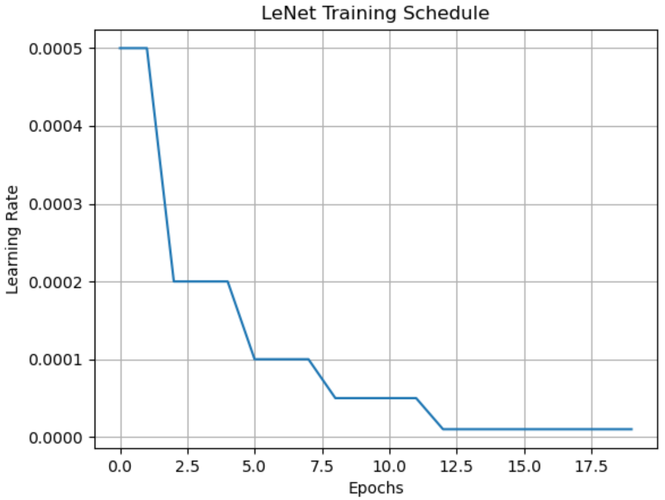
# Scaling Schedule-Free and Learning Rate Free Learning To LLMs

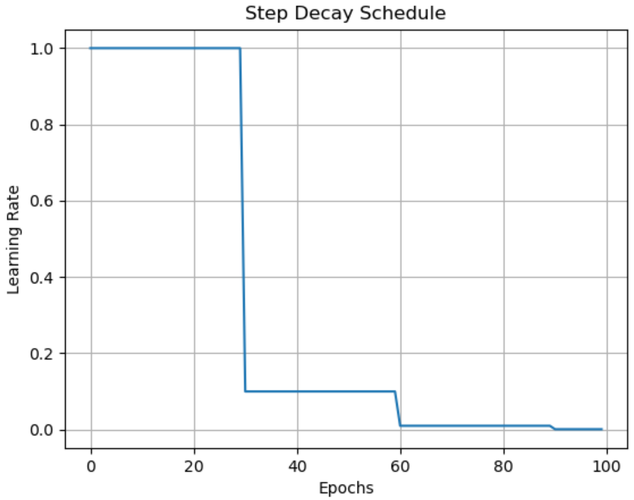
**Aaron Defazio**

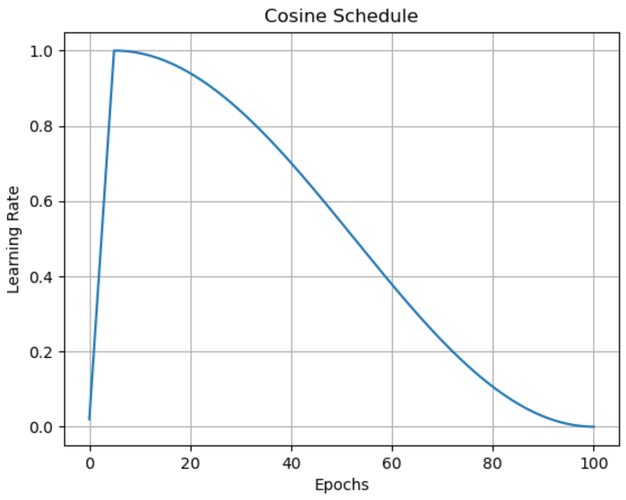
Research Scientist

FAIR, Meta Superintelligence Labs

Meta




1989 - 1998

2012

2017

*Handwritten Digit Recognition with a Back-Propagation Network*

Y. Le Cun, B. Boser, J. S. Denker, D. Henderson,
R. E. Howard, W. Hubbard, and L. D. Jackel
AT&T Bell Laboratories, Holmdel, N. J. 07733

**Gradient-Based Learning Applied to Document Recognition**

YANN LECUN, MEMBER, IEEE, LÉON BOTTOU, YOSHUA BENGIO, AND PATRICK HAFFNER

**ImageNet Classification with Deep Convolutional Neural Networks**

**Alex Krizhevsky**
University of Toronto
kriz@cs.utoronto.ca

**Ilya Sutskever**
University of Toronto
ilya@cs.utoronto.ca

**Geoffrey E. Hinton**
University of Toronto
hinton@cs.utoronto.ca

"we adjusted manually throughout training. The heuristic which we followed was to divide the learning rate by 10 when the validation error rate stopped improving with the current learning rate"

Published as a conference paper at ICLR 2017

SGDR: STOCHASTIC GRADIENT DESCENT WITH WARM RESTARTS

**Ilya Loshchilov & Frank Hutter**
University of Freiburg
Freiburg, Germany,
{ilya,fh}@cs.uni-freiburg.de

# A different perspective on Scheduling: Schedules are just an online-to-batch conversion!

$$x_{t+1} = x_t - \gamma_t g_t$$

$$\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$$

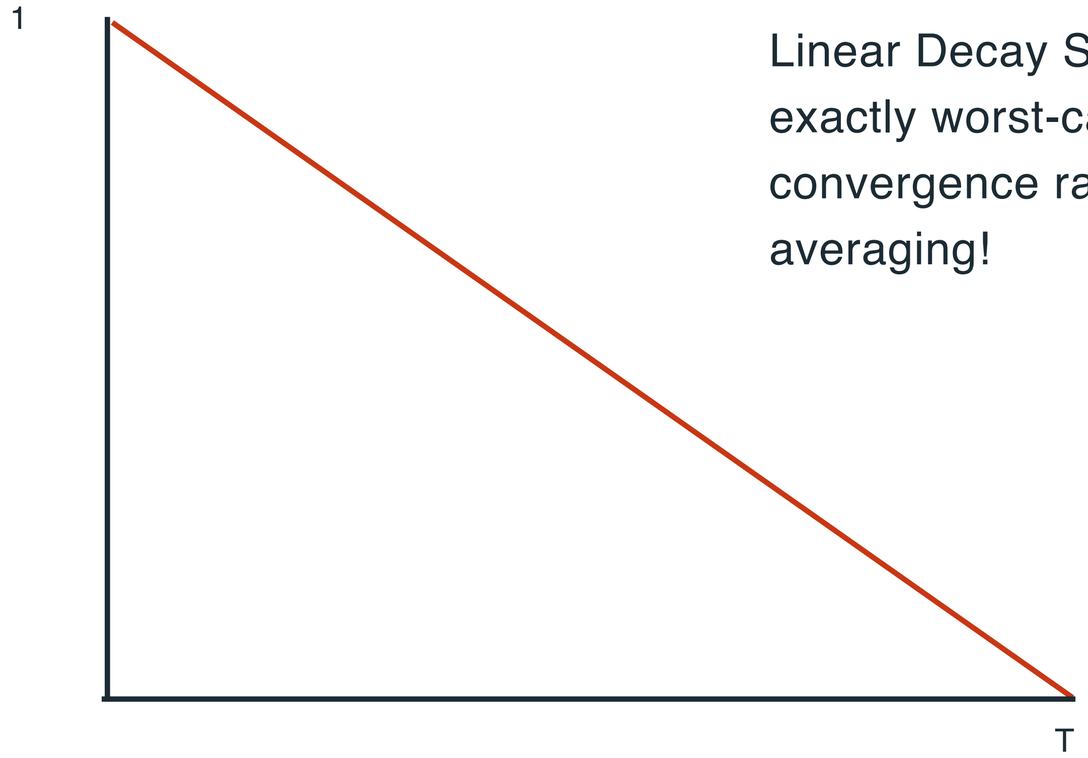
The schedules used by experimentalists are not replacing this $D/G\sqrt{T}$ part!

