Lecture 6

- RMS → RMS norm
- Maximal update parameterization. (μP)

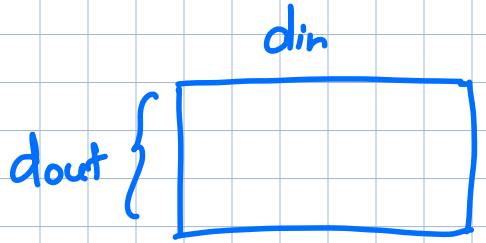
(1) Recall, induced matrix norm:

$$\|A\|_{\alpha \to \beta} = \max_{\|\vec{x}\|_\alpha = 1} \|A\vec{x}\|_\beta$$

RMS → RMS norm ⟶ Motivated by Xavier initialization.

$$\|A\|_{RMS \to RMS} = \max_{\|\vec{x}\|_{RMS} = 1} \|A\vec{x}\|_{RMS}$$

$A \in \mathbb{R}^{d_{out} \times d_{in}}$



$$A\vec{x} = \vec{h}$$

$\vec{x} \in \mathbb{R}^{d_{in}}$, $\vec{h} \in \mathbb{R}^{d_{out}}$

$$= \max_{\|\vec{x}\|_2 = \sqrt{d_{in}}} \frac{1}{\sqrt{d_{out}}} \cdot \|A\vec{x}\|_2.$$

$$= \sqrt{\frac{d_{in}}{d_{out}}} \max_{\|\vec{x}\|_2 = 1} \|A\vec{x}\|_2$$

Spectral norm $\|A\|_2$!

$$= \sqrt{\frac{d_{in}}{d_{out}}} \|A\|_2$$

② Recall, optimizer recipe.

$$\underset{\|\Delta w\| \le \eta}{\operatorname{argmin}} \; \langle \nabla_w \mathcal{L}(w), \Delta w \rangle .$$

$\nabla_w \mathcal{L}(w)$: Gradient of Loss

Δw: change in w.

$\|\Delta w\| \le \eta$: Choose appropriate norm.

→ Choose norm

→ Choose η (requires sweep!)

$\Longrightarrow$ Get optimizer.

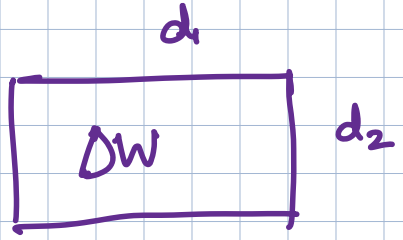
$$\underset{\|\Delta w\|_2 \le \eta}{\operatorname{argmin}} \; \langle \nabla_w \mathcal{L}(w), \Delta w \rangle$$

So can we use this induced RMS→RMS norm in our optimizer recepie?

$$\|\Delta W\|_{RMS^*} \leq \gamma \; ?$$

Key observation: If we choose this norm, we can focus on only one hyperparameter η across all layers, but the effect will be layer specific learning rates, due to fan-in and fan-out.

$$\|\Delta W\|_{RMS^*} \leq \gamma \Rightarrow \sqrt{\frac{d_1}{d_2}}\,\|\Delta W\|_2 \leq \gamma$$



