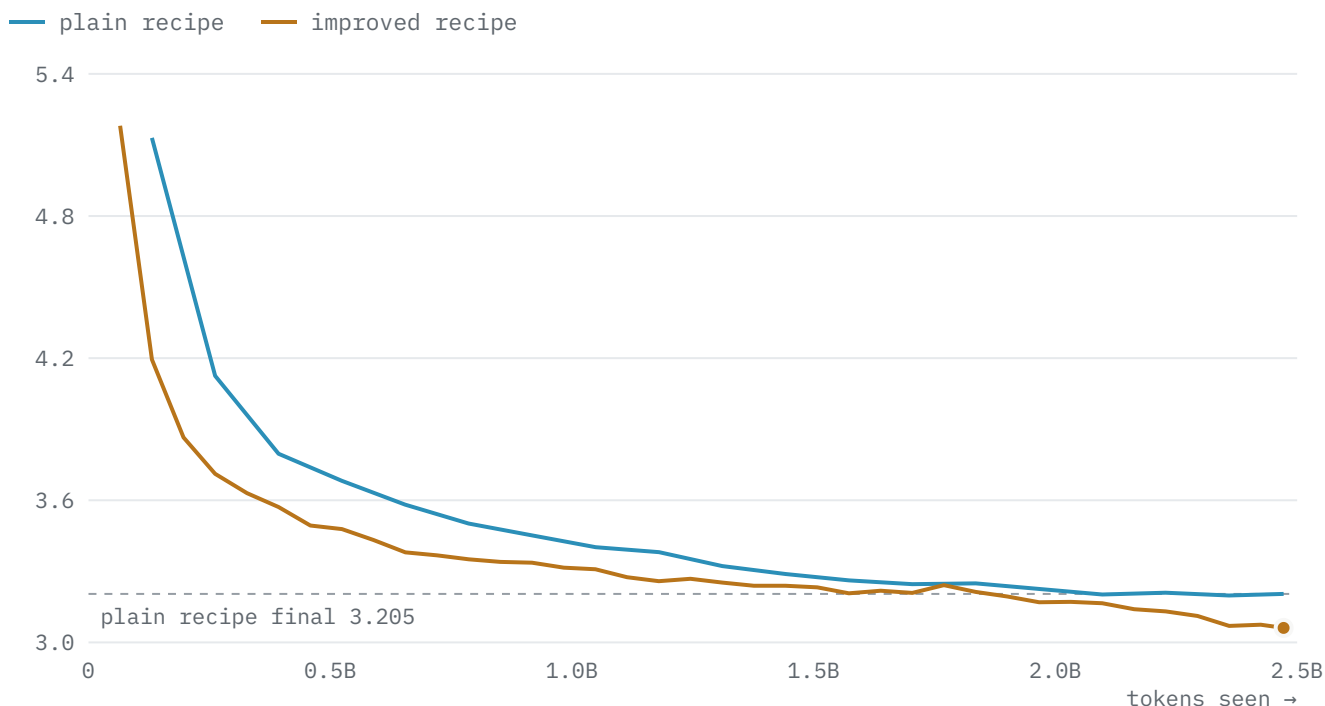


Figure 4. Validation loss of the plain and improved recipes, by tokens seen. The dashed line marks the plain recipe's final value.

The improved recipe's first validation reading at or below the plain recipe's final loss came after 1.90B tokens, 77% of the data (80% on a three-reading average). That is at the pre-specified threshold of at least 20% fewer tokens, and within the noise of single validation readings, so we treat it as roughly 20% rather than a precise figure. The measure understates the recipe: its learning rate only begins to decay at about 1.85B tokens, and most of its advantage arrives during that cool-down (Figure 4). At an equal number of training tokens, the improved recipe ends with a sealed-test loss of 3.150 against 3.278, about thirteen times the noise floor of Section 9.1. The core average rises from 37.3 to 38.9, with clear gains on SciQ (+4.3), ARC-Easy (+3.4) and LAMBADA (+2.3); the changes on the other six benchmarks are within their margins. We adopt the combined recipe as our default.

