Training smol and BIG LLM

Elie Bakouch, Researcher @ Hugging Face

GOSIM PARIS 2025

@eliebak on Hugging Face, Github, Linkedin | @eliebakouch on X

You might know Hugging Face for

transformers Public

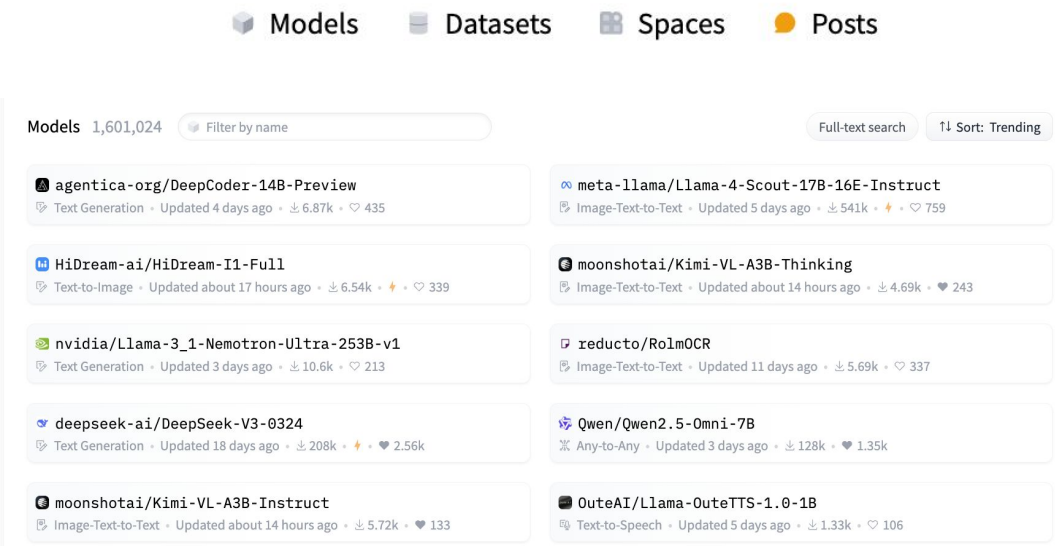
```python
# transformers.py
from transformers import pipeline
import torch

model_id = "meta-llama/Llama-4-Scout-17B-16E"

pipe = pipeline(
    "text-generation",
    model=model_id,
    device_map="auto",
    torch_dtype=torch.bfloat16,
)

output = pipe("Roses are red,", max_new_tokens=200)
```

-> Train/Use any model
-> Reference implementation



-> Share model / dataset / app / papers
-> Interact with the community with post / blog
-> (NEW) Efficient/Fast access storage with Xet and lightning inference speed with Inference Provider

We love open science/research

Let’s dive into SmolLM

Scaling laws (chinchilla)

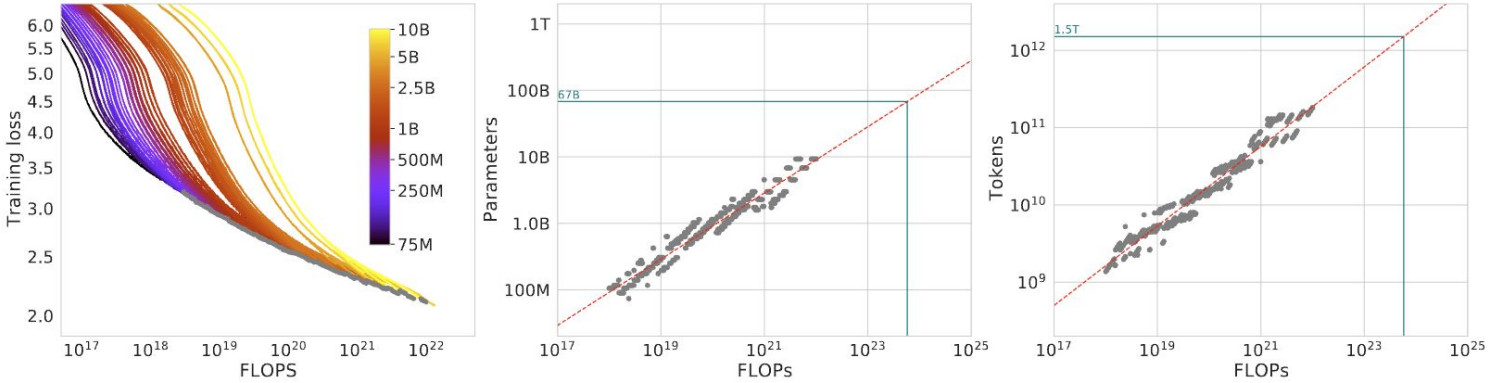


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* (5.76×10^{23}).

https://arxiv.org/abs/2203.15556

What chinchilla is telling us

If you want the best model given a compute budget - ***and that's the only thing you care about*** - you should follow me!

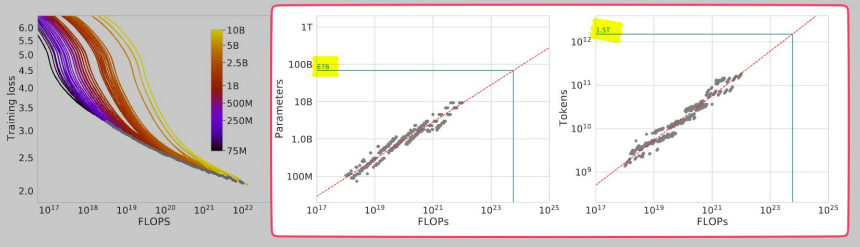


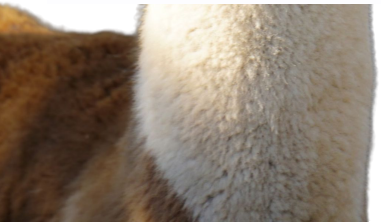
Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* (5.76×10^{23}).

So if you follow chinchilla 67B model, you should train it on 1.5T tokens!

In reality…

I'm train on 40T token and i'm only 17B active

meta-llama/Llama-4-Scout-17B-16E-Instruct like 868

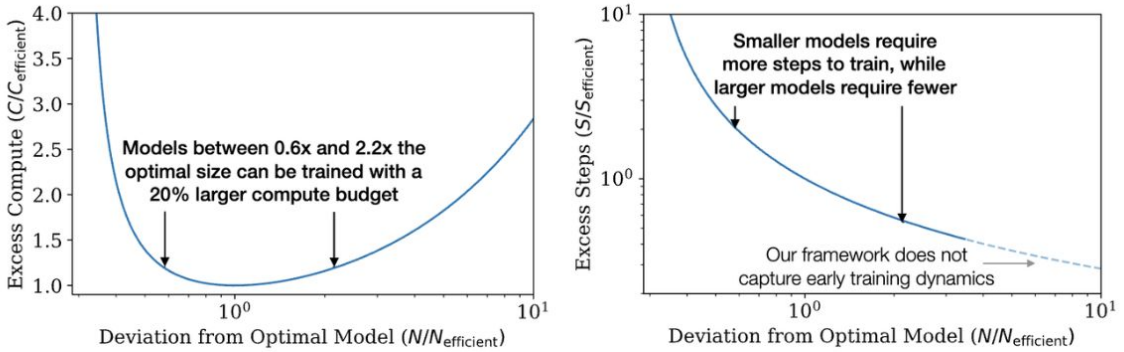


I'm train on 36T token for only 0.6B ????

Qwen/Qwen3-0.6B like 200

Why?

1. Model are still learning after chinchilla optimal point
2. Big model are slow at inference => you care about usability!

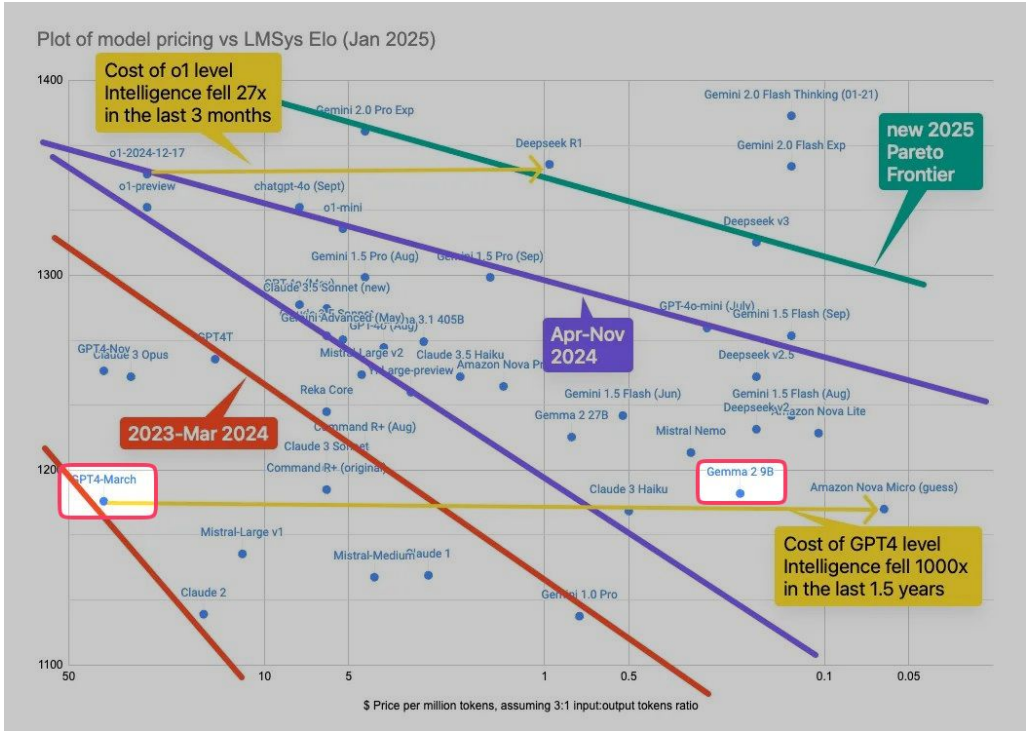


From the original scalling laws paper!

Figure 12 **Left:** Given a fixed compute budget, a particular model size is optimal, though somewhat larger or smaller models can be trained with minimal additional compute. **Right:** Models larger than the compute-efficient size require fewer steps to train, allowing for potentially faster training if sufficient additional parallelism is possible. Note that this equation should not be trusted for very large models, as it is only valid in the power-law region of the learning curve, after initial transient effects.