$$\Rightarrow \|\Delta W\|_2 \leq \gamma \sqrt{\frac{d_1}{d_2}}$$

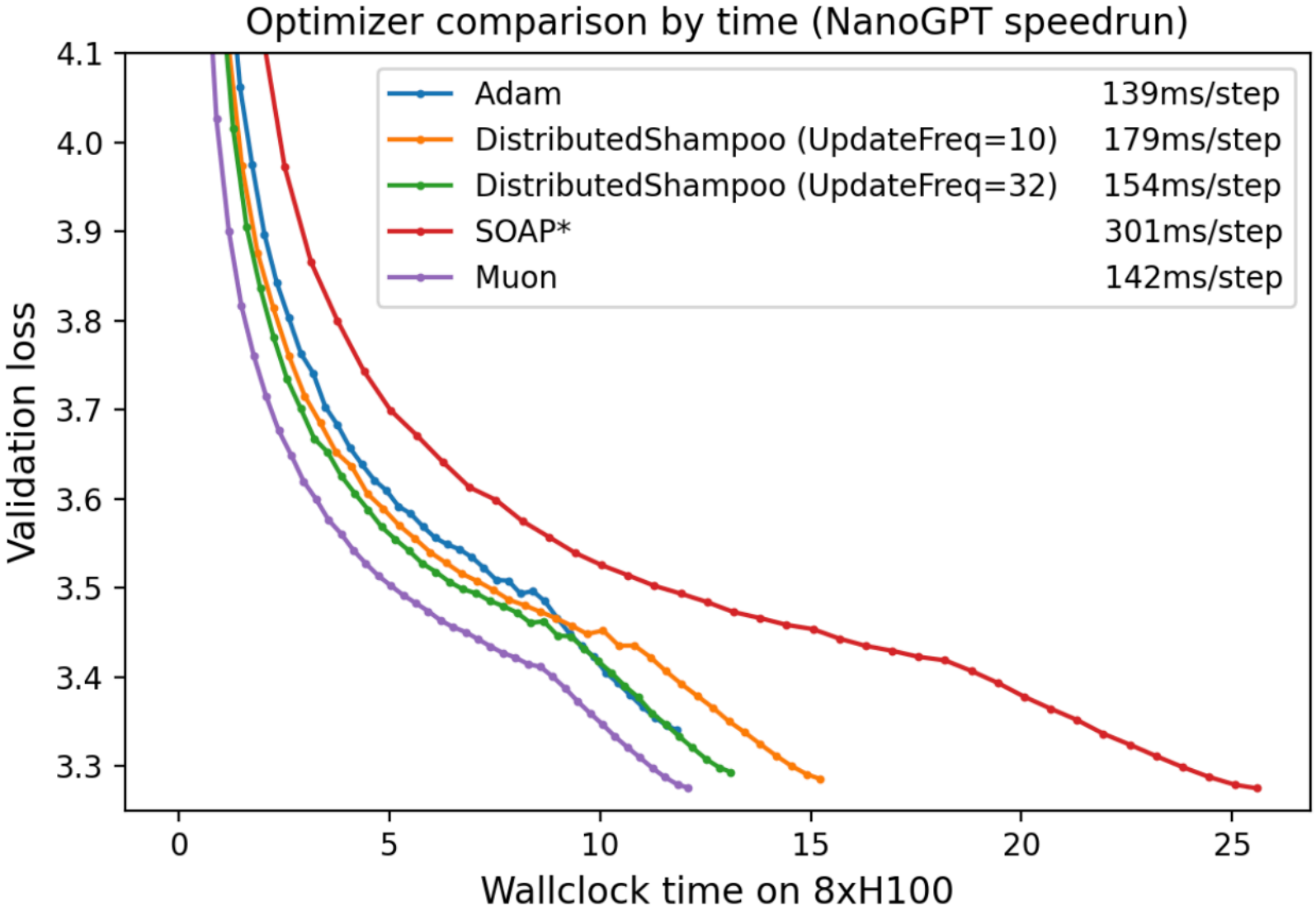
Observe that RMS→RMS norm is rescaling of the spectral norm.

→ Use in recepie will give a scaled version of problem from last time

→ Leads to the Muon optimizer.