They are actually replacing **averaging.**

High-Performance schedules arise naturally from **theory** by analyzing the last iterate $x_T$ rather than $\bar{x}_T$

Theoretically Optimal Schedules
(for convex problems)

$$\gamma_t = \frac{D}{G\sqrt{T}} \cdot \left(1 - \frac{t}{T}\right)$$




Linear Decay Schedule

Linear Decay Schedules give exactly worst-case optimal convergence rates without averaging!

$$f(\bar{x}_T) - f_* \leq \frac{DG}{\sqrt{T}}$$

**EXACT CONVERGENCE RATE OF THE LAST ITERATE IN SUBGRADIENT METHODS**

MOSLEM ZAMANI* AND FRANÇOIS GLINEUR †

---

**Optimal Linear Decay Learning Rate Schedules and Further Refinements**

---

**Aaron Defazio**
Fundamental AI Research Team, Meta

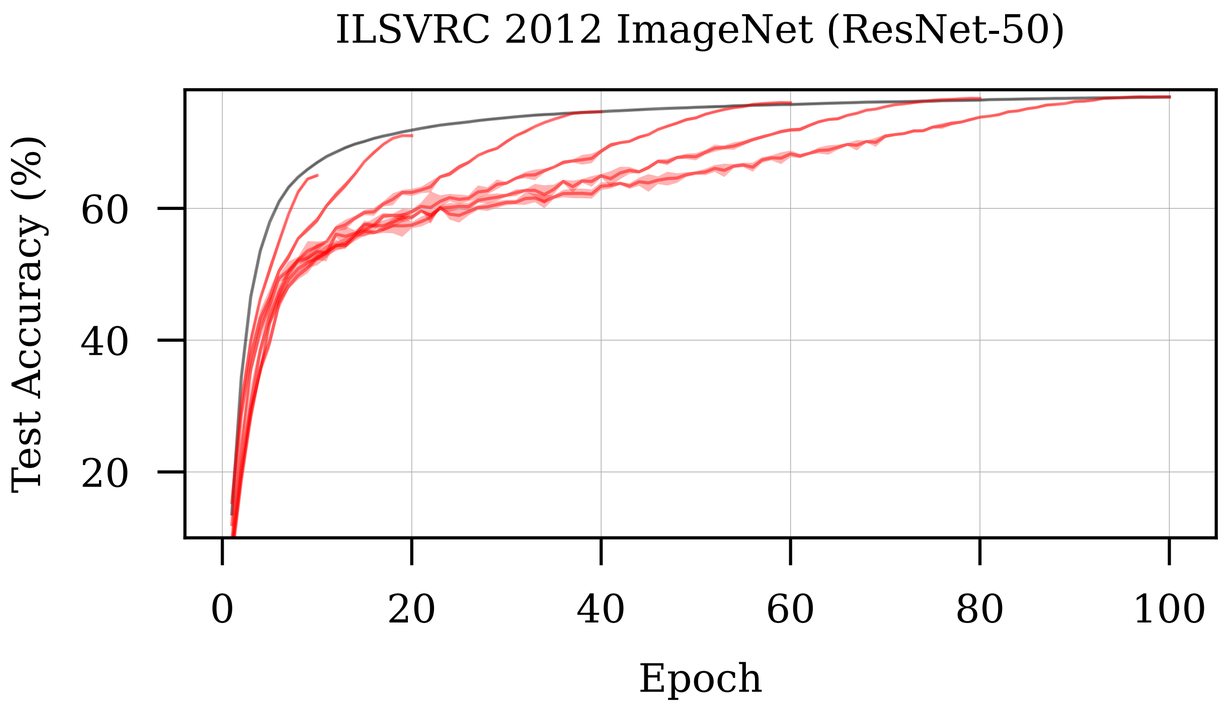
**Ashok Cutkosky**
Boston University

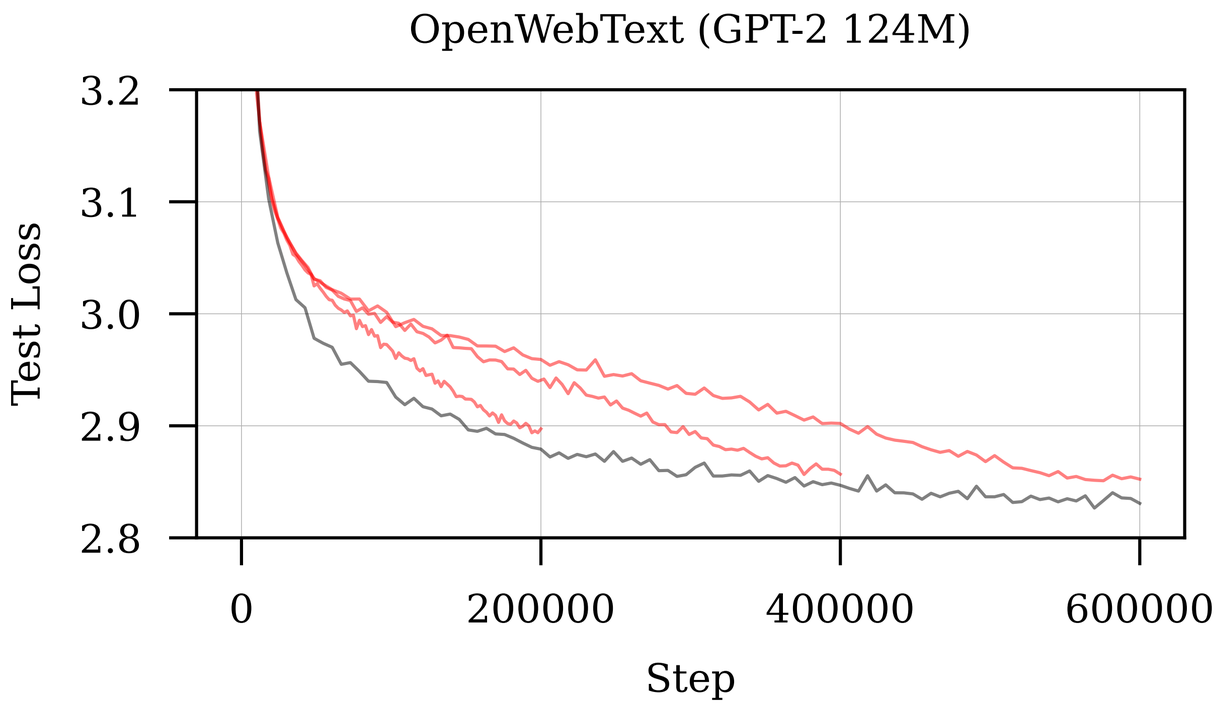
**Harsh Mehta**
Google Research

**Konstantin Mishchenko**
Samsung AI Center

# Beyond Learning Rate Schedules

# Varying cosine-schedule length shows how Schedule-Free closely tracks the Pareto frontier of Loss v.s. training time.



# Schedule-Free Learning - closing the theory/practice gap

- An alternative to schedules that doesn't need to know the stopping time T in advance
- Maintains the same **theoretically optimal** rate of convergence as linear decay schedules and Polyak averaging
- Works as well and often better than cosine or linear decay schedules!