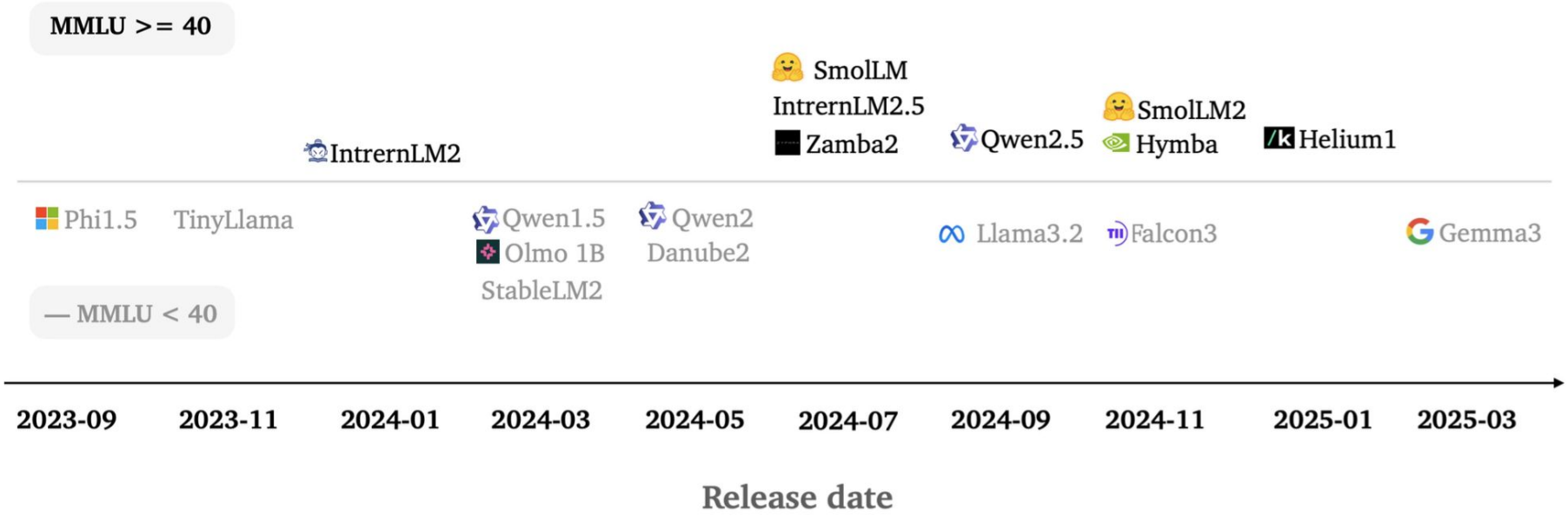
https://arxiv.org/abs/2001.08361

Intelligence per bytes is increasing

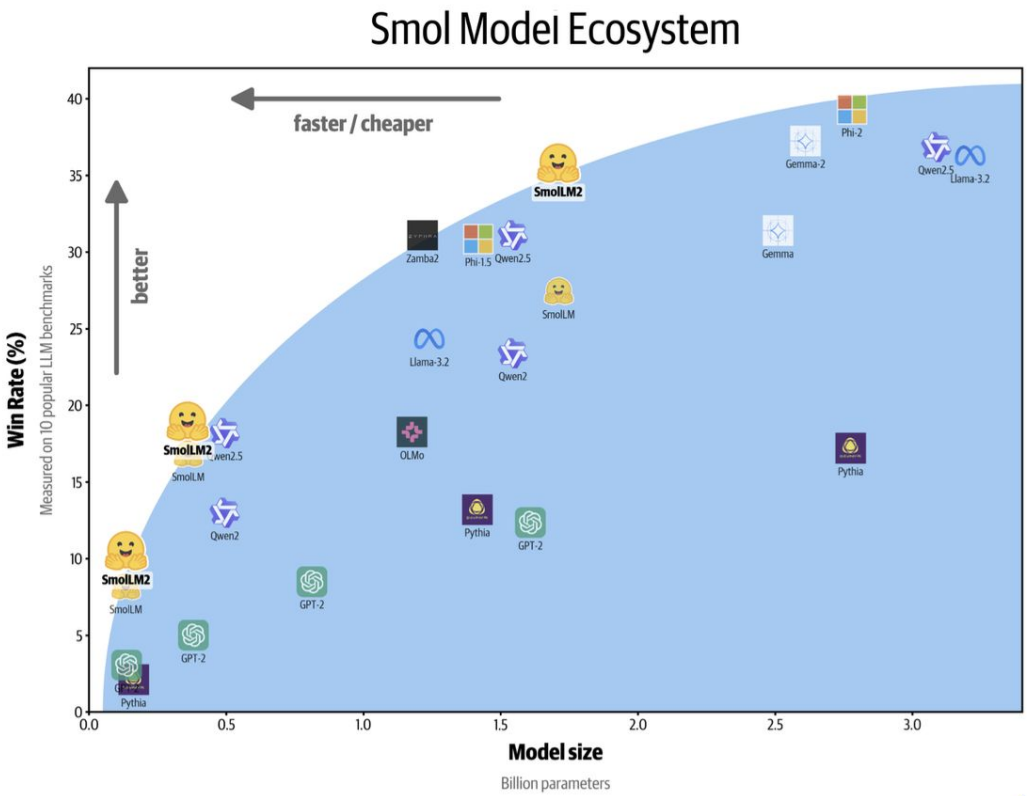


https://x.com/swyx/status/1882933368444309723

Same trend at <3B scale

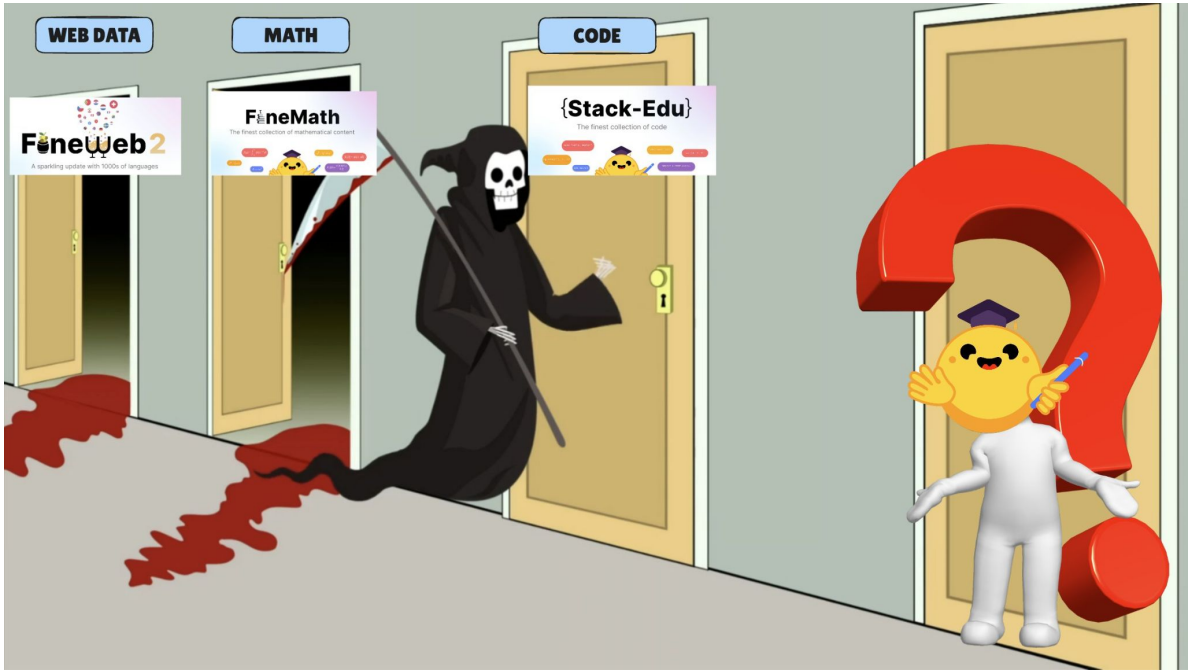


SmolLM2



https://huggingface.co/papers/2502.02737

SmolLM2 secret? Quality data!



HuggingFaceFW/fineweb-2
Viewer • Updated Jan 8 • 12.5B • 51k • ♥ 474

HuggingFaceFW/fineweb-edu
Viewer • Updated Jan 31 • 3.3B • 161k • ♥ 671

HuggingFaceTB/cosmopedia
Viewer • Updated Aug 13, 2024 • 31.1M • 28.2k • ♥ 611

HuggingFaceTB/smoltalk
Viewer • Updated Feb 10 • 2.2M • 7.3k • ♥ 333

HuggingFaceTB/finemath
Viewer • Updated Feb 6 • 48.3M • 16.6k • ♥ 308

HuggingFaceTB/stack-edu
Viewer • Updated Mar 20 • 167M • 1.96k • ♥ 33

HuggingFaceTB/dclm-edu
Viewer • Updated Mar 7 • 1B • 14.6k • ♥ 26

https://huggingface.co/papers/2502.02737

Quality data AND training for very long

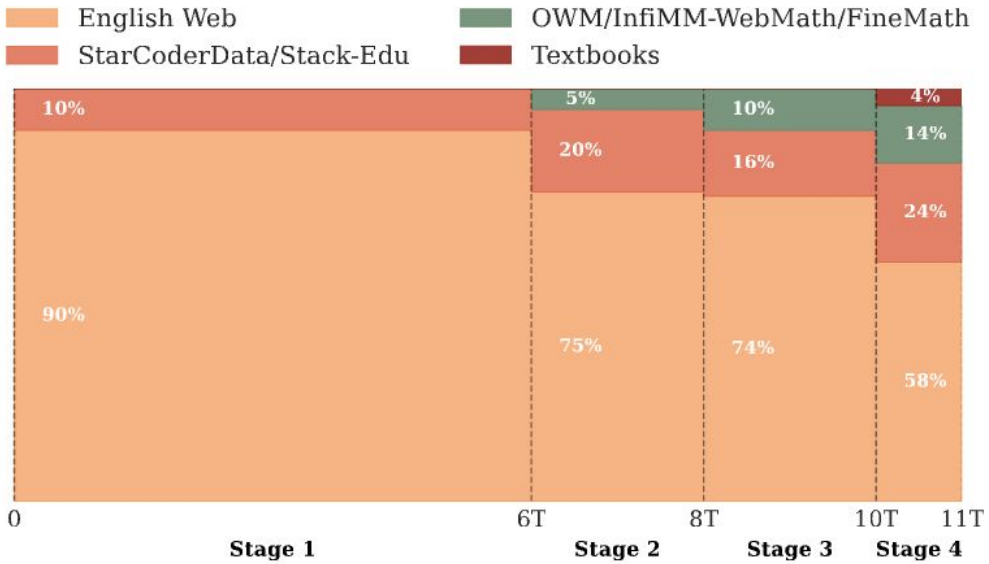
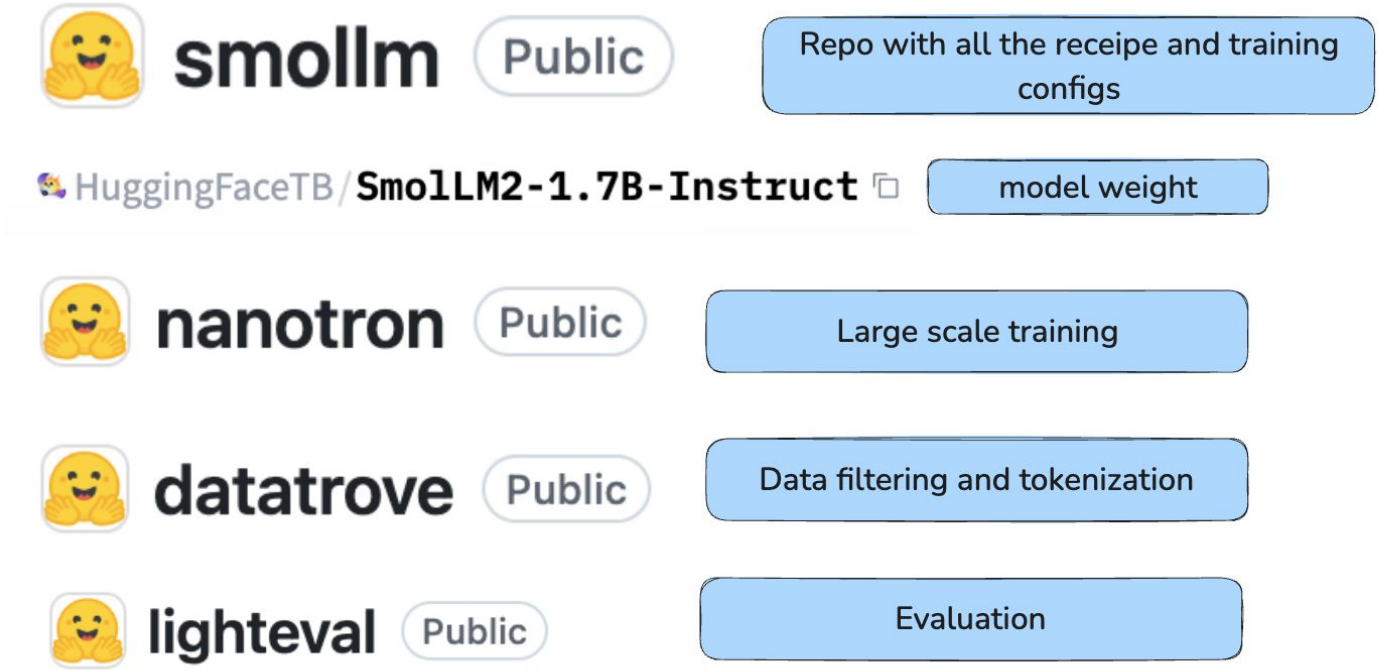