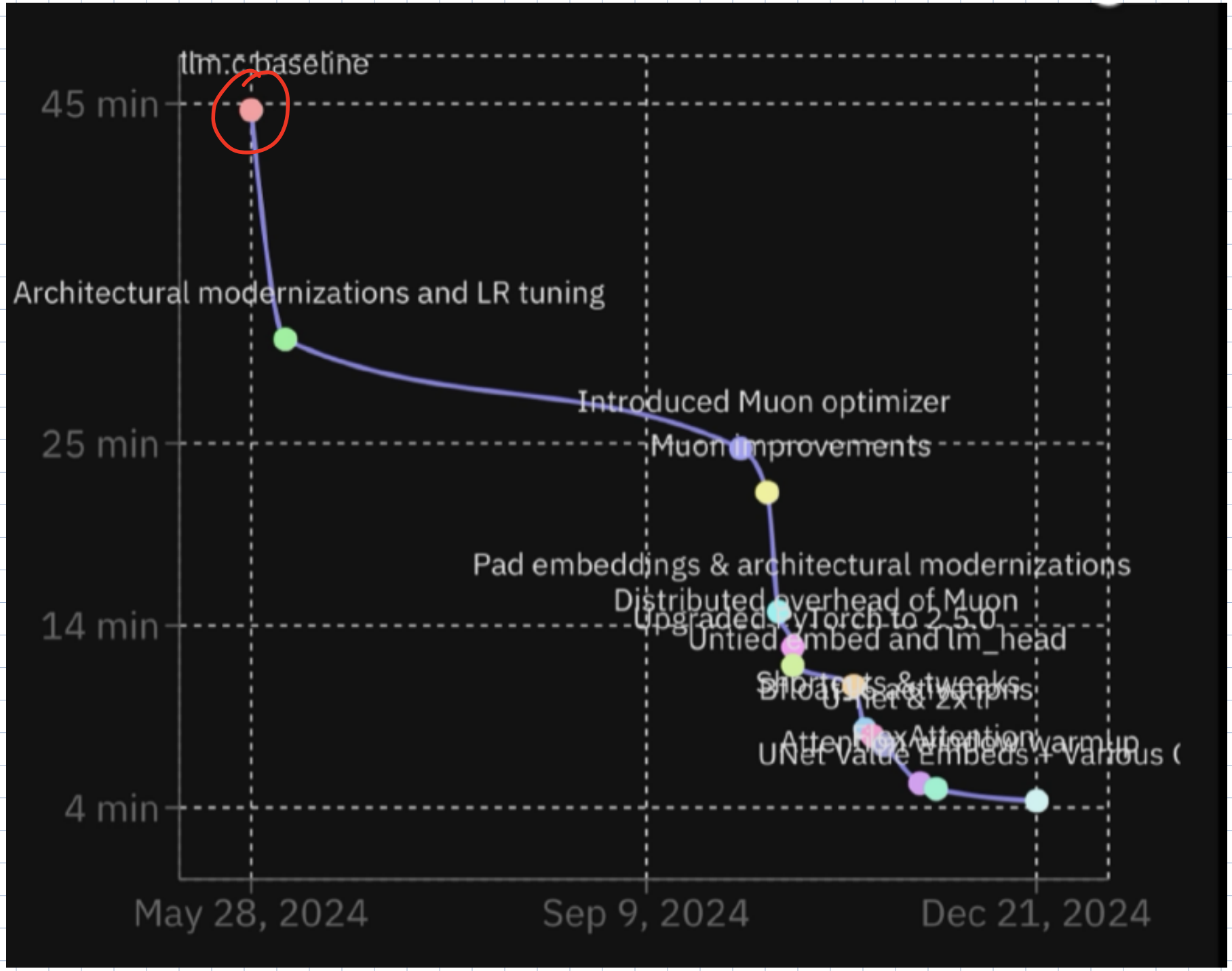
Impact.

kellerjordan.github.io.



*SOAP is under active development. Future versions will significantly improve the wallclock overhead.

NanoGPT speedrun



MUON IS SCALABLE FOR LLM TRAINING

TECHNICAL REPORT

Jingyuan Liu[1] Jianlin Su[1] Xingcheng Yao[2] Zhejun Jiang[1] Guokun Lai[1] Yulun Du[1]
Yidao Qin[1] Weixin Xu[1] Enzhe Lu[1] Junjie Yan[1] Yanru Chen[1] Huabin Zheng[1]
Yibo Liu[1] Shaowei Liu[1] Bohong Yin[1] Weiran He[1] Han Zhu[1] Yuzhi Wang[1]
Jianzhou Wang[1] Mengnan Dong[1] Zheng Zhang[1] Yongsheng Kang[1] Hao Zhang[1]
Xinran Xu[1] Yutao Zhang[1] Yuxin Wu[1] Xinyu Zhou[1] * Zhilin Yang[1]

[1] Moonshot AI [2] UCLA

KIMI

22 February 2025

Different approach.

Commit to Sign SGD/Adam.

$$W_{t+1} = W_t - \eta \cdot \text{sign}(\nabla_W \mathcal{L}(W)).$$

But I want to rethink stepsize choice
Factor in that different layers have different architectures.

Maybe RMS norm can help?

Want: $\|\Delta W\|_{\text{RMS} \to \text{RMS}} \leq ?$

Consider mini-batch size 1.

$\Rightarrow$ $\nabla_W \mathcal{L}$ gradient is low-rank, specifically rank 1.