---

**The Road Less Scheduled**

---


**Aaron Defazio**[1]
Fundamental AI Research Team, Meta

**Xingyu (Alice) Yang**[2]
Fundamental AI Research Team, Meta

**Harsh Mehta**
Google Research

**Konstantin Mishchenko**
Samsung AI Center

**Ahmed Khaled**
Princeton University

**Ashok Cutkosky**[3]
Boston University

[1]Research Co-lead [2] Engineering Co-lead [3] Senior Author

# Schedule-Free Learning Paradigm

Interpolation beta=0.9

(A kind of momentum)

$$y_t = (1-\beta)z_t + \beta x_t$$

Base optimizer update

$$z_{t+1} = z_t - \gamma \nabla f(y_t)$$

$$x_{t+1} = \left(1 - \frac{1}{t+1}\right) x_t + \frac{1}{t+1} z_{t+1}$$

Running Average

Equivalent to:

$$x_t = \frac{1}{t} \sum_{i=1}^{t} z_i$$

# Schedule-Free does momentum in a different, more gradual way….

$$y_t = (1-\beta)z_t + \beta x_t$$

$$z_{t+1} = z_t - \gamma \nabla f(y_t)$$

$$x_{t+1} = \left(1 - \frac{1}{t+1}\right) x_t + \frac{1}{t+1} z_{t+1}$$

$\beta = 0.9$ results in the current gradient evaluation point y containing 0.1 of the most recent gradient $g_{t-1}$

Classical momentum does the same thing! 0.1 of the most recent gradient is included in the step

$$m_{t+1} = \beta m_t + \left(1-\beta\right) \nabla f(x_t)$$

$$x_{t+1} = x_t - \alpha_t m_{t+1}$$

But classical momentum incorporates the rest of the gradient over the next ~10 steps, whereas Schedule-Free incorporates it much **slower**, of the reminder of optimization

# Online-to-Batch Theory

**Theorem 2.** *Let $F$ be a convex function. Let $\zeta_1, \dots, \zeta_T$ be an iid sequence such that $F(x) = \mathbb{E}_\zeta[f(x,\zeta)]$. Let $z_1, \dots, z_T$ be arbitrary vectors and let $w_1, \dots, w_T$ and $\beta_1, \dots, \beta_T$ be arbitrary numbers in $[0,1]$ such that $z_t$, $w_t$ and $\beta_t$ are independent of $\zeta_t, \dots, \zeta_T$. Set:*

$$x_t = \frac{\sum_{i=1}^t w_i z_i}{\sum_{i=1}^t w_i} = x_{t-1} \underbrace{\left(1 - \frac{w_t}{\sum_{i=1}^t w_i}\right)}_{\triangleq 1 - c_t} + \underbrace{\frac{w_t}{\sum_{i=1}^t w_i}}_{\triangleq c_t} z_t \tag{9}$$

$$y_t = \beta_t x_t + (1-\beta_t) z_t \tag{10}$$

$$g_t = \nabla f(y_t, \zeta_t). \tag{11}$$

*Then we have for all $x_\star$:*

$$\mathbb{E}[F(x_T) - F(x_\star)] \leq \frac{\mathbb{E}[\sum_{t=1}^T w_t \langle g_t, z_t - x_\star \rangle]}{\sum_{i=1}^T w_i}. \tag{12}$$

Beta is a free “momentum-like” parameter that can be tuned in the range [0,1] for best performance.

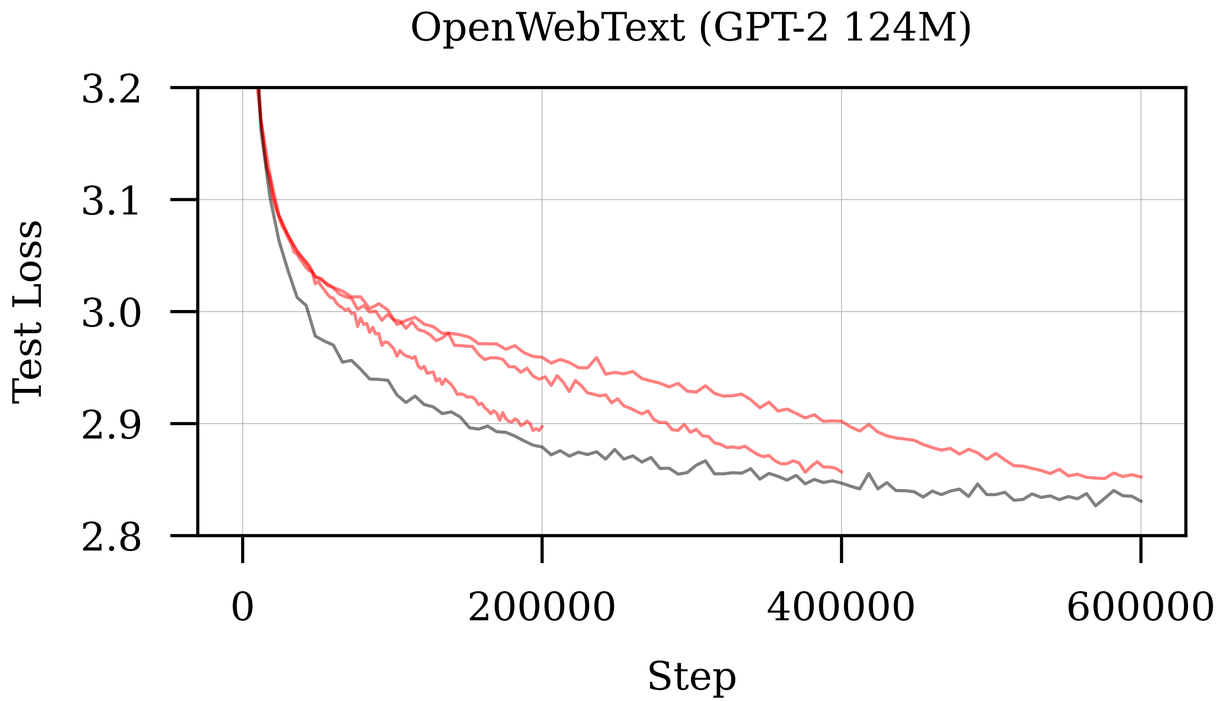
Unlike standard momentum, it won’t break your convergence theory!