Figure 2. Dataset mixtures across training stages. Detailed descriptions are provided in Section 4. The x-axis represents the number of training tokens.

https://huggingface.co/papers/2502.02737

Everything is OPEN



https://huggingface.co/papers/2502.02737

Smol SmolLM

Base Pre-Trained Model

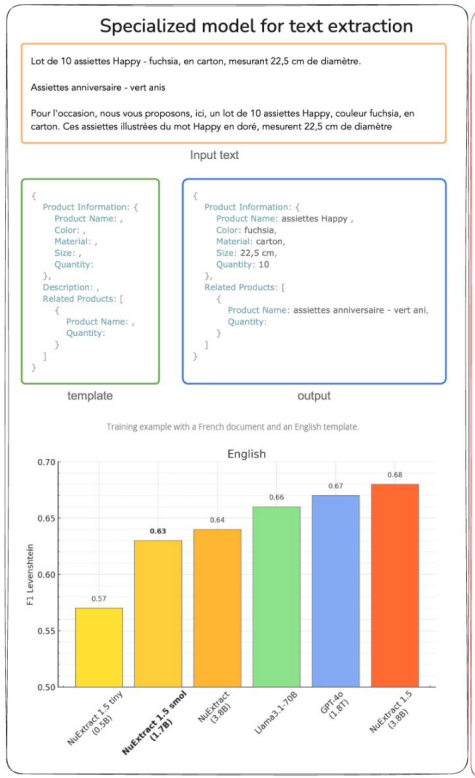
Metrics	SmolLM2-360M	Qwen2.5-0.5B	SmolLM-360M
HellaSwag	**54.5**	51.2	51.8
ARC (Average)	**53.0**	45.4	50.1
PIQA	**71.7**	69.9	71.6
MMLU (cloze)	**35.8**	33.7	34.4
CommonsenseQA	**38.0**	31.6	35.3
TriviaQA	**16.9**	4.3	9.1
Winogrande	52.5	**54.1**	52.8
OpenBookQA	**37.4**	**37.4**	37.2
GSM8K (5-shot)	3.2	**33.4**	1.6

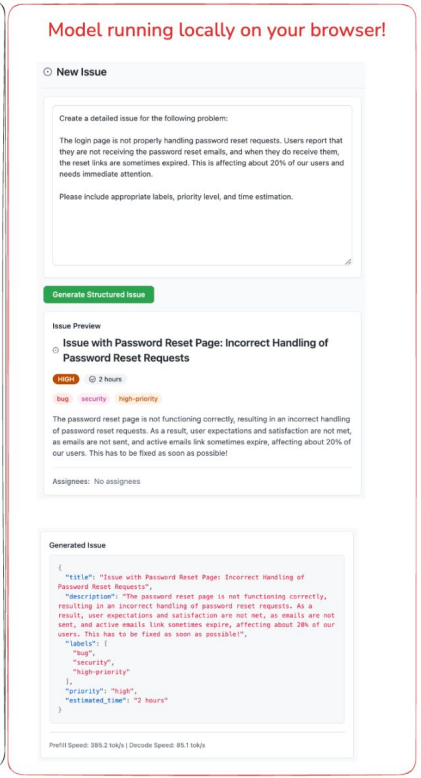
Base pre-trained model

Metrics	SmolLM2-135M-8k	SmolLM-135M
HellaSwag	**42.1**	41.2
ARC (Average)	**43.9**	42.4
PIQA	68.4	68.4
MMLU (cloze)	**31.5**	30.2
CommonsenseQA	**33.9**	32.7
TriviaQA	4.1	**4.3**
Winogrande	51.3	51.3
OpenBookQA	**34.6**	34.0
GSM8K (5-shot)	**1.4**	1.0

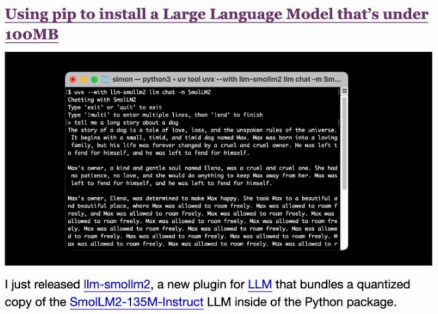
https://huggingface.co/papers/2502.02737