10 Data mix

Treepai's weakest areas are everyday common sense and narrative text, which FineWeb-Edu covers poorly. To test whether broader web text helps, we trained four 1B-token runs with the improved recipe, changing only the training data: our FineWeb-Edu data twice with different seeds, to measure noise, and two mixes that replace 40% of it with DCLM-Baseline^[9] (broad, quality-filtered web text) or DCLM-Edu (the same corpus, filtered for educational value). Benchmark accuracy at this scale is too coarse to separate data mixes^[10], so we decided in advance to compare the model's per-character log-probability of the correct answer, averaged over each run's last three checkpoints: a mix wins only if it

raises common sense (HellaSwag and PIQA) by more than the gap between the two control runs without lowering science (ARC-Easy and ARC-Challenge) by more than that gap.

Table 15. Data-mix runs (1B tokens each, improved recipe). Held-out loss on three kinds of text (lower is better) and log-probability per character of the correct answer (higher is better), mean of the last three checkpoints.

Training data	Loss FineWeb-Edu	Loss DCLM	Loss FineWeb	Common sense	Science	Verdict
FineWeb-Edu (control)	3.288	3.956	3.822	-0.882	-1.024	control
FineWeb-Edu, second seed	3.284	3.954	3.820	-0.888	-1.027	noise floor
60% FineWeb-Edu + 40% DCLM-Baseline	3.335	3.485	3.706	-0.861	-1.045	no win
60% FineWeb-Edu + 40% DCLM-Edu	3.317	3.570	3.757	-0.875	-1.049	no win

Both mixes made the model much better at predicting general web text and DCLM-Baseline raised common sense by 3.6× the noise floor (the 1.2× with DCLM-Edu is not distinguishable from noise), but both lowered science by 7–9× the noise floor, so neither passes. One explanation is that at 110M parameters broader data competes with educational data for limited capacity. Another is simply that each mix contains 40% less educational text. This test cannot separate the two. HellaSwag accuracy itself barely moved (28.5–28.9%), consistent with accuracy being too coarse at this scale. We keep FineWeb-Edu as the default data and will retest broader mixes at larger model sizes, where more capacity may let them add without trading.

11 Limitations

- Each full-length recipe was trained once, and the two-seed noise floor comes from 500M-token runs. It may understate variation at full length.
- The plain recipe's learning rate was not tuned, so the ablation gains are measured against an untuned baseline and may be smaller against a tuned one.
- Two explanations in this report are hypotheses, not tested results: that the LAMBADA gap comes from the lack of fiction in the training data, and that the strict quality filter helped. Testing each needs further training runs.
- Training samples windows with replacement, so 37% of the training set was never seen. Sampling without replacement would expose more distinct text at the same compute; we have not yet tested it.
- The data-mix test ran once per mix, its noise floor comes from a single pair of seeds, and each mix also removed 40% of the educational text. Its conclusion is suggestive, not established.
- Most benchmarks sit close to chance at this scale. Differences under about 2 points are not meaningful.

- Decontamination cannot check items shorter than 13 words, and four benchmarks were checked only after training.
- Speed was measured on consumer hardware in one engine. Datacenter throughput at high concurrency is not yet measured.
- The model is a base model with no instruction tuning or safety training. It produces confident falsehoods, has not been checked for memorized personal information, and is not suitable for any user-facing use.
- English only, and from a single source. Fiction, conversation, code and mathematics are nearly absent.

12 Conclusion

A small, carefully filtered dataset and a plain recipe produced a 110M model within 1.3 points of GPT-2 small on the core benchmark average after under five hours on one consumer GPU, and far ahead of an early Pythia-160M checkpoint at a similar token count. An improved recipe, tested first at small scale under pre-specified rules, then made the same number of training tokens buy a clearly better model. By our estimates Treepai g0 also needs less compute per character than GPT-2, Pythia or SmolLM2, and in measured tests it generated text faster than GPT-2 and SmolLM2. Its clearest weaknesses, story-level reading and encyclopedic text, track gaps in its training data rather than in its design, and the 37% of training text that was never sampled points to a further, untested gain from sampling without replacement. In a single test, a broader data mix traded science for common sense rather than adding to both at this size. Next we carry the improved recipe, sampling without replacement and a larger budget of educational text to models of 0.5B parameters and above, where we will retest broader data mixes.