SCALING YOUR RESEARCH

# Can Schedule-Free Learning be applied to (*larger*)-Large Language Models?

Original Schedule-Free paper experiments:

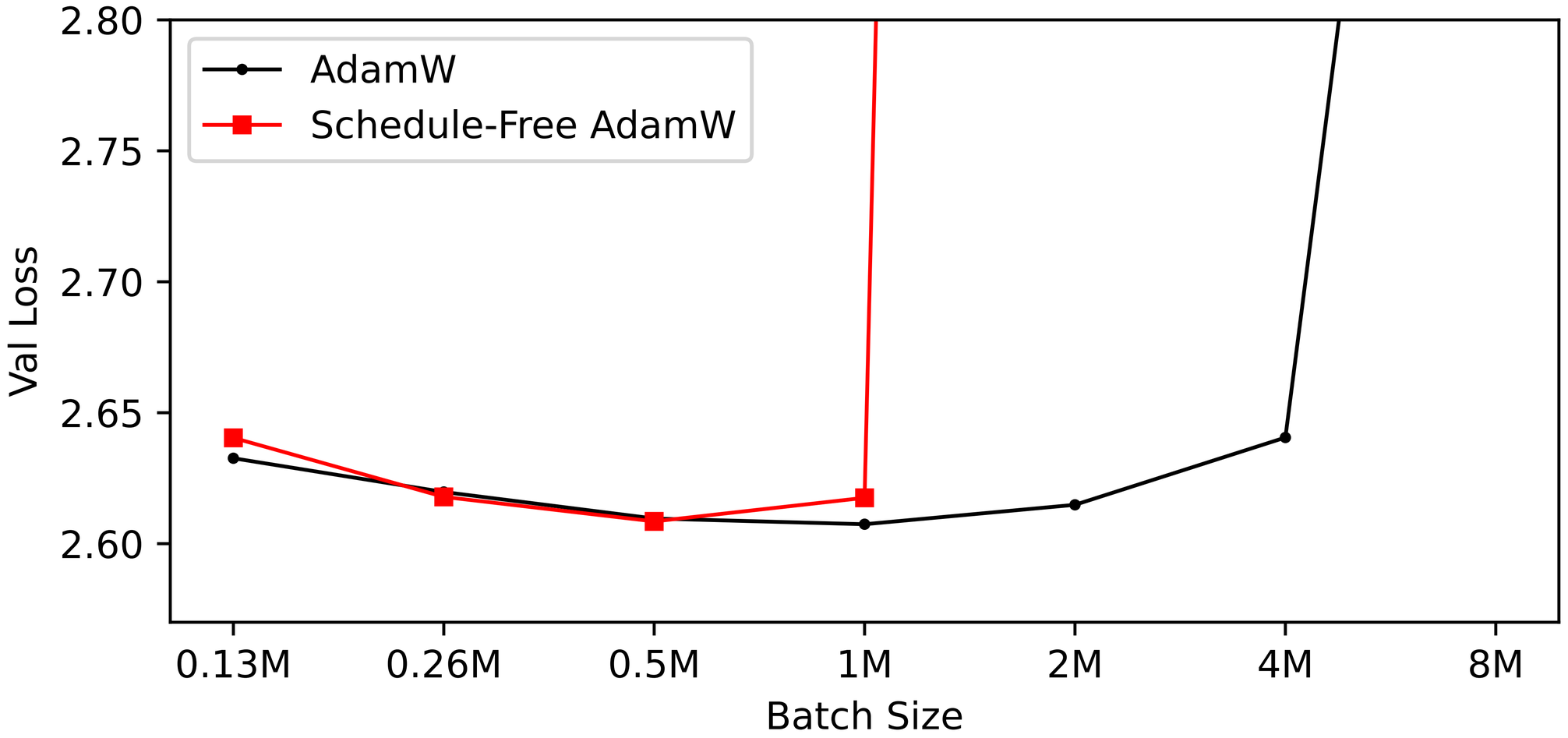
- Trained using warm-starting
- Trained for a large number of steps (over-trained?)
- Trained with a small batch size
- Very small model (124M)