A few applications

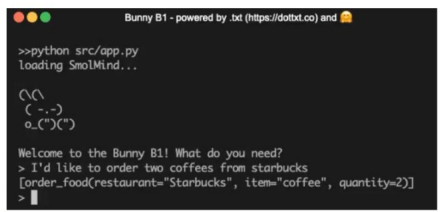




Use an llm with pip install or inside a pdf

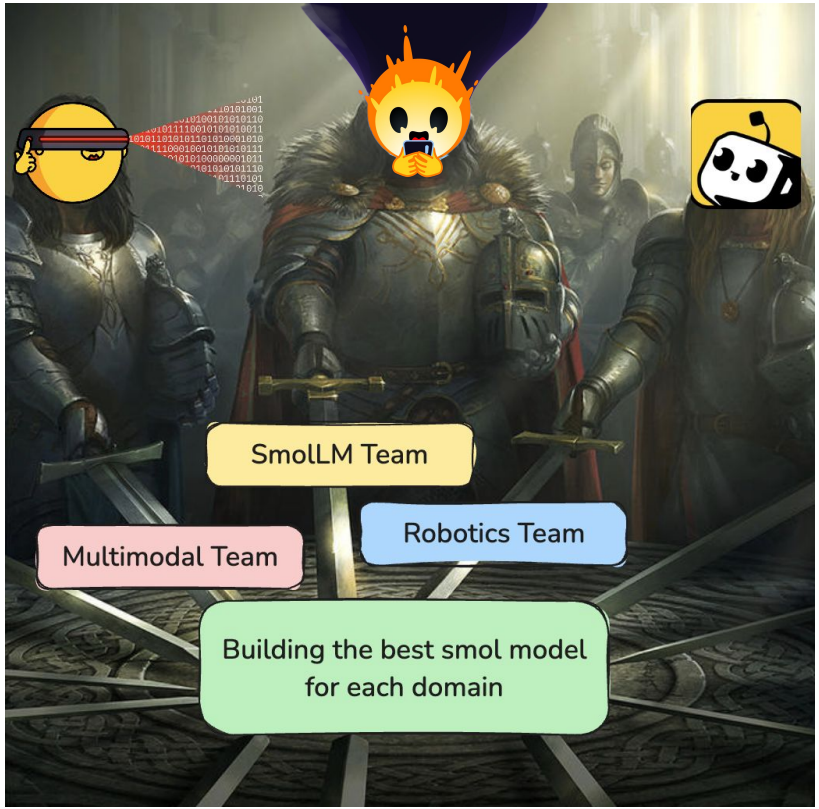


Simple (or not) function calling



https://huggingface.co/spaces/reach-vb/github-issue-generator-webgpu

Collaboration at Hugging Face



SmolVLM

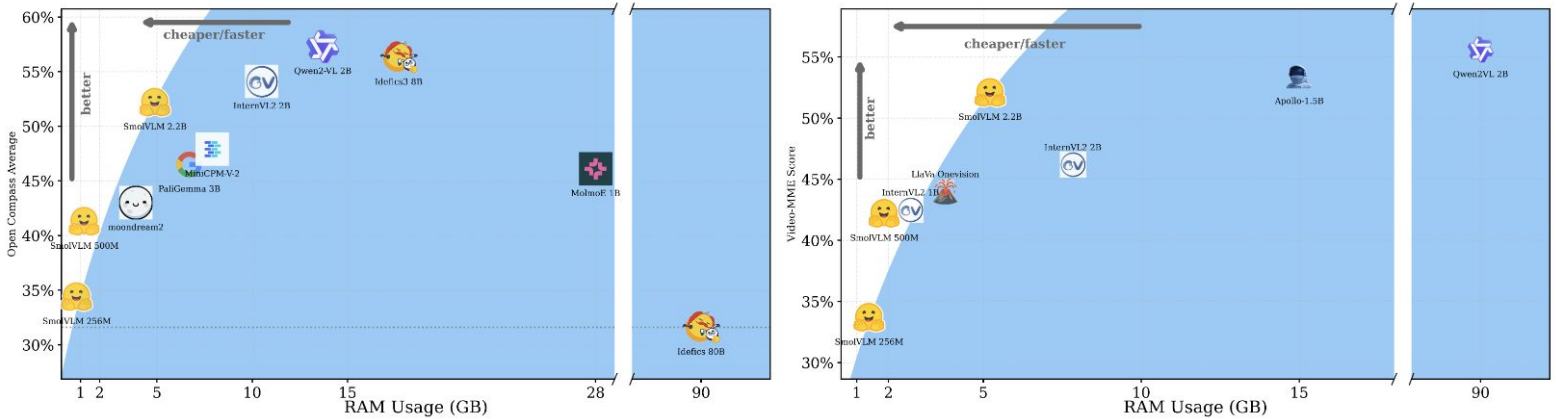


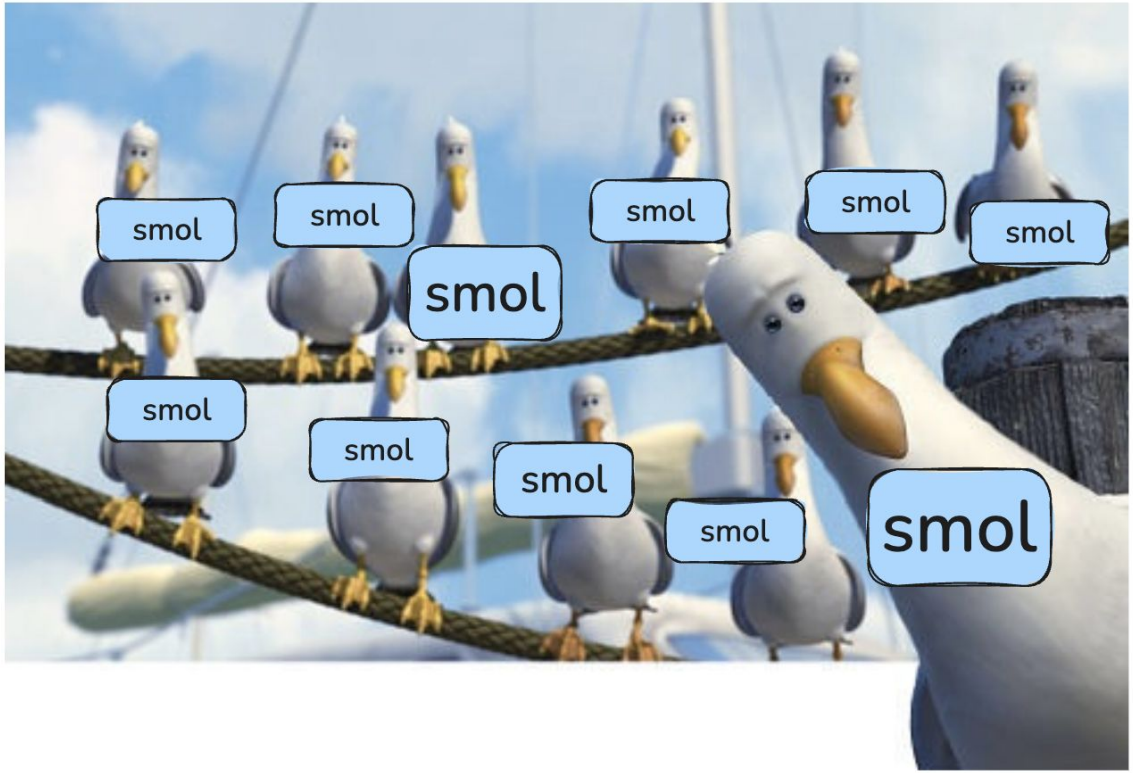
Figure 1 | Smol yet Mighty: comparison of SmolVLM with other state-of-the-art small VLM models. Image results are sourced from the OpenCompass OpenVLM leaderboard (Duan et al., 2024).

https://huggingface.co/papers/2504.05299

SmolVLA

- Finetune SmolVLM into a VLA for robotics
- In progress!

Smol smol smol smol



What about big models?

How to alleviate the compute bottleneck

DiLoCo

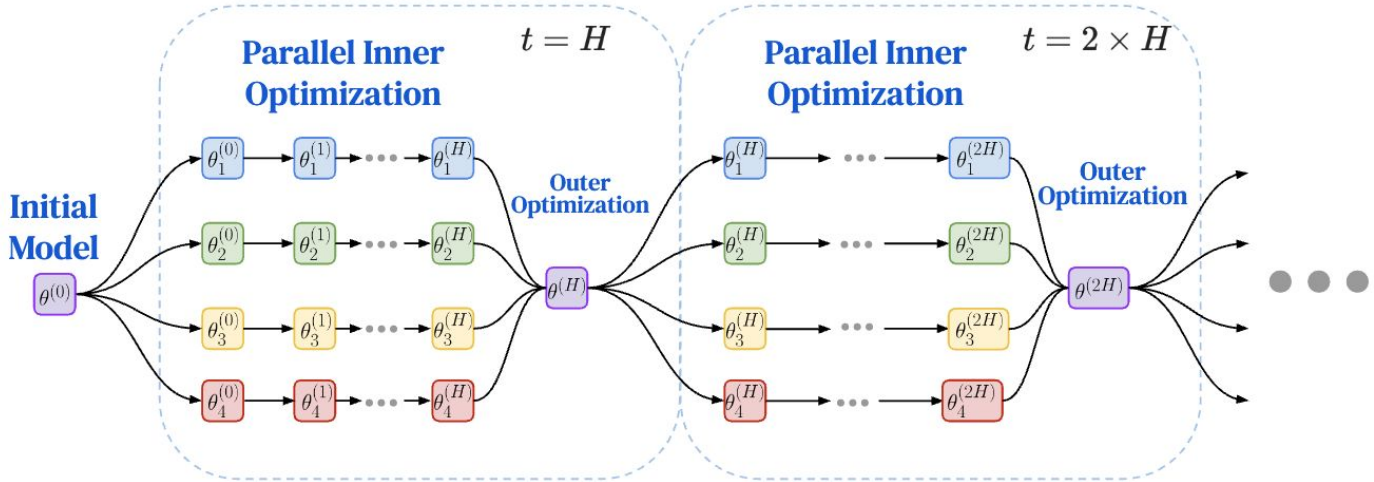


Figure 1: **DiLoCo**. Each DiLoCo model replica trains independently for H inner optimization steps. These models are synchronized via an outer optimization step, usually involving momentum across outer optimization steps. In this figure, there are $M = 4$ replicas.

https://arxiv.org/abs/2503.09799v1

DiLoCo

Algorithm 1 DiLoCo Algorithm

Require: Initial model $\theta^{(0)}$
Require: k workers
Require: Data shards $\{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$
Require: Optimizers `InnerOpt` and `OuterOpt`

for `outer step` $t = 1 \ldots T$ **do**
  for `worker` $i = 1 \ldots k$ **do**
    $\theta_i^{(t)} \leftarrow \theta^{(t-1)}$
    for `inner step` $h = 1 \ldots H$ **do**
      $x \sim \mathcal{D}_i$
      $\mathcal{L} \leftarrow f(x, \theta_i^{(t)})$
      ▷ Inner optimization:
      $\theta_i^{(t)} \leftarrow \texttt{InnerOpt}(\theta_i^{(t)}, \nabla_{\mathcal{L}})$
    end for
  end for
  ▷ Averaging outer gradients:
  $\Delta^{(t)} \leftarrow \frac{1}{k} \sum_{i=1}^{k} (\theta^{(t-1)} - \theta_i^{(t)})$
  ▷ Outer optimization:
  $\theta^{(t)} \leftarrow \texttt{OuterOpt}(\theta^{(t-1)}, \Delta^{(t)})$
end for

Inner optimizer: AdamW
Outer optimizer: Nesterov SGD :o

https://arxiv.org/abs/2311.08105

Intellect-1

INTELLECT-1 Technical Report

Sami Jaghouar Prime Intellect
Jack Min Ong Prime Intellect
Manveer Basra Prime Intellect
Fares Obeid Prime Intellect

Jannik Straube Prime Intellect
Michael Keiblinger Prime Intellect
Elie Bakouch Hugging Face
Lucas Atkins Arcee AI

Maziyar Panahi Arcee AI
Charles Goddard Arcee AI
Max Ryabinin Together AI
Johannes Hagemann Prime Intellect
johannes@primeintellect.ai

Abstract

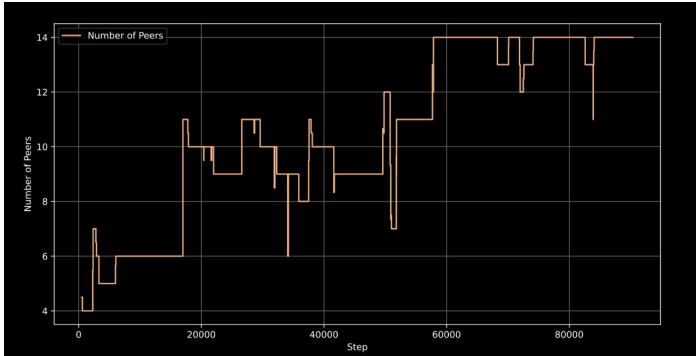
In this report, we introduce INTELLECT-1, the first 10 billion parameter language model collaboratively trained across the globe, demonstrating that large-scale model training is no longer confined to large corporations but can be achieved through a distributed, community-driven approach.

INTELLECT-1 was trained on 1 trillion tokens using up to 14 concurrent nodes distributed across 3 continents, with contributions from 30 independent compute providers dynamically joining and leaving the training process, while maintaining 83-96% compute utilization and 36.2–41.4% model FLOPS utilization.

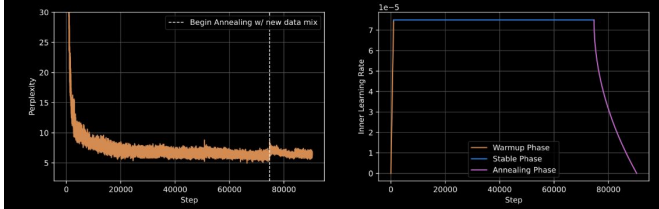
We leverage PRIME, our scalable distributed training framework designed for fault-tolerant, high-performance training on unreliable, globally distributed nodes. Key innovations in PRIME include the `ElasticDeviceMesh`, which manages dynamic global process groups for fault-tolerant communication across the internet and local process groups for communication within a node, live checkpoint recovery, kernels, and a hybrid DiLoCo-FSDP2 implementation.

Using PRIME with DiLoCo and our custom int8 all-reduce, we achieve a 400× reduction in communication bandwidth compared to traditional data-parallel training settings while delivering comparable performance.

These results demonstrate the feasibility and promise of training frontier foundation models in a decentralized network of global GPU resources.



Number of active training nodes over training steps. The graph demonstrates PRIME's ability to handle dynamic node participation, starting with 4 nodes and scaling up to 14 nodes, while maintaining training stability despite frequent node fluctuations.