Acknowledgments

This work relies on openly released data and tools. We thank the Hugging Face team for FineWeb and FineWeb-Edu, EleutherAI for the Pythia models and the lm-evaluation-harness, OpenAI for GPT-2, the SmolLM team for SmolLM2, the llama.cpp contributors, and the creators of every benchmark used here. FineWeb-Edu is used under the ODC-By 1.0 license.

Appendix A · Model card

Table 16. Model card, following Mitchell et al. [25]

Model	Treepai g0, 110.1M-parameter decoder-only transformer, base (pretrained) model only.
Developer	Treepai, Inc., October 2026.
Availability	Research model. Weights, tokenizer and training code are not released.
Intended use	Research at Treepai on training and serving efficiency; a baseline for later models.
Out of scope	Any user-facing use, factual question answering, advice of any kind, and languages other than English.
Training data	2.47B tokens processed from a 2.47B-token English FineWeb-Edu set (educational score ≥ 3 ; 117 benchmark-overlapping documents removed); 63% of the set seen.
Evaluation	Nine zero-shot capability benchmarks, held-out bits per byte and CrowS-Pairs (Section 7).
Known risks	States false facts fluently; may reproduce stereotypes (Table 10); not checked for memorized personal information.
Compute	4.67 GPU-hours on one RTX 4090; at most ~2.1 kWh at the GPU.

Appendix B · Ablation settings

All ablation runs share the plain recipe of Table 5 except where listed: 500M tokens (954 steps of 524,288 tokens), 100 warmup steps, seed 1337. The full-length improved-recipe run uses 9,430 steps of 262,144 tokens with 500 warmup steps.

Table 17. Settings that differ from the control in each ablation arm.

Arm	Change
C2	Seed 2 (initialization and data sampling)
A1	Learning rate constant after warmup, then linear decay to zero over the last 25% of steps
A2	Muon on all hidden weight matrices, with query, key and value orthogonalized separately; learning rate 0.02, momentum 0.95 (Nesterov), 5 Newton-Schulz steps, no weight decay. AdamW as in Table 5 for embeddings and norms.
A3	RMSNorm on queries and keys, zero-initialized attention-output and MLP down-projections, logits soft-capped at 15
A4	4 key/value heads shared across 12 query heads (100.7M parameters)
A5	262,144-token batches for 1,908 steps (the same 500M tokens in total), 200 warmup steps

Appendix C · All 19 validation samples

Prompt in grey; temperature 0.8, top-k 50; unedited. Loss is validation loss at that step.

0.13B tokens step 250 · val 5.13	The most important thing about the problem is to make a more precise and more complex. The study showed the ability to create a more more more difficult, more important to have access to the new knowledge in the world. In this
0.26B tokens step 500 · val 4.13	The most important thing about why all the same person can have a different purpose in different ways. The other person can be asked not to write the same character as the other person. This is because of the different parts of the
0.39B tokens step 750 · val 3.80	The most important thing about these very strange things is that they are a collection of very small objects and they will, in fact, cover a whole lot of space and time, it is the only thing they can see. Here
0.52B tokens step 1,000 · val 3.68	The most important thing about the topic is getting the word “homeland” that everyone can call their own. If I have a question over one, then I can still get a question over one and just have a topic
0.66B tokens step 1,250 · val 3.58	The most important thing about the water table, is that it helps to draw moisture. While water is a natural element, it can make it more difficult for the water to flow through the plant. When the water is less than
0.79B tokens step 1,500 · val 3.50	The most important thing about the history of the early American Revolution was the fact that the first American statesmen were not immigrants due to their immigration status, but because they were immigrants. It was true that the United States was the
0.92B tokens step 1,750 · val 3.45	The most important thing about this is that it provides new insight into the life of each individual, rather than a story that is entirely unrelated to one’s own life. It is often said that the life of an individual is the
1.05B tokens step 2,000 · val 3.40	The most important thing about the project is that there is an effective way to achieve this goal. By using a variety of techniques and tools, the project is able to develop skills that will help you and your family to be successful
1.18B tokens step 2,250 · val 3.38	The most important thing about the C++ programmer is that he had to understand the core elements of the C++ language. They were, therefore, known as the C++ compiler. He had to study the C++ programming vocabulary