**General case**: without warm-starting, for varying token budgets, larger batches, and for larger (1B+) models?

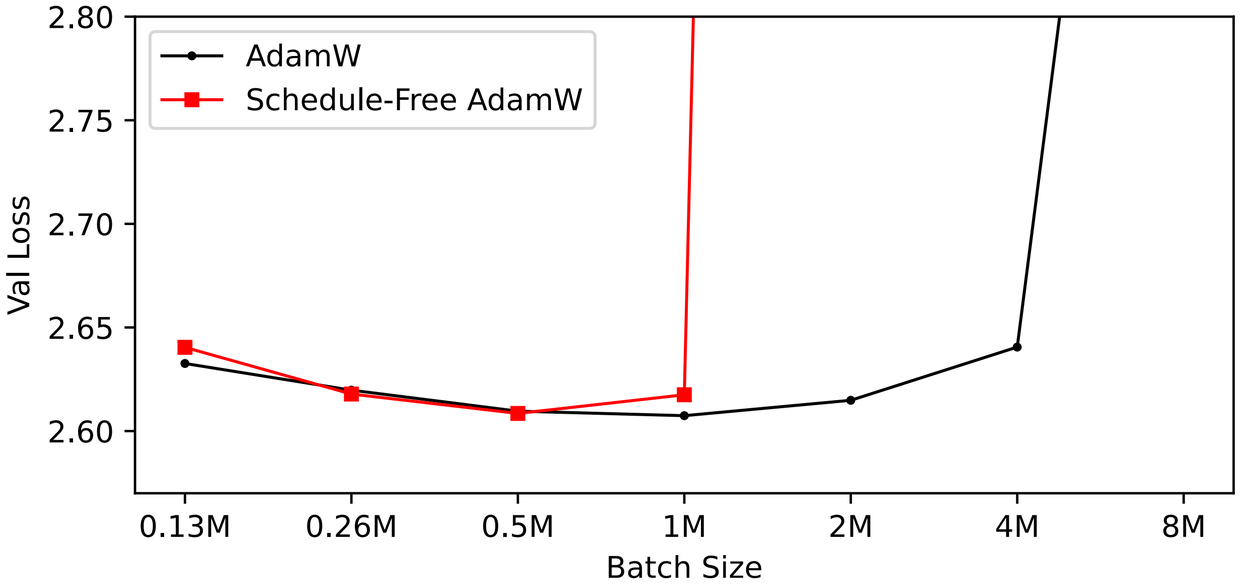
# Complete failure initially

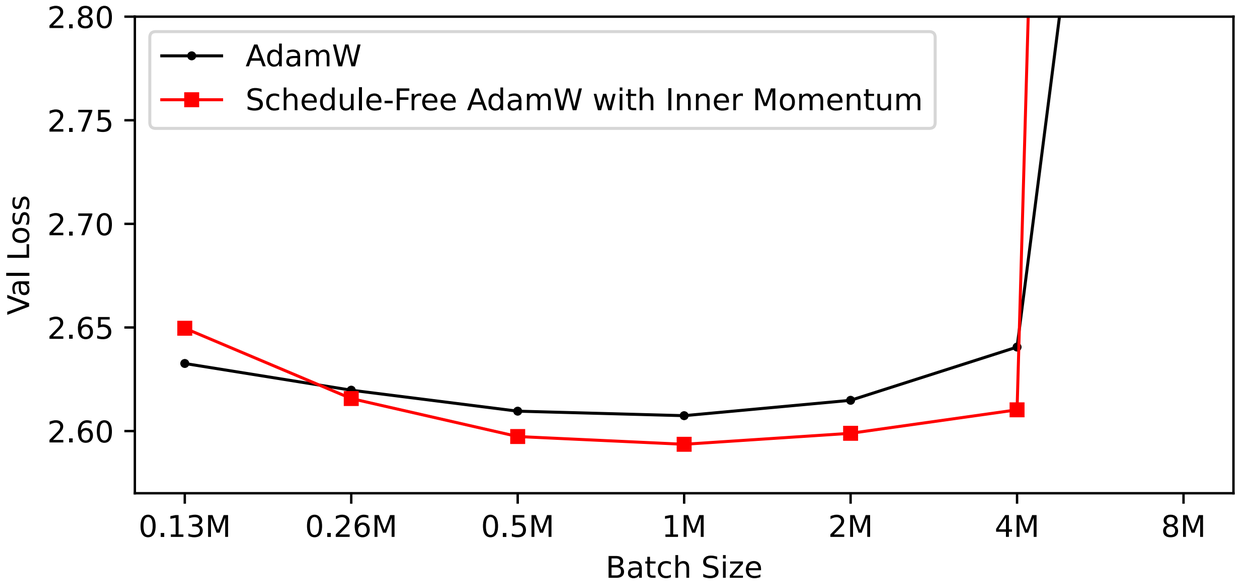
# Large Batch-Sizes break it!



# FIX: Add back in inner momentum!

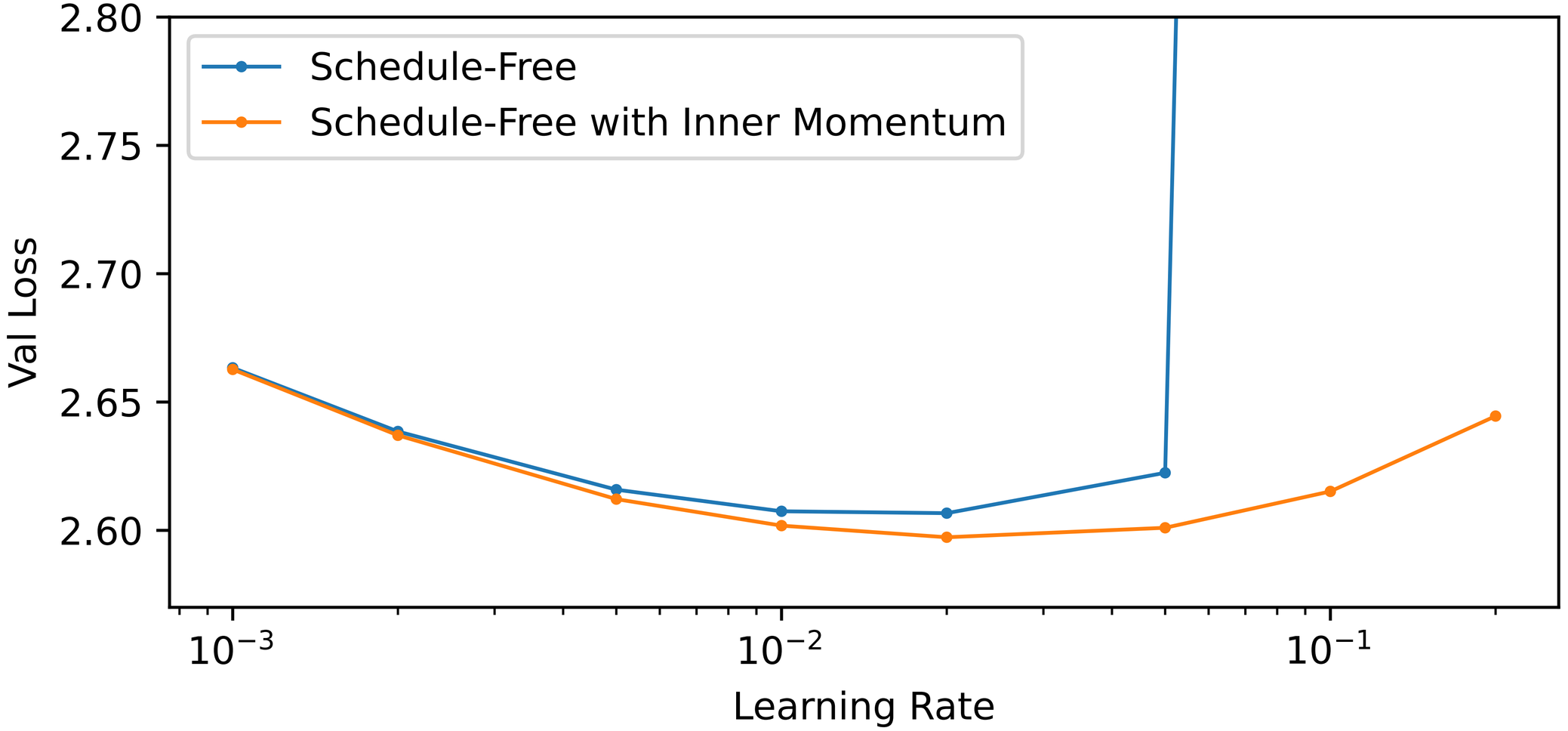
The Schedule-Free original formulation removed AdamW's momentum for simplicity. Turns out its CRITICAL to large batch training