Training dynamics showing model perplexity and learning rate over training steps, including warmup, stable, and annealing phases.

https://arxiv.org/abs/2412.01152v1

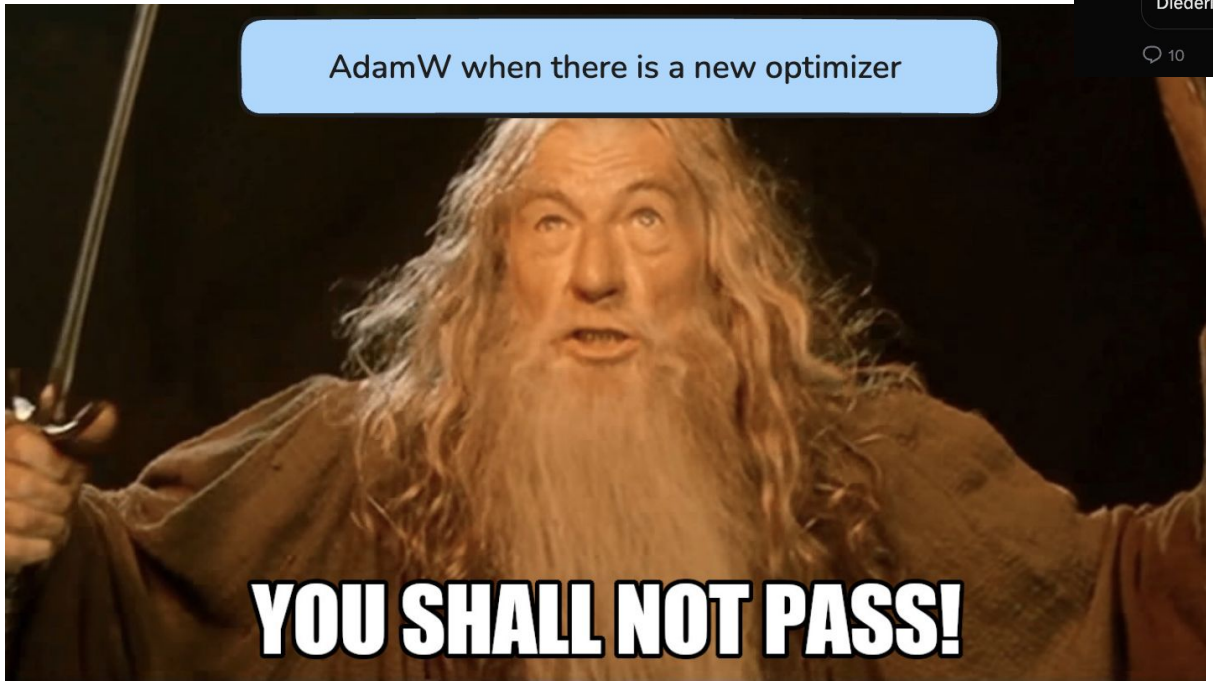
The BOOM

Goal: extend SmolLM recipe to larger scale (70B-100B on 20-30T tokens?)

Starting a pilot *very soon* in partnership with big cluster and company like Prime Intellect

Let’s now see the day to day of a researcher working in pre-training

10 years of using Adam..



What does big labs do ?

They use Adam
- Llama1/2/3/4
- Olmo
- Qwen
- Deepseek
- …
- ~Apple

Others ?
- Google
- xAI?
- OpenAI??
- Anthorpic???

Skin care routine of our optimizer

“New” paradigm: Pre-Conditioner

pre-conidionner.py

```
# sgd
W_t = W_t-1 - lr * G_t
# preconditionner
P = ????
W_t = W_t-1 - lr * P^-1 @ G_t
```

Muon (Keller and al.)

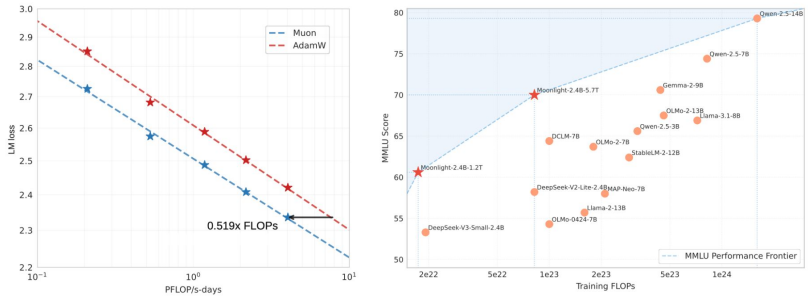
Algorithm 2 Muon

Require: Learning rate η, momentum μ

1: Initialize $B_0 \leftarrow 0$
2: **for** $t = 1, \ldots$ **do**
3: Compute gradient $G_t \leftarrow \nabla_\theta \mathcal{L}_t(\theta_{t-1})$
4: $B_t \leftarrow \mu B_{t-1} + G_t$
5: $O_t \leftarrow \text{NewtonSchulz5}(B_t)$
6: Update parameters $\theta_t \leftarrow \theta_{t-1} - \eta O_t$
7: **end for**
8: **return** θ_t

Open source lab are our savior: Kimi

better than adam!



(a) (b)

Figure 1: Scaling up with Muon. **(a)** Scaling law experiments comparing Muon and Adam. Muon is $\sim 2\times$ more computational efficient than Adam with compute optimal training. **(b)** The MMLU performance of our Moonlight model optimized with Muon and other comparable models. Moonlight advances the Pareto frontier of performance vs training FLOPs.

Table 2: Scaling Law Models and Hyper-Parameters

# Params. w/o Embedding	Head	Layer	Hidden	Tokens	LR	Batch Size*
399M	12	12	1536	8.92B	9.503e-4	96
545M	14	14	1792	14.04B	9.143e-4	128
822M	16	16	2048	20.76B	8.825e-4	160
1.1B	18	18	2304	28.54B	8.561e-4	192
1.5B	20	20	2560	38.91B	8.305e-4	256

*In terms of number of examples in 8K context length.

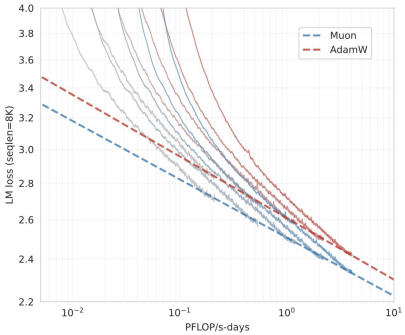


Figure 3: Fitted scaling law curves for Muon and AdamW optimizers.

https://x.com/Kimi_Moonshot/status/1893379158472044623

Open source lab are our savior: Essential AI

And allow us to scale the BS and work with muP

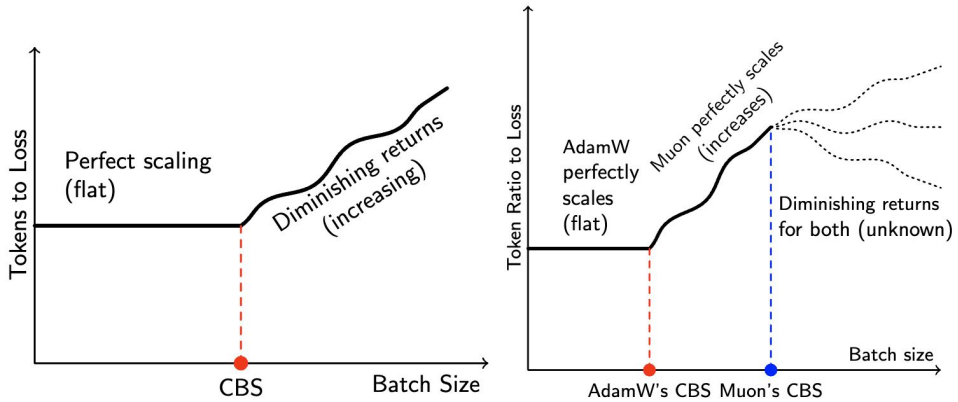
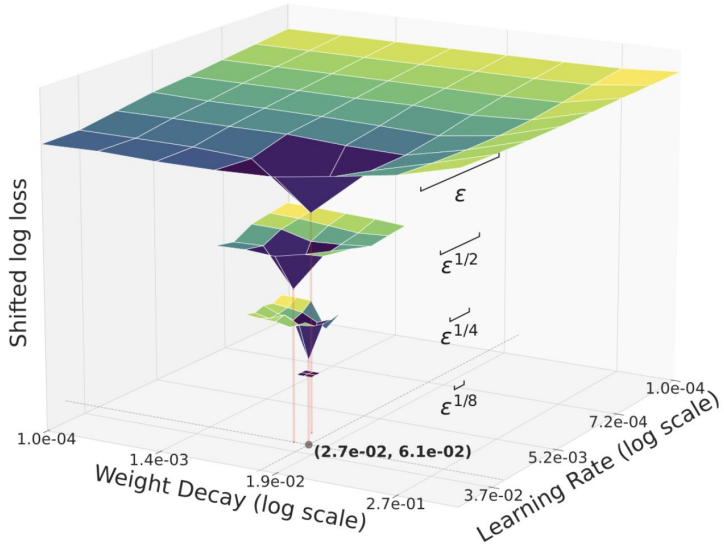


Figure 4: *Token ratios to loss* give a clearer picture of the practical advantages of Muon over AdamW, compared to *tokens to loss*



https://arxiv.org/abs/2505.02222

AdEMAMix

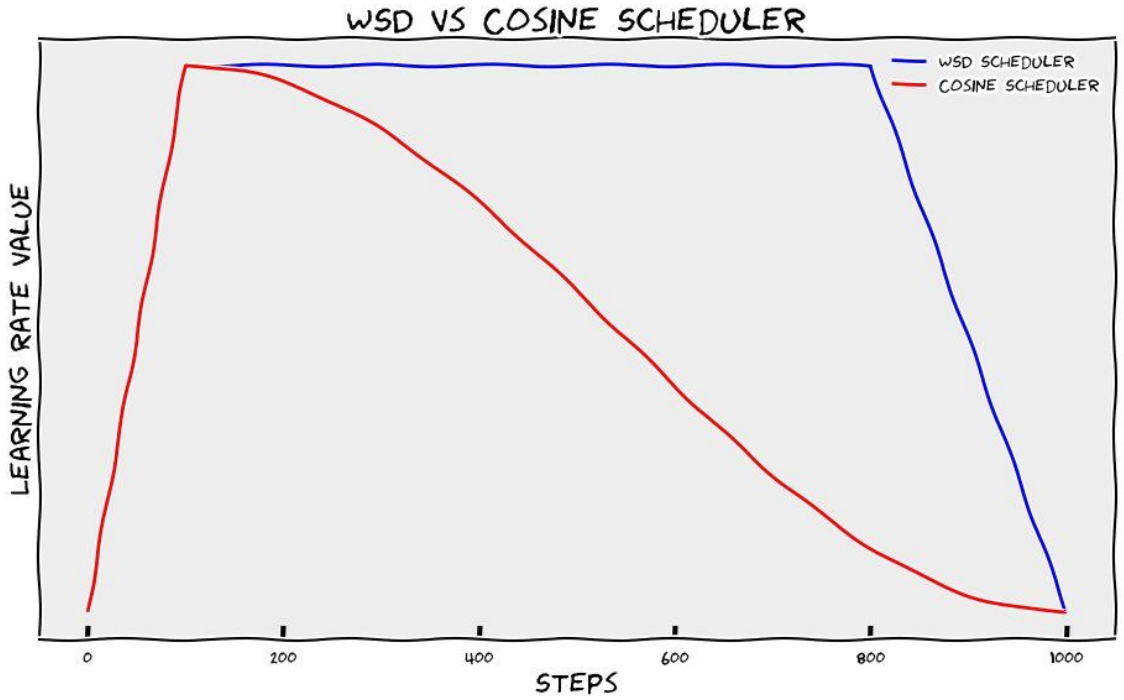
AdEMAMix. To keep a high sensitivity to recent gradients, while also incorporating information from older gradients, we add a second EMA (changes compared to AdamW are in Blue):

$$\begin{cases} \boldsymbol{m}_1^{(t)} = \beta_1 \boldsymbol{m}_1^{(t-1)} + (1-\beta_1)\boldsymbol{g}^{(t)}, & \hat{\boldsymbol{m}}_1^{(t)} = \frac{\boldsymbol{m}_1^{(t)}}{1-\beta_1^t} \\ \boldsymbol{m}_2^{(t)} = \beta_3 \boldsymbol{m}_2^{(t-1)} + (1-\beta_3)\boldsymbol{g}^{(t)} & \\ \boldsymbol{\nu}^{(t)} = \beta_2 \boldsymbol{\nu}^{(t-1)} + (1-\beta_2)\boldsymbol{g}^{(t)^2}, & \hat{\boldsymbol{\nu}}^{(t)} = \frac{\boldsymbol{\nu}^{(t)}}{1-\beta_2^t} \\ \boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta\left(\frac{\hat{\boldsymbol{m}}_1^{(t)} + \alpha \boldsymbol{m}_2^{(t)}}{\sqrt{\hat{\boldsymbol{\nu}}^{(t)}}+\epsilon} + \lambda \boldsymbol{\theta}^{(t-1)}\right). & \end{cases} \quad \text{(AdEMAMix)}$$