1.31B tokens step 2,500 · val 3.32	The most important thing about people is that they are not immune and are very sick. They have very little immunity. However, with their vaccination they can be very sick to the commoners. So, if not, they should
1.44B tokens step 2,750 · val 3.29	The most important thing about the environment is maintaining the balance of its ecosystems. A healthy balance of the ecosystems on which the ecosystem functions is vital for maintaining the continuity of life on earth. A healthy balance of the ecosystems on which
1.57B tokens step 3,000 · val 3.26	The most important thing about the study of the climate is that it has a pretty good chance of being a fair and realistic assessment of the consequences for human health since the end of the century. This is a real life example of
1.70B tokens step 3,250 · val 3.25	The most important thing about mathematics is that students will be encouraged to study for the material and to explore their own ideas. In part, the way they learn is by doing, but the way they think and understand their world is
1.83B tokens step 3,500 · val 3.25	The most important thing about the study is how much more is known about the natural history of the bird. It is important to understand that certain studies have been done to better understand this bird, including the birds we have studied,
1.97B tokens step 3,750 · val 3.23	The most important thing about this particular type of exercise is that it is a lot of fun – for the body as well as the mind. So don't be afraid to exercise your body!A New Way to Get an
2.10B tokens step 4,000 · val 3.20	The most important thing about it is the fact that it is true. If a person has an active allergy (for example, a cold) then the person isn't aware of the allergy (they just want to go out).
2.23B tokens step 4,250 · val 3.21	The most important thing about the study of ancient people is that they lived a very large part of the world and it was very important to them for the development of our country, and the culture of the modern world. It is
2.36B tokens step 4,500 · val 3.20	The most important thing about your essay is that it explains the topic clearly. Don't just discuss what you're already saying. Just think of what it says. Write a brief summary of your essay. An important part of the

2.47B tokens
step 4,715 · val
3.20

The most important thing about the course is not to be overlooked. The only things that can be learned are the lessons learned and the lessons learned. This course is written in the language of the Bible. It is taught through an

References

- [1] Radford, A., Wu, J., Child, R., et al. Language Models are Unsupervised Multitask Learners. OpenAI technical report, 2019.
- [2] Biderman, S., Schoelkopf, H., Anthony, Q., et al. Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling. ICML 2023.
- [3] Ben Allal, L., Lozhkov, A., Bakouch, E., et al. SmolLM2: When Smol Goes Big – Data-Centric Training of a Fully Open Small Language Model. COLM 2025.
- [4] Gao, L., Biderman, S., Black, S., et al. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. arXiv:2101.00027, 2020.
- [5] Groeneveld, D., Beltagy, I., Walsh, P., et al. OLMo: Accelerating the Science of Language Models. ACL 2024.
- [6] Gunasekar, S., Zhang, Y., Aneja, J., et al. Textbooks Are All You Need. arXiv:2306.11644, 2023.
- [7] Penedo, G., Kydlíček, H., Ben Allal, L., et al. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. NeurIPS 2024 Datasets and Benchmarks Track.
- [8] Lozhkov, A., Ben Allal, L., von Werra, L., Wolf, T. FineWeb-Edu: the Finest Collection of Educational Content. Hugging Face, 2024. doi:10.57967/hf/2497.
- [9] Li, J., Fang, A., Smyrnis, G., et al. DataComp-LM: In Search of the Next Generation of Training Sets for Language Models. NeurIPS 2024 Datasets and Benchmarks Track.
- [10] Magnusson, I., Tai, N., Bogin, B., et al. DataDecide: How to Predict Best Pretraining Data with Small Experiments. ICML 2025.
- [11] Kaplan, J., McCandlish, S., Henighan, T., et al. Scaling Laws for Neural Language Models. arXiv:2001.08361, 2020.
- [12] Hoffmann, J., Borgeaud, S., Mensch, A., et al. Training Compute-Optimal Large Language Models. arXiv:2203.15556, 2022.
- [13] Karpathy, A. nanoGPT. Software, github.com/karpathy/nanoGPT.
- [14] Jordan, K., Bernstein, J., Rappazzo, B., et al. modded-nanogpt: Speedrunning the NanoGPT Baseline. Software, github.com/KellerJordan/modded-nanogpt, 2024.