# Why?

**MOMENTUM ENABLES LARGER LEARNING RATES**

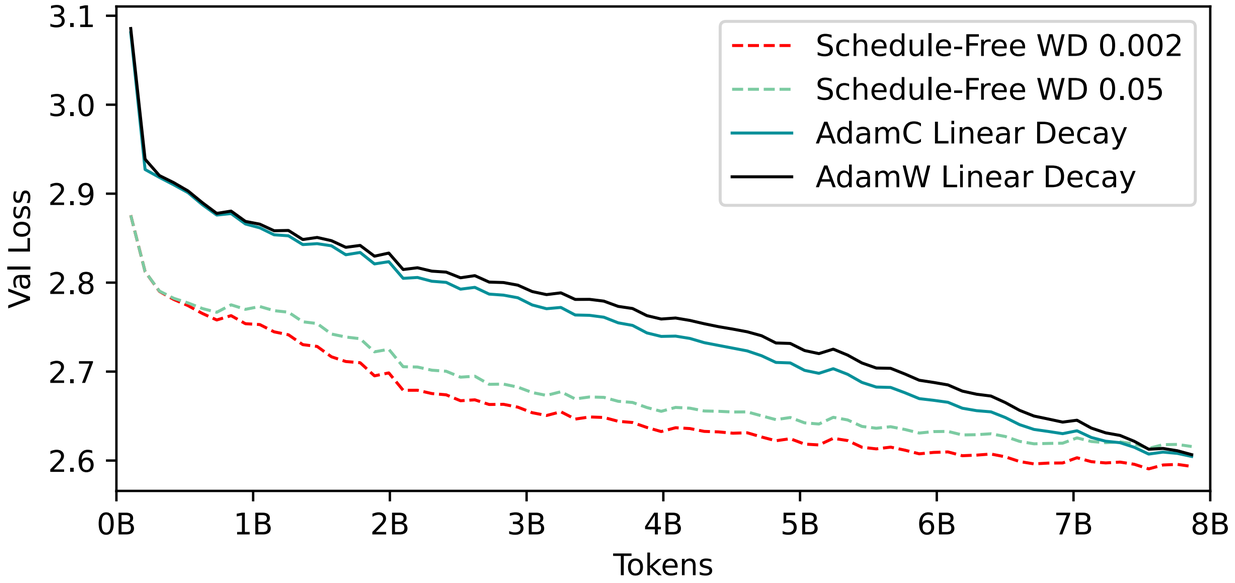


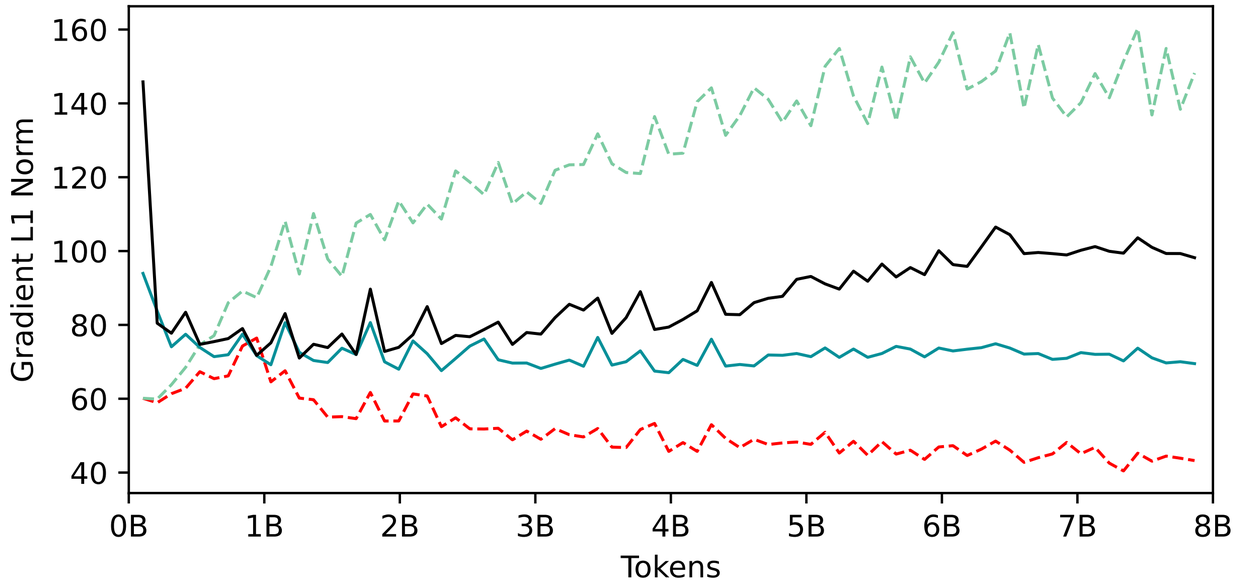

Learning rates must be scaled with batch-size (sqrt(B)) to maintain performance. Momentum appears to be critical to using large step sizes.

PART #2

# Understanding weight norms & gradient norms is **critical** for LLM training

Strange things happen to weight and gradient norms during training…..




Why does this matter?

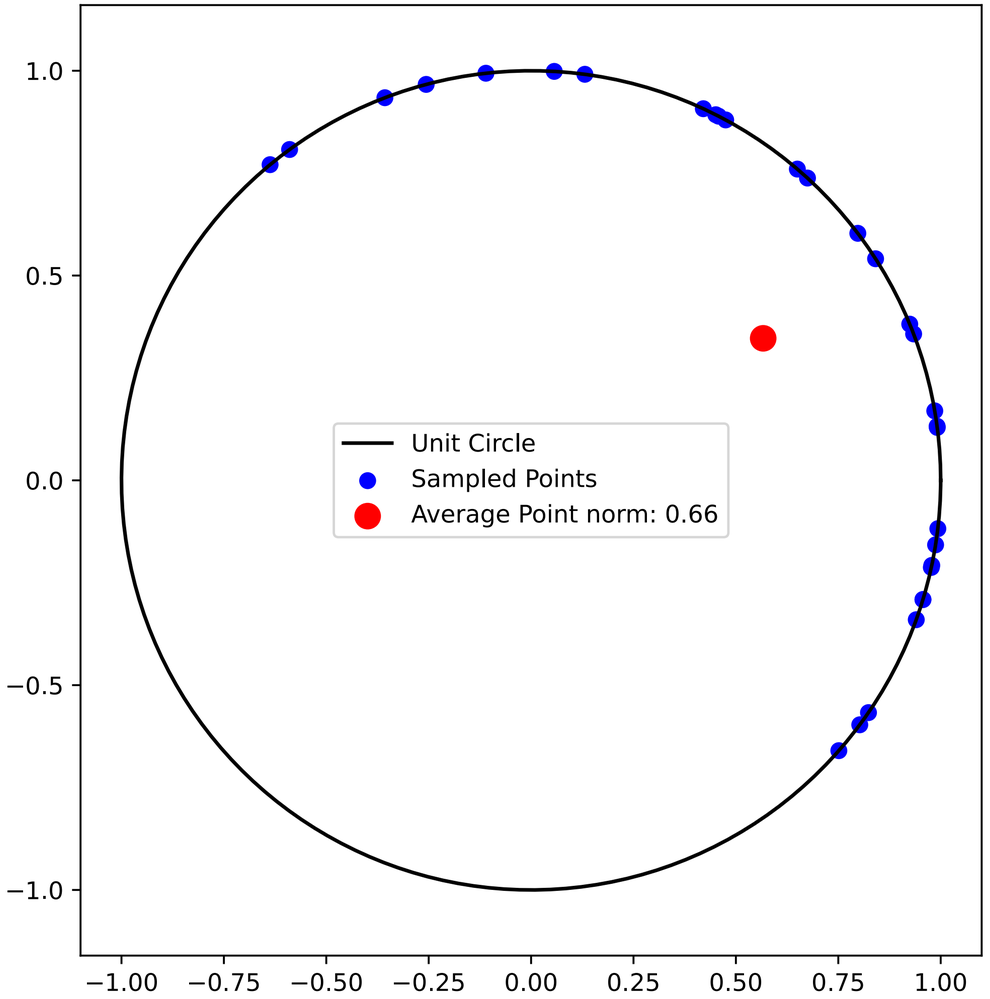
Gradient norm has no relation to convergence for LLM training, instead it controls the effective learning rate!