In our experiments, while the values of β_1, β_2 remain similar to those of equation AdamW, we often use $\beta_3 = 0.9999$. We find $\alpha \in [4, 10]$ to work well in practice.

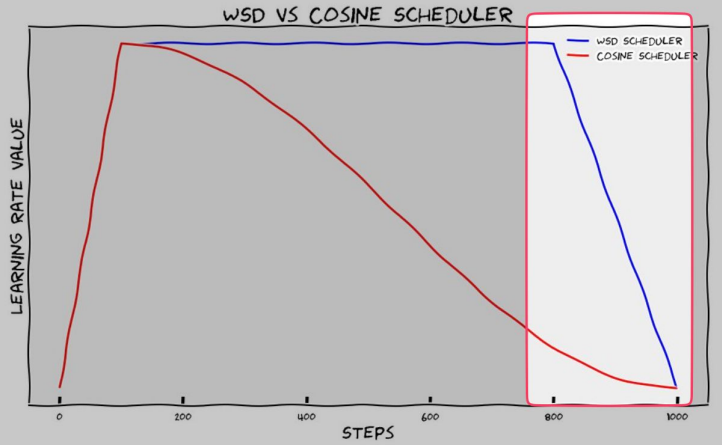
https://arxiv.org/abs/2409.03137

Learning rate schedule



https://x.com/eliebakouch/status/1790769969719828668

Let’s be careful…



AdEMAMix. To keep a high sensitivity to recent gradients, while also incorporating information from older gradients, we add a second EMA (changes compared to AdamW are in Blue):

$$\begin{cases} \boldsymbol{m}_1^{(t)} = \beta_1 \boldsymbol{m}_1^{(t-1)} + (1-\beta_1)\boldsymbol{g}^{(t)}, \qquad \hat{\boldsymbol{m}}_1^{(t)} = \frac{\boldsymbol{m}_1^{(t)}}{1-\beta_1^t} \\ \boldsymbol{m}_2^{(t)} = \beta_3 \boldsymbol{m}_2^{(t-1)} + (1-\beta_3)\boldsymbol{g}^{(t)} \\ \boldsymbol{\nu}^{(t)} = \beta_2 \boldsymbol{\nu}^{(t-1)} + (1-\beta_2)\boldsymbol{g}^{(t)2}, \qquad \hat{\boldsymbol{\nu}}^{(t)} = \frac{\boldsymbol{\nu}^{(t)}}{1-\beta_2^t} \\ \boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta\left(\frac{\hat{\boldsymbol{m}}_1^{(t)} + \alpha \boldsymbol{m}_2^{(t)}}{\sqrt{\hat{\boldsymbol{\nu}}^{(t)}} + \epsilon} + \lambda \boldsymbol{\theta}^{(t-1)}\right). \end{cases} \quad \text{(AdEMAMix)}$$

In our experiments, while the values of β_1, β_2 remain similar to those of equation AdamW, we often use $\beta_3 = 0.9999$. We find $\alpha \in [4, 10]$ to work well in practice.

A common issue..

Some ideas: Muon with more momentum?

Algorithm 2 ADEMAMUON — sorry for this awful naming

Require: Learning rate η, momentum μ

1: Initialize $B_0 \leftarrow 0$
2: **for** $t = 1, \ldots$ **do**
3: Compute gradient $G_t \leftarrow \nabla_\theta \mathcal{L}_t(\theta_{t-1})$
4: ~~$B_t \leftarrow \mu B_{t-1} + G_t$~~

$\boldsymbol{m}_1^{(t)} = \beta_1 \boldsymbol{m}_1^{(t-1)} + (1-\beta_1)\boldsymbol{g}^{(t)},$ — from ademamix?
$\boldsymbol{m}_2^{(t)} = \beta_3 \boldsymbol{m}_2^{(t-1)} + (1-\beta_3)\boldsymbol{g}^{(t)}$
$\boldsymbol{\nu}^{(t)} = \beta_2 \boldsymbol{\nu}^{(t-1)} + (1-\beta_2)\boldsymbol{g}^{(t)2},$

5: $O_t \leftarrow \text{NewtonSchulz5}(B_t)$
6: Update parameters $\theta_t \leftarrow \theta_{t-1} - \eta O_t$
7: **end for**
8: **return** θ_t

Some clarification: LR Scaling

Kimi Moonlight Scaling

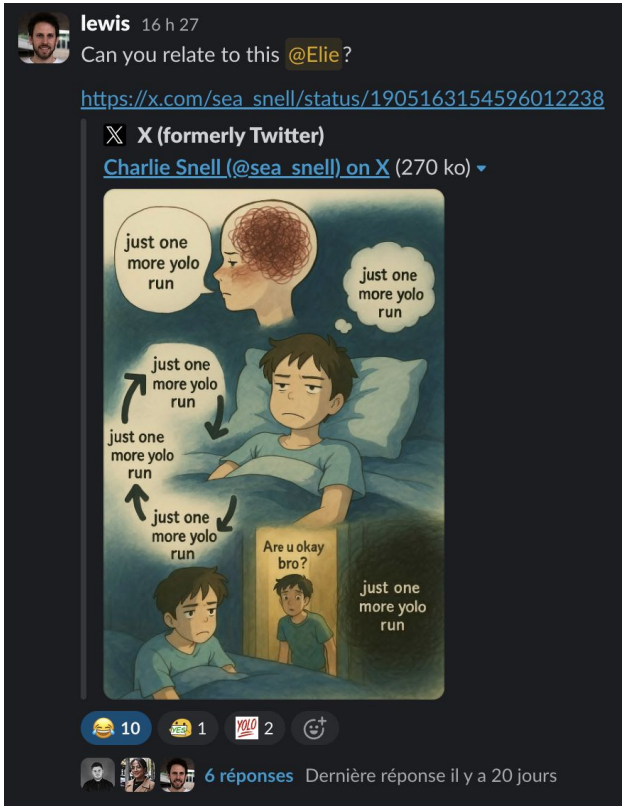
3. `Adjusted LR`. For each update matrix, we can scale its learning rate by a factor of $0.2 \cdot \sqrt{\max(A, B)}$ based on its shape.

$$\mathbf{W}_t = \mathbf{W}_{t-1} - \eta_t(0.2 \cdot \mathbf{O}_t \cdot \sqrt{\max(A, B)} + \lambda \mathbf{W}_{t-1}) \tag{7}$$

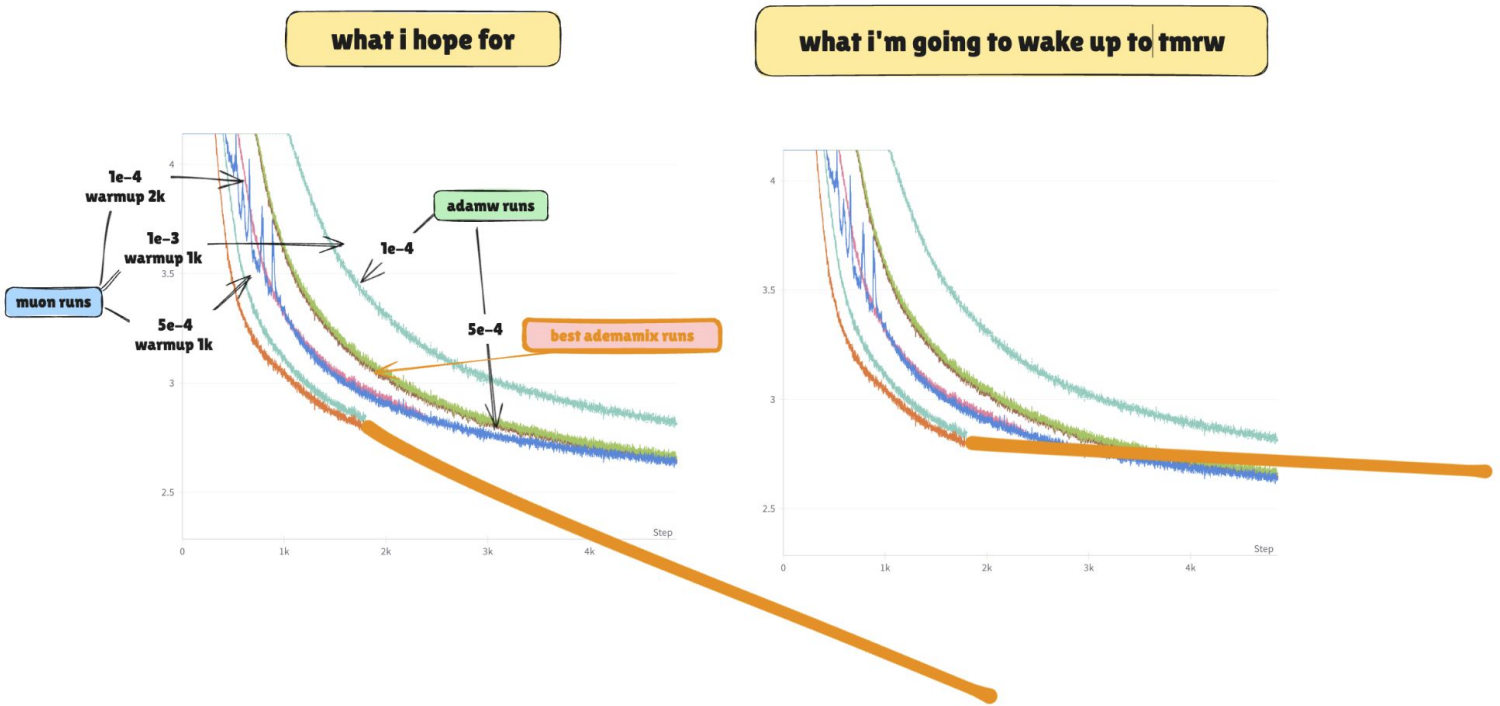
Spectral MuP Scaling

$$W \leftarrow W - \eta \times \sqrt{\frac{\texttt{fan-out}}{\texttt{fan-in}}} \times \text{NewtonSchulz}(\nabla_W \mathcal{L}).$$