$\Rightarrow$ $\nabla_W \mathcal{L} = \sigma \cdot \vec{u}\vec{v}^T$

$W\vec{x} = \vec{h}$

$\frac{\partial L}{\partial W} = \frac{\partial h}{\partial W}\frac{\partial L}{\partial h}$

What is the spectral norm? Frobenius norm?

$\|\nabla_W \mathcal{L}\|_2 = \sigma$ $\|\nabla_W \mathcal{L}\|_F = \sigma$

Consider $\operatorname{sgn}(\nabla_W \mathcal{L}) = S$. (elementwise)

What is rank S?

$S = \operatorname{sgn}(\sigma u \vec{v}^T) = \operatorname{sgn}(\vec{u}\vec{v}^T)$

$= \operatorname{sgn}(\vec{u}) \cdot \operatorname{sgn}(\vec{v}^T)$

$\operatorname{sgn}(u_i v_j) = \operatorname{sgn}(u_i)\operatorname{sgn}(v_j)$

$\Rightarrow$ Rank S must be 1

Consider mini-batch size 1.

$\Rightarrow$ $\nabla_W \mathcal{L}$ gradient is low-rank, specifically rank 1.

$\Rightarrow$ $\nabla_W \mathcal{L} = \sigma \cdot \vec{u}\vec{v}^T$

$$\|\nabla_W \mathcal{L}\|_2^2 = \sigma^2$$

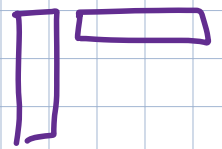
$$\|\nabla_W \mathcal{L}\|_F^2 = \text{trace}\left(\sigma^2 \vec{v}\vec{u}^T\vec{u}\vec{v}^T\right) = \sigma^2$$ (sum of singular values squared)

Consider $\text{sgn}(\nabla_W \mathcal{L}) = S$. (elementwise)

What is rank S?

$$\text{sgn}(S) = \text{sgn}(\sigma \vec{u}\vec{v}^T)$$
$$= \text{sgn}(\vec{u}\vec{v}^T)$$
$$= \text{sgn}(\vec{u}) \cdot \text{sgn}(\vec{v}^T)$$

$\text{sgn}(u_i v_j) = \text{sgn}(u_i) \cdot \text{sgn}(v_j)$



So S is also rank 1.

$$W_{t+1} = W_t - \underbrace{\eta \cdot S}_{\Delta W}.$$

Recall, we want ΔW to be small.

$$\|\Delta W\|_{RMS \to RMS} \leq \gamma$$

$$\eta \cdot \|S\|_{RMS \to RMS} \leq \gamma \implies \eta \cdot \sqrt{\frac{d_{in}}{d_{out}}} \cdot \|S\|_2 \leq \gamma$$

What is $\|S\|_2$?

Example: $A = \begin{bmatrix} +2 & -1 \\ -3 & +1 \end{bmatrix}$

$$P = \text{Sgn}(A) = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$(-1)^2 = 1^2 = 1$

$$\|P\|_F^2 = 2 \times 2 \times 1 = 4$$

$$\|A\|_F^2 = \sum_{\text{singular values}} \sigma_i^2 = \text{trace}(A^T A) = \text{trace}(A A^T) = \sum_{i,j} a_{ij}^2$$

Recall, we want ΔW to be small.

$$\|\Delta W\|_{RMS \to RMS} \leq \gamma$$

$$\eta \cdot \|S\|_{RMS \to RMS} \leq \gamma \implies \eta \cdot \sqrt{\frac{d_{in}}{d_{out}}} \cdot \|S\|_2 \leq \gamma$$

What is $\|S\|_2$?

S is rank 1, so spectral norm = Frobenius norm.

Entries are just ± 1.

$$\|S\|_F^2 = d_{in} \times d_{out}$$

$$\implies \|S\|_2 = \sqrt{d_{in} \times d_{out}}$$

$$\implies \eta \cdot \sqrt{\frac{d_{in}}{d_{out}}} \cdot \sqrt{d_{in} \times d_{out}} \leq \gamma$$

$$\implies \eta \leq \frac{1}{d_{in}} \cdot \gamma$$

Learning rate can scale with layer.