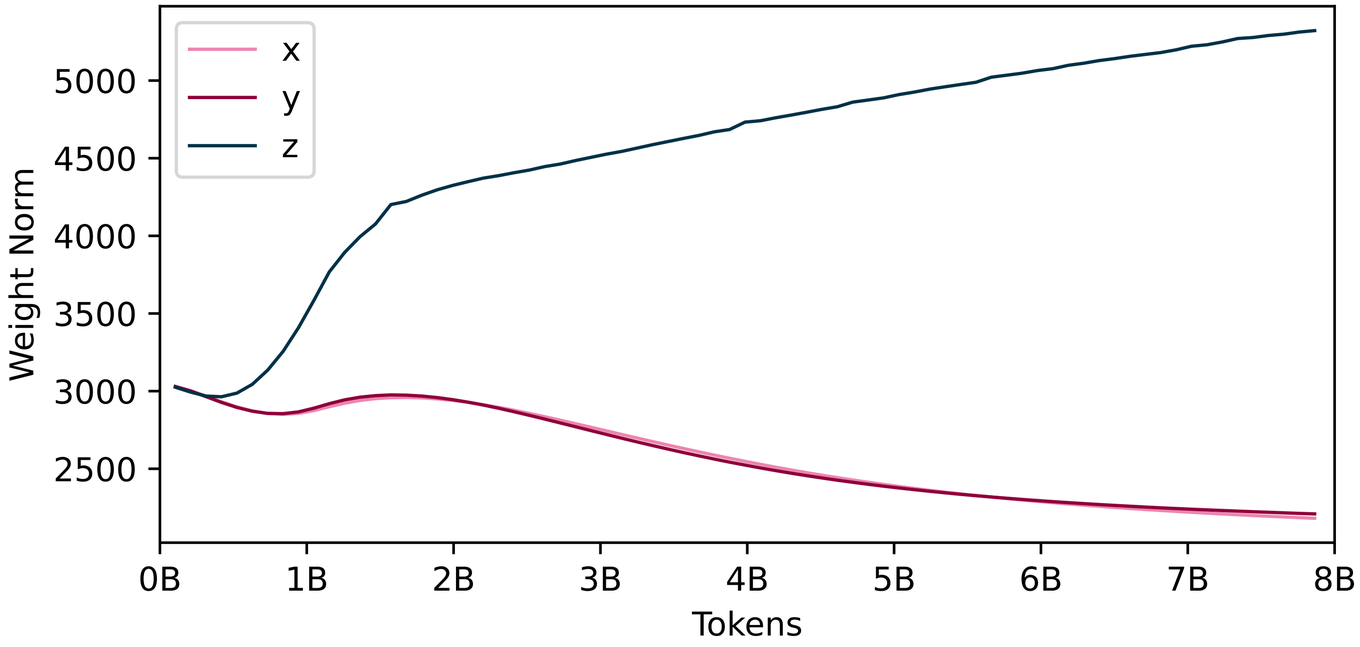
$$\eta_{\text{eff}} \propto \eta_t \left\| g_t \right\|^2$$

A flat gradient norm is ideal, it keeps the learning rate following expected behavior!

# Why does this happen?

Weight-Decay maintains weights on a ball. Averaging gives a point inside the ball with smaller weight norm. Gradient norm scales inversely with weight norm, so the gradient norm increases.



# FIX:

If the *effective* learning rate scales with gradient norm

$$\eta_{\text{eff}} \propto \eta_t \left\| g_t \right\|^2$$

Then just explicitly scale the learning rate by the gradient norm:

$$\gamma_t = \frac{?}{\left\| \nabla f(x_t) \right\|^2}.$$

Hmmm… doesn't that remind us of the Polyak Step Size?

$$\gamma_t = \frac{f(x_t) - f_*}{\left\| \nabla f(x_t) \right\|^2}.$$

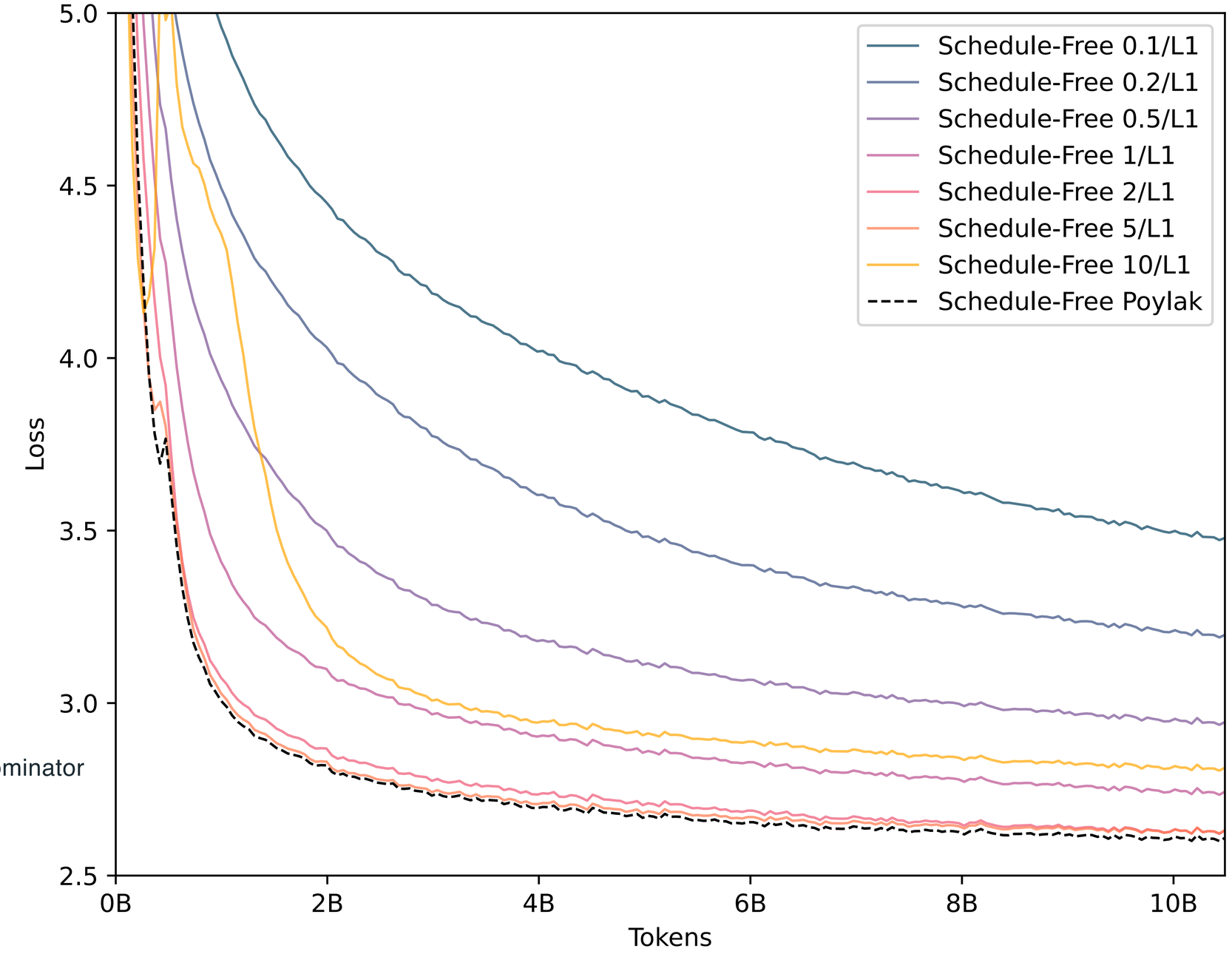
## POLYAK WORKS!

As good as any fixed numerator

Practical version of Polyak+SF:

$$\gamma_t = \frac{f(y_t) - f_* + \beta \langle \nabla f(y_t), z_{t-1} - x_t \rangle}{\sqrt{\frac{\pi}{2}} \|\nabla f(y_t)\|_1}$$

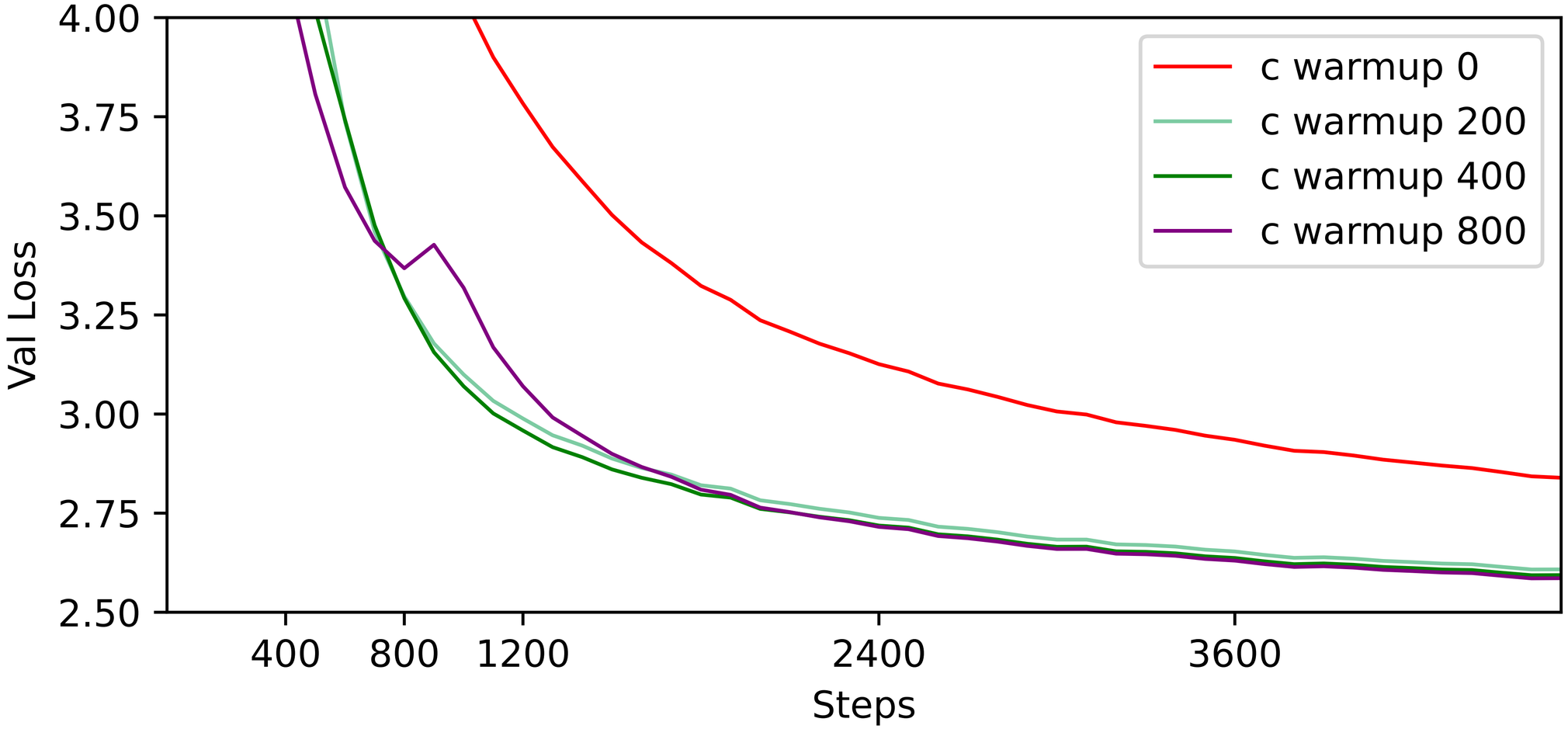
Pun et al. (2025) + a scaled L1 norm in the denominator



PART #3

# Warmup….

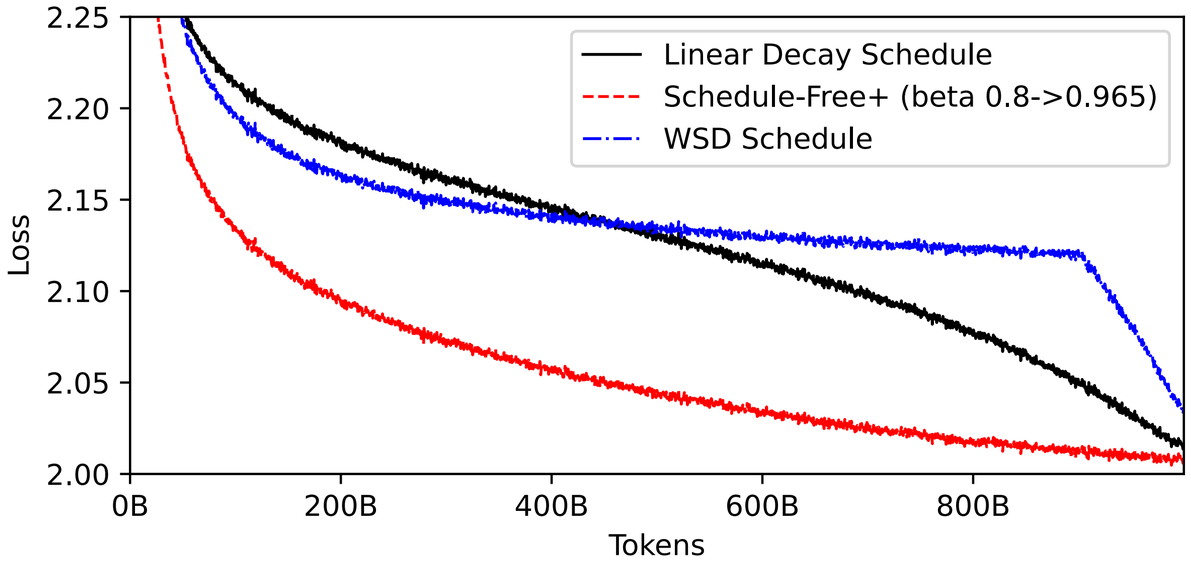
## WARMING UP THE AVERAGING HELPS A LOT