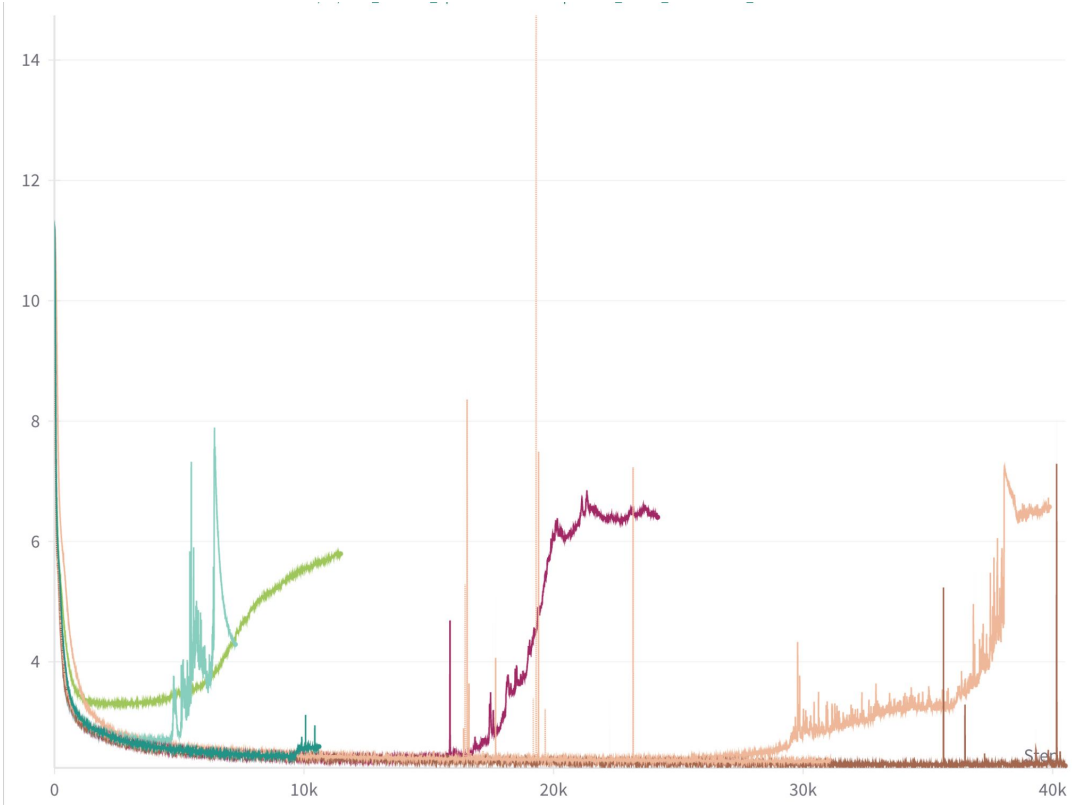
Yoloing our way

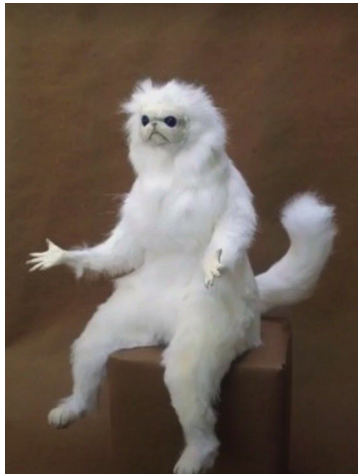


Let’s see

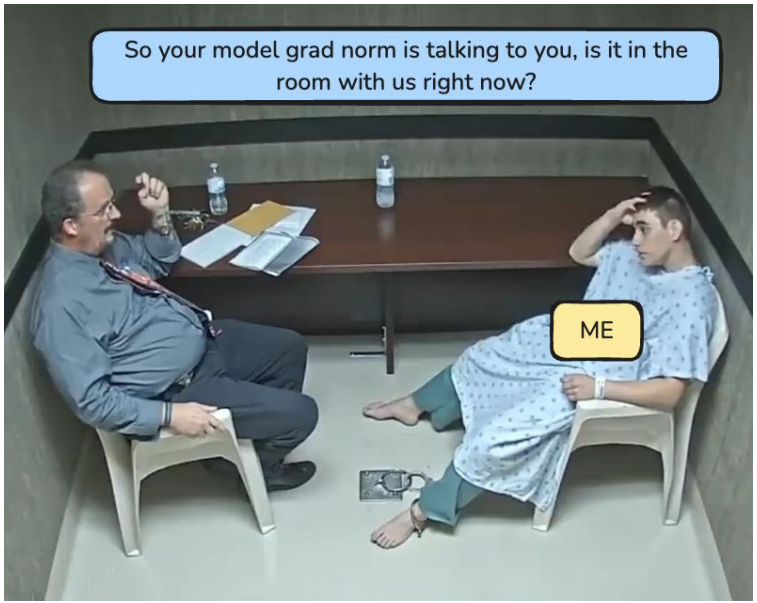

what i hope for
what i'm going to wake up to tmrw
1e-4
warmup 2k
1e-3
warmup 1k
muon runs
5e-4
warmup 1k
adamw runs
1e-4
5e-4
best ademamix runs
4
3.5
3
2.5
0
1k
2k
3k
4k
Step

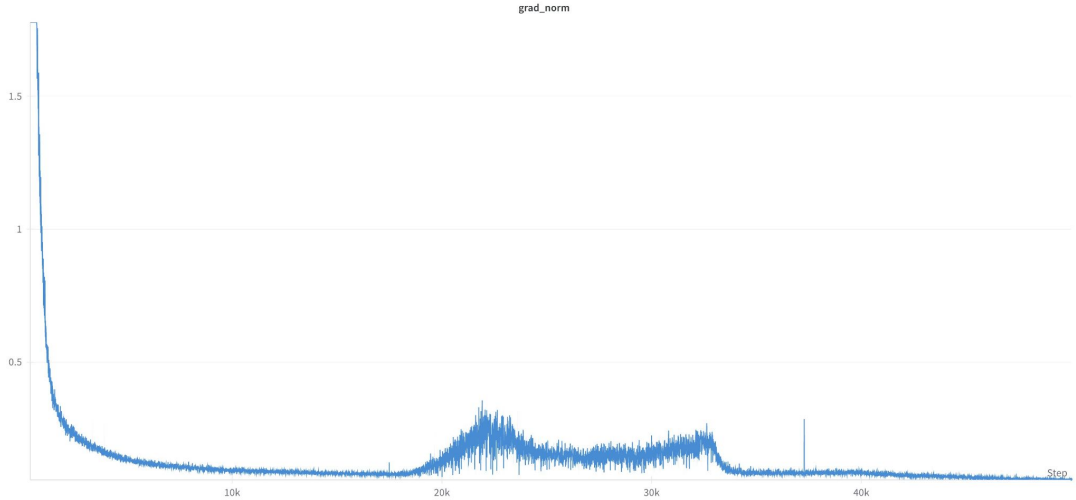
The reality is even worst





The “M” of Muon…


So your model grad norm is talking to you, is it in the room with us right now?
ME


grad_norm
1.5
1
0.5
10k
20k
30k
40k
Step

Reaction from the team

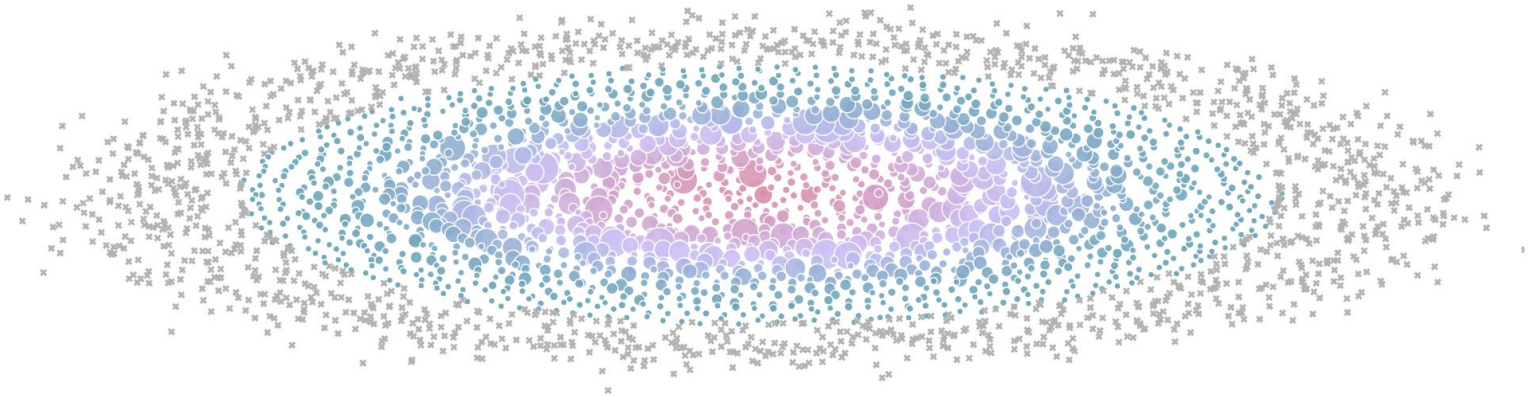
The end?

2 weeks after..