- [15] Jordan, K., Jin, Y., Boza, V., et al. Muon: An Optimizer for Hidden Layers in Neural Networks. Blog post, 2024. kellerjordan.github.io/posts/muon
- [16] Liu, J., Su, J., Yao, X., et al. Muon is Scalable for LLM Training. arXiv:2502.16982, 2025.
- [17] Wen, K., Hall, D., Ma, T., Liang, P. Fantastic Pretraining Optimizers and Where to Find Them. ICLR 2026.
- [18] Hu, S., Tu, Y., Han, X., et al. MiniCPM: Unveiling the Potential of Small Language Models with Scalable Training Strategies. COLM 2024.
- [19] Hägele, A., Bakouch, E., Kosson, A., et al. Scaling Laws and Compute-Optimal Training Beyond Fixed Training Durations. NeurIPS 2024.
- [20] Bergsma, S., Dey, N., Gosal, G., et al. Straight to Zero: Why Linearly Decaying the Learning Rate to Zero Works Best for LLMs. ICLR 2025.
- [21] McCandlish, S., Kaplan, J., Amodei, D., OpenAI Dota Team. An Empirical Model of Large-Batch Training. arXiv:1812.06162, 2018.
- [22] Marek, M., Lotfi, S., Somasundaram, A., et al. Small Batch Size Training for Language Models: When Vanilla SGD Works, and Why Gradient Accumulation is Wasteful. NeurIPS 2025.
- [23] Gao, L., Tow, J., Abbasi, B., et al. The Language Model Evaluation Harness. Zenodo, 2024. doi:10.5281/zenodo.12608602.
- [24] Brown, T. B., Mann, B., Ryder, N., et al. Language Models are Few-Shot Learners. NeurIPS 2020.
- [25] Mitchell, M., Wu, S., Zaldivar, A., et al. Model Cards for Model Reporting. FAT* 2019.
- [26] Strubell, E., Ganesh, A., McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. ACL 2019.
- [27] Touvron, H., Lavril, T., Izacard, G., et al. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971, 2023.
- [28] Zhang, B., Sennrich, R. Root Mean Square Layer Normalization. NeurIPS 2019.
- [29] Su, J., Lu, Y., Pan, S., et al. RoFormer: Enhanced Transformer with Rotary Position Embedding. Neurocomputing 568:127063, 2024.
- [30] Shazeer, N. GLU Variants Improve Transformer. arXiv:2002.05202, 2020.
- [31] Karpathy, A. Reproducing GPT-2 (1.6B) in llm.c. GitHub discussion, github.com/karpathy/llm.c/discussions/677, 2024.
- [32] Zellers, R., Holtzman, A., Bisk, Y., et al. HellaSwag: Can a Machine Really Finish Your Sentence? ACL 2019.
- [33] Clark, P., Cowhey, I., Etzioni, O., et al. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. arXiv:1803.05457, 2018.

- [34] Bisk, Y., Zellers, R., Le Bras, R., et al. PIQA: Reasoning about Physical Commonsense in Natural Language. AAAI 2020.
- [35] Paperno, D., Kruszewski, G., Lazaridou, A., et al. The LAMBADA Dataset: Word Prediction Requiring a Broad Discourse Context. ACL 2016.
- [36] Sakaguchi, K., Le Bras, R., Bhagavatula, C., Choi, Y. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. AAAI 2020.
- [37] Mihaylov, T., Clark, P., Khot, T., Sabharwal, A. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. EMNLP 2018.
- [38] Welbl, J., Liu, N. F., Gardner, M. Crowdsourcing Multiple Choice Science Questions. Workshop on Noisy User-generated Text (W-NUT), 2017.
- [39] Hendrycks, D., Burns, C., Basart, S., et al. Measuring Massive Multitask Language Understanding. ICLR 2021.
- [40] Nangia, N., Vania, C., Bhalerao, R., Bowman, S. R. CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models. EMNLP 2020.
- [41] Blodgett, S. L., Lopez, G., Olteanu, A., et al. Stereotyping Norwegian Salmon: An Inventory of Pitfalls in Fairness Benchmark Datasets. ACL 2021.
- [42] Gerganov, G., et al. llama.cpp. Software, github.com/ggml-org/llama.cpp.
- [43] Dehghani, M., Djolonga, J., Mustafa, B., et al. Scaling Vision Transformers to 22 Billion Parameters. ICML 2023.
- [44] Ainslie, J., Lee-Thorp, J., de Jong, M., et al. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. EMNLP 2023.