So my Adam / signSGD learning rate should scale as $\frac{1}{d_{in}}$

to maintain $\|\Delta W\|_{RMS \to RMS} \leq ?$

$\longrightarrow$ This captures an essence of the μP parameterization (Maximal update parameterization)

Yang et al. 2022.

Notes:

(1) As batch size increases, Frobenious norm will only be an upper bound to spectral norm.

So our bound will be conservative.

GPT-3, 2020.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

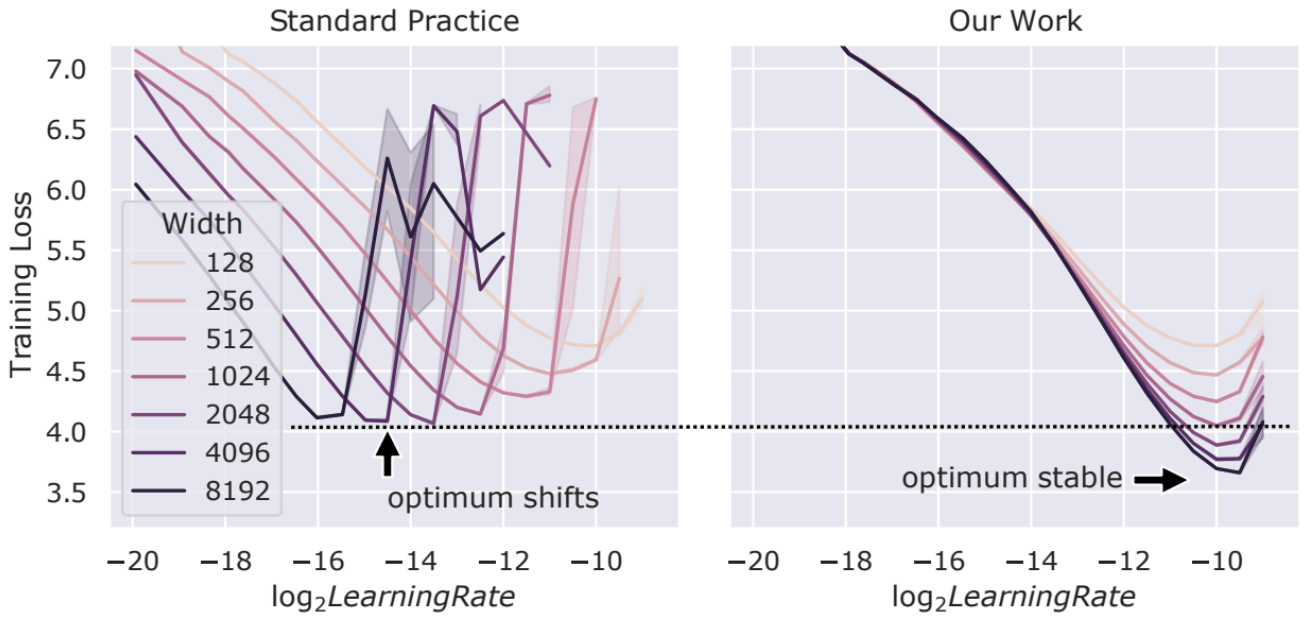


Figure 1: Training loss against learning rate on Transformers of varying d_{model} trained with Adam. Conventionally and in contrast with our technique, different widths do not share the same optimal hyperparameter; wider networks do not always perform better than narrower ones; in fact they underperform the same-width networks in our technique even after tuning learning rate (see dashed line). See Sections 3 and 4 for experimental setup.

Yang etal 2022.

What is parameterization*? Why does it matter?

* μP: How to change hyp. params when NN width changes.

→ About choosing the right "units"

→ A key challenge for neural net training is hyperparameter search.

→ Particularly bad if the network is large.

→ Experimenting is very expensive.

→ Can we optimize hyperparameters on smaller networks and then "transfer" them to larger ones? What is the right scaling?

Example: $x_1, x_2, \cdots x_n$ iid from zero-mean, unit variance distribution.

Then $\frac{1}{\sqrt{n}}(x_1+x_2+\cdots+x_n) \xrightarrow[n\to\infty]{} N(0,1)$

But $\frac{1}{n}(x_1+x_2+\cdots+x_n) \xrightarrow[n\to\infty]{} 0$

$\frac{1}{1}(x_1+x_2+\cdots+x_n) \xrightarrow[n\to\infty]{} \infty$

$\frac{1}{\sqrt{n}}$ is the right order of "scaling factor" so that $c_n(\sum x_i)$ converges to something non-trivial.

Consider $F_n(C) \doteq \mathop{\mathbb{E}}_{x_1, x_2 \dots x_n} f\left(C(x_1 + x_2 + \dots + x_n)\right)$

$C \in \mathbb{R}$, $f : \mathbb{R} \to \mathbb{R}$ bounded, continuous.

Reparametrize $C = \alpha/\sqrt{n}$.

$$G_n(\alpha) \doteq F_n\left(\frac{\alpha}{\sqrt{n}}\right) \to \mathbb{E}\left(f(N(0, \alpha^2))\right)$$

(only function of α.)

as $n \to \infty$

$\alpha_n^* = \underset{\alpha}{\operatorname{argmin}}\ G_n(\alpha)$ → Then $\alpha_n^* \approx \alpha_N^*$ for large enough n, N

→ We can copy α_n^* from a smaller n to a larger N.

If instead $C_n^* = \underset{C}{\operatorname{argmin}}\ F_n(C)$ → Cannot transfer parameters in the same way.

Can we do something similar for NN hyperparameters?

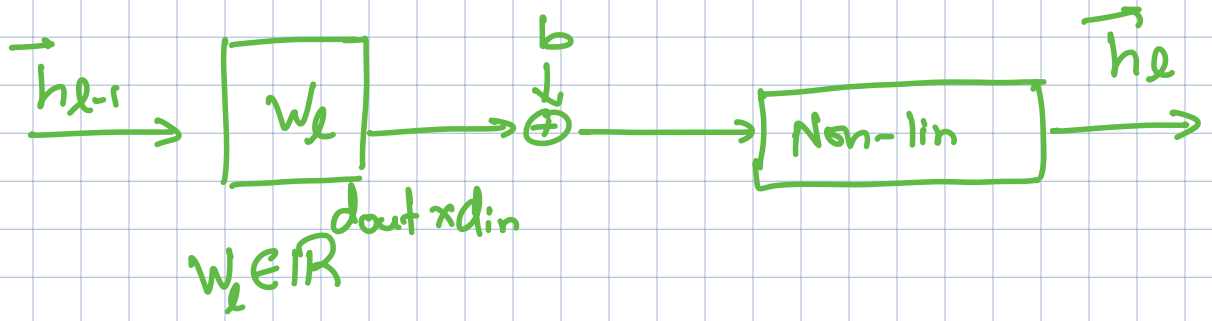
$\eta \leq \frac{1}{d_{in}} \cdot \gamma$ ← still a hyperparameter.

$\frac{1}{d_{in}}$: the "right scaling".

Conditions for Feature Learning

$\vec{h}_\ell$: hidden layer ℓ.

$\vec{h}_{\ell-1} \in \mathbb{R}^{d_{in}}$ $\quad \vec{h}_\ell \in \mathbb{R}^{d_{out}}$



Focus on scaling with network width.

Xavier initialization thinking suggests we want $\|\vec{h}_\ell\|_{RMS} \approx 1$.

$$\|\vec{h}_\ell\|_{RMS} = \frac{1}{\sqrt{d_{out}}} \|\vec{h}_\ell\|_2$$

$$\Rightarrow \quad \|\vec{h}_\ell\|_2 \approx \sqrt{d_{out}}$$

Relaxation of all entries approx 1.

$$\|\vec{h}_\ell\|_{RMS} = \Theta(1)$$

Also, want to ensure that our updates to the layers are not negligible as width grows.

$$\|\Delta \vec{h}_\ell\|_{RMS} = \Theta(1)$$

Desiderata: (1) $\boxed{\|\vec{h}_\ell\|_{RMS} = \Theta(1)}$ (2) $\boxed{\|\Delta \vec{h}_\ell\|_{RMS} = \Theta(1)}$

How can we achieve this?

For now, ignore bias + non-linearity.

$$\vec{h}_\ell = W_\ell \vec{h}_{\ell-1}$$

$\Rightarrow$ if $\vec{h}_{\ell-1}$ aligns with top singular vector of W_ℓ. [Can show this is the case]

$$\|\vec{h}_\ell\|_2 = \|W_\ell\|_2 \cdot \|\vec{h}_{\ell-1}\|_2$$

$$\Theta(\sqrt{d_{out}}) = \|W_\ell\|_2 \, \Theta(\sqrt{d_{in}})$$

$$\Rightarrow \quad \|W_\ell\|_2 = \Theta\left(\sqrt{\frac{d_{out}}{d_{in}}}\right) \quad \text{i.e} \quad \|W_\ell\|_{RMS \to RMS} = \Theta(1)$$

i.e. fixed RMS norm as we started with!

Thus, if $\|W_\ell\|_{RMS \to RMS} = \Theta(1)$ and $\|\vec{h}_\ell\|_{RMS} = \Theta(1)$, then

$$\|\vec{h}_\ell\|_{RMS} = \Theta(1)$$

What about $\|\Delta h_\ell\|_{RMS}$?

$\rightarrow$ Need one more condition.

$$\|\Delta W_\ell\|_{RMS} = \Theta(1) \implies \|\Delta W_\ell\|_2 = \Theta\left(\sqrt{\frac{d_{out}}{d_{in}}}\right)$$

$$\vec{h}_\ell = W_\ell \vec{h}_{\ell-1}$$

$$\implies \Delta\vec{h}_\ell = \Delta W_\ell \cdot \vec{h}_{\ell-1} + W_\ell \cdot \Delta\vec{h}_{\ell-1}$$

$$\implies \|\Delta\vec{h}_\ell\|_2 \le \|\Delta W_\ell\|_2 \cdot \|\vec{h}_{\ell-1}\|_2 + \|W_\ell\|_2 \|\Delta\vec{h}_{\ell-1}\|_2.$$

$$= \boxed{?}\,\Theta(\sqrt{d_{in}}) + \Theta\left(\sqrt{\frac{d_{out}}{d_{in}}}\right)\cdot\Theta(\sqrt{d_{in}})$$

$$\implies \|\Delta h_\ell\|_{RMS} = \Theta(1)$$

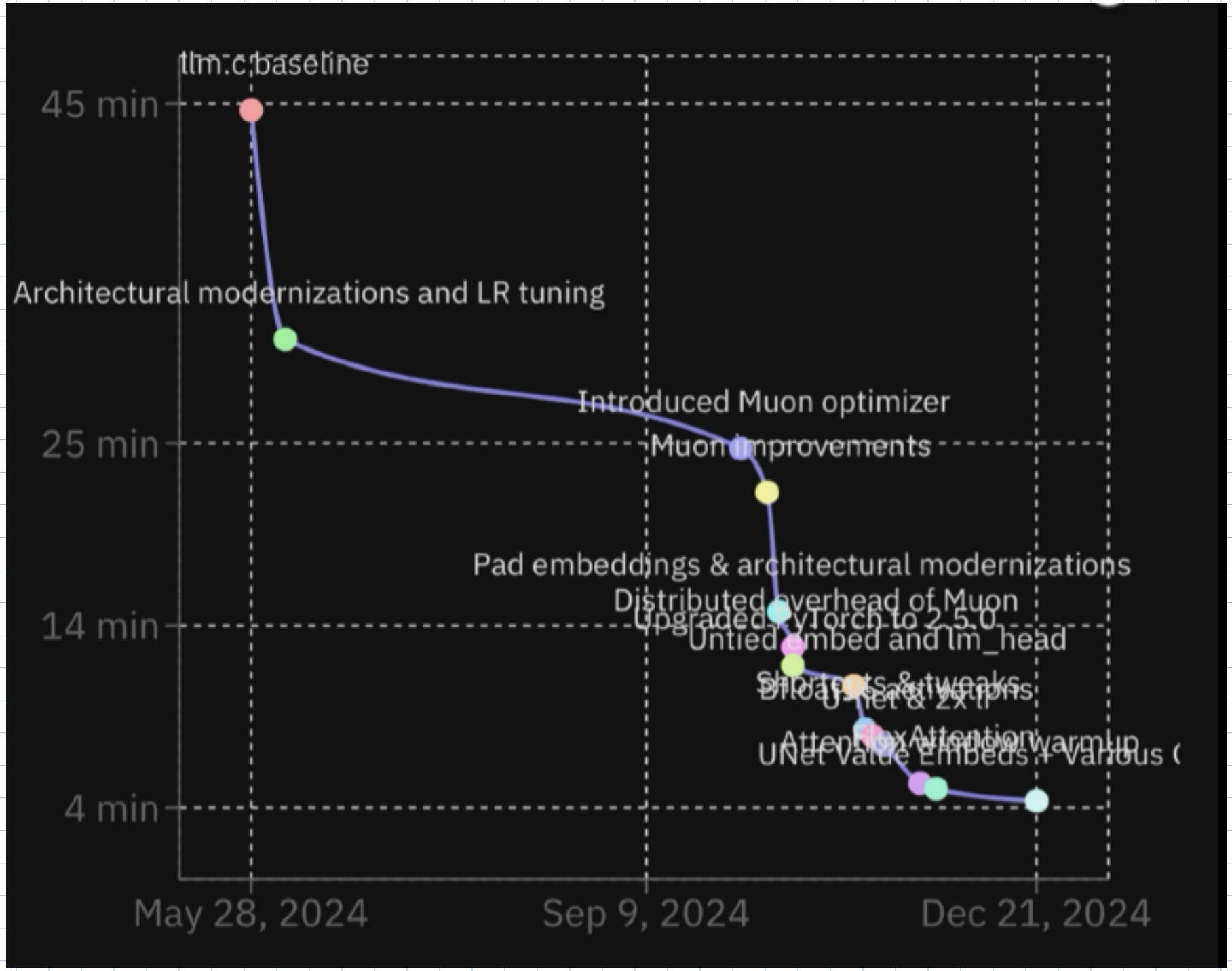
Conditions: (1) $\|W\|_{RMS\to RMS} = \Theta(1)$ (2) $\|\Delta W\|_{RMS\to RMS} = \Theta(1)$

$$\Delta \vec{h_\ell} = \Delta W_\ell \cdot \vec{h}_{\ell-1} + W_\ell \cdot \Delta \vec{h}_{\ell-1}$$

$$\| \Delta \vec{h_\ell} \|_2 = \| \Delta W \vec{h}_{\ell-1} + W_\ell \Delta \vec{h}_{\ell-1} \|_2$$

$$\leq \| \Delta W \|_2 \| \vec{h}_{\ell-1} \|_2 + \| W_\ell \|_2 \| \Delta \vec{h}_{\ell-1} \|_2$$

→ apply triangle inequality
+ sub-multiplicativity property


llm.c baseline
45 min
Architectural modernizations and LR tuning
Introduced Muon optimizer
25 min
Muon improvements
Pad embeddings & architectural modernizations
Distributed overhead of Muon
Upgraded PyTorch to 2.5.0
14 min
Untied embed and lm_head
Shortcuts & tweaks
4 min
May 28, 2024
Sep 9, 2024
Dec 21, 2024

Consider mini-batch signSGD. Batch size 1..

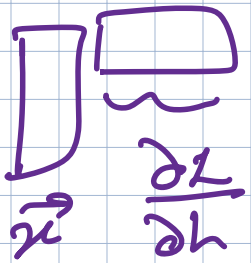
$\nabla_W \mathcal{L}(W)$ must be in the span of the datapoint.

let $W\vec{x} = \vec{h}$

$W = [w_{ij}]$ $\begin{matrix} w_{11} & w_{12} \\ w_{21} & \end{matrix}$

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial h}{\partial W} \cdot \frac{\partial \mathcal{L}}{\partial h}$$

$$h_i = \sum_j w_{ij} \cdot x_j$$



$$\frac{\partial h_i}{\partial w_{ij}} = x_j$$

$$\frac{\partial \mathcal{L}}{\partial W} = \vec{x} \frac{\partial \mathcal{L}}{\partial h}$$

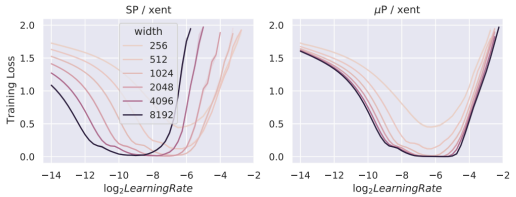


Figure 3: MLP width different hidden sizes trained for 20 epoch on CIFAR-10 using SGD. **Left** uses standard parametrization (SP); **right** uses maximal update parametrization (μP). μP networks exhibit better learning rate stability than their SP counterparts.