Context: some issues with our current training

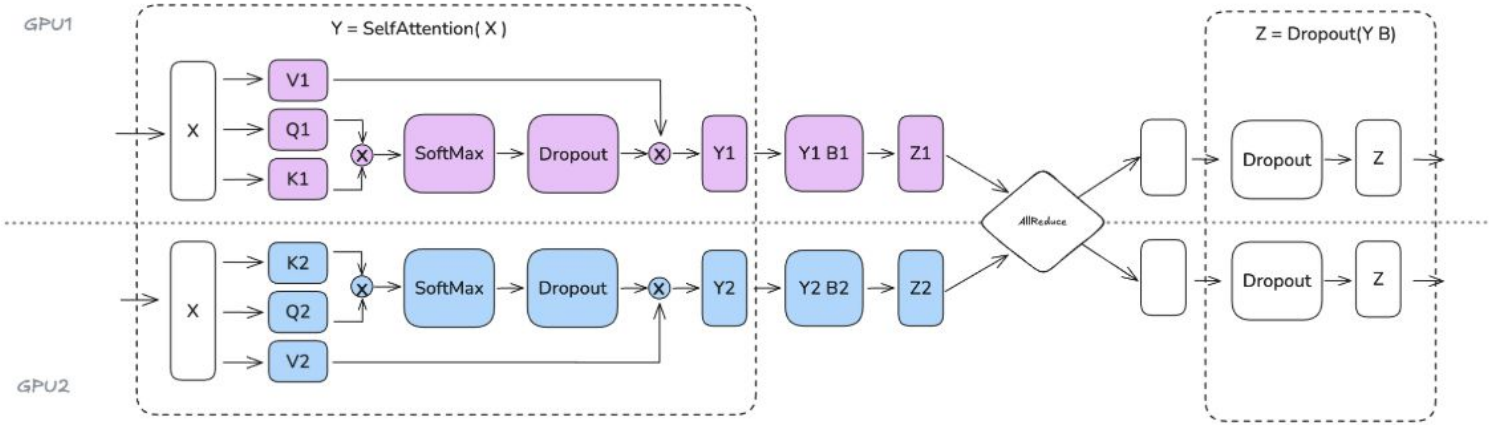
Nanotron ✨ & Distributed Training

The Ultra-Scale Playbook: Training LLMs on GPU Clusters

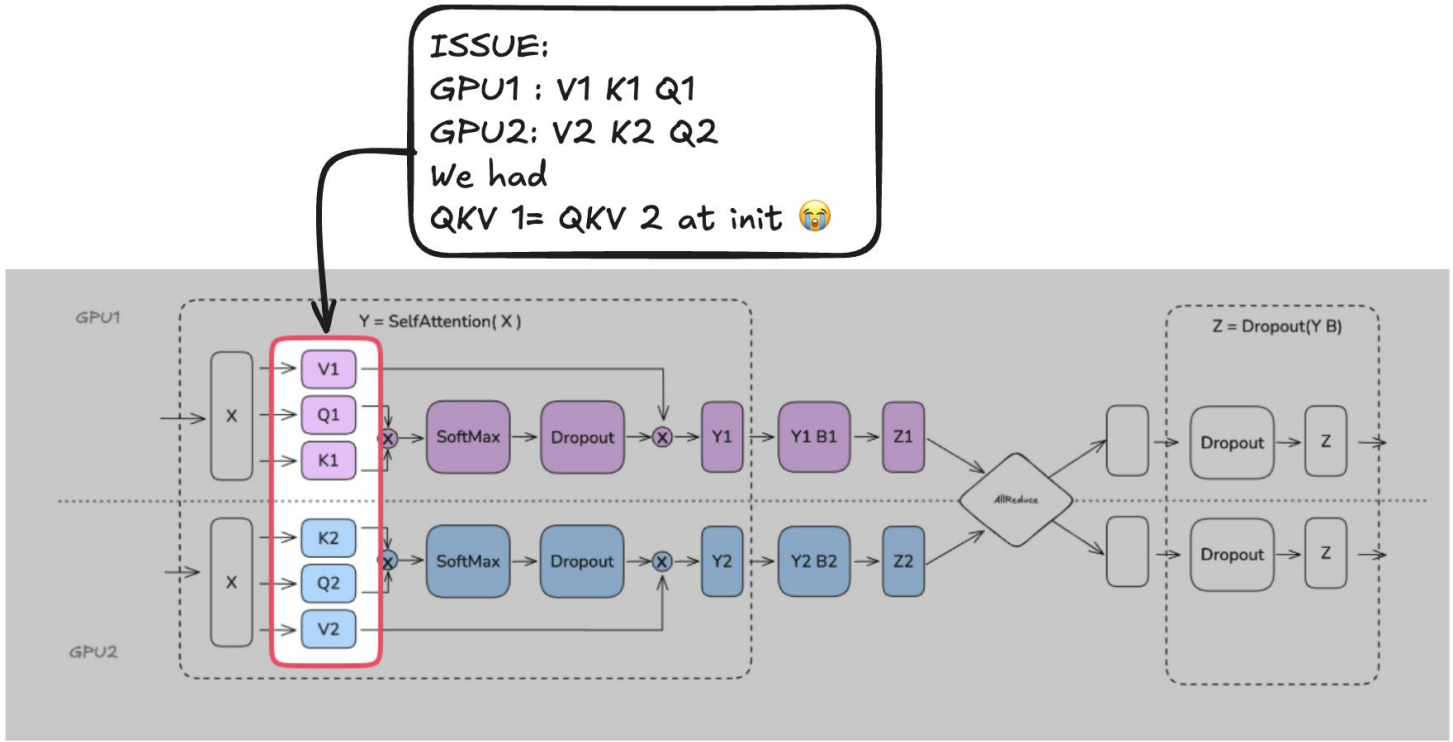


We ran over 4000 scaling experiments on up to 512 GPUs and measured throughput (size of markers) and GPU utilization (color of markers). Note that both are normalized per model size in this visualization.

Tensor Parallelism

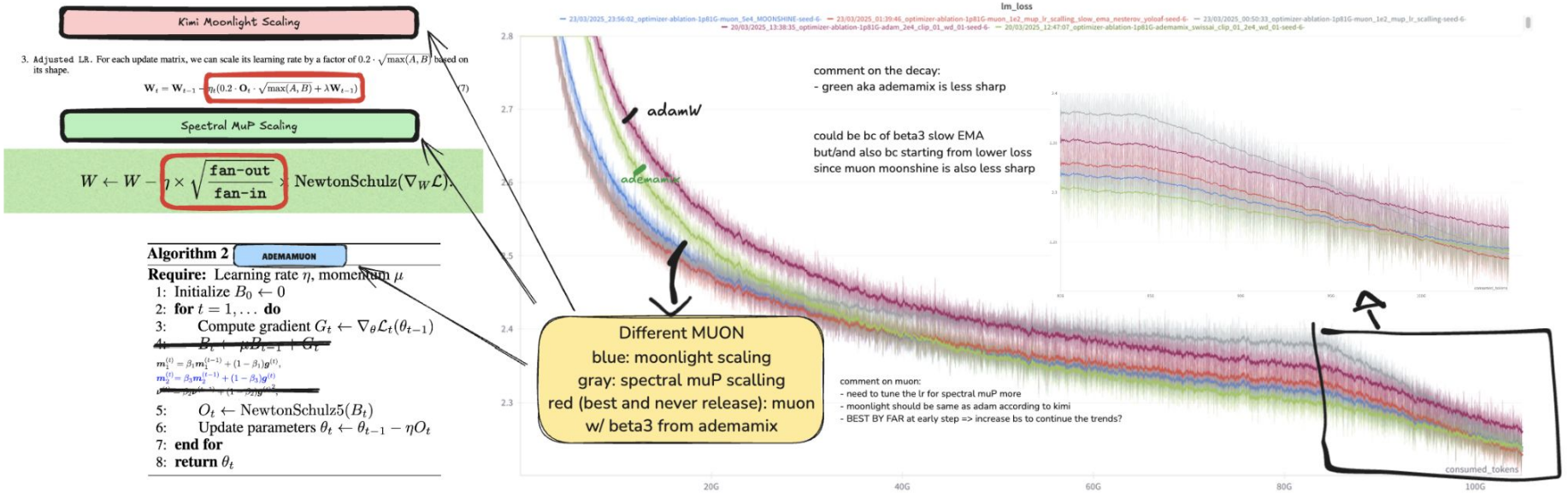


The issue…

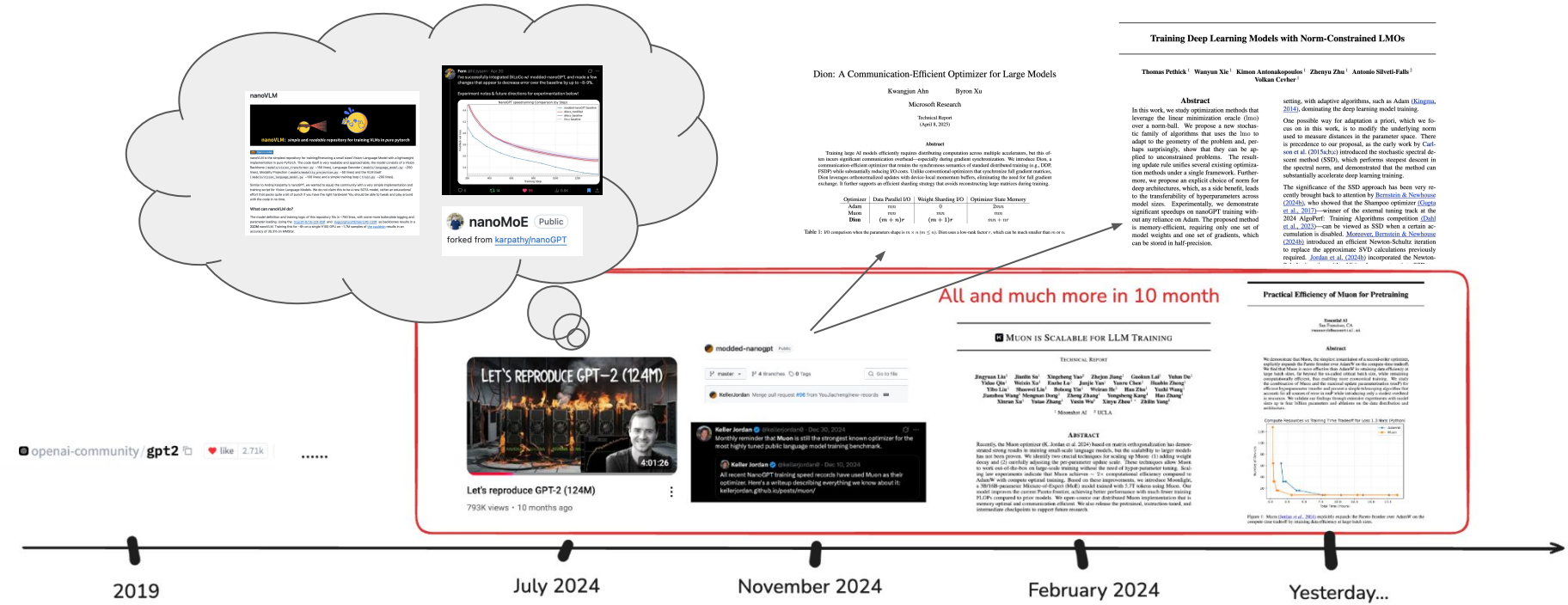

ISSUE:
GPU1 : V1 K1 Q1
GPU2: V2 K2 Q2
We had
QKV 1= QKV 2 at init 😭
GPU1
GPU2
Y = SelfAttention(X)
X
V1
Q1
K1
K2
Q2
V2
SoftMax
Dropout
Y1
Y2
Y1 B1
Y2 B2
Z1
Z2
AllReduce
Z = Dropout(Y B)
Dropout
Z

To late for this training but let’s go back to the run without the TP bug…

Going back to the results at 1.7B scale on 100B



Open science is moving faster and faster



Thanks to labs like: Kimi, Qwen, Deepseek, Essential AI, AI2, Prime Intellect, Meta and a lot more…
And also with initiative like modded-gpt by Keller Jordan and al. and Algo perf

Stay tuned and follow our team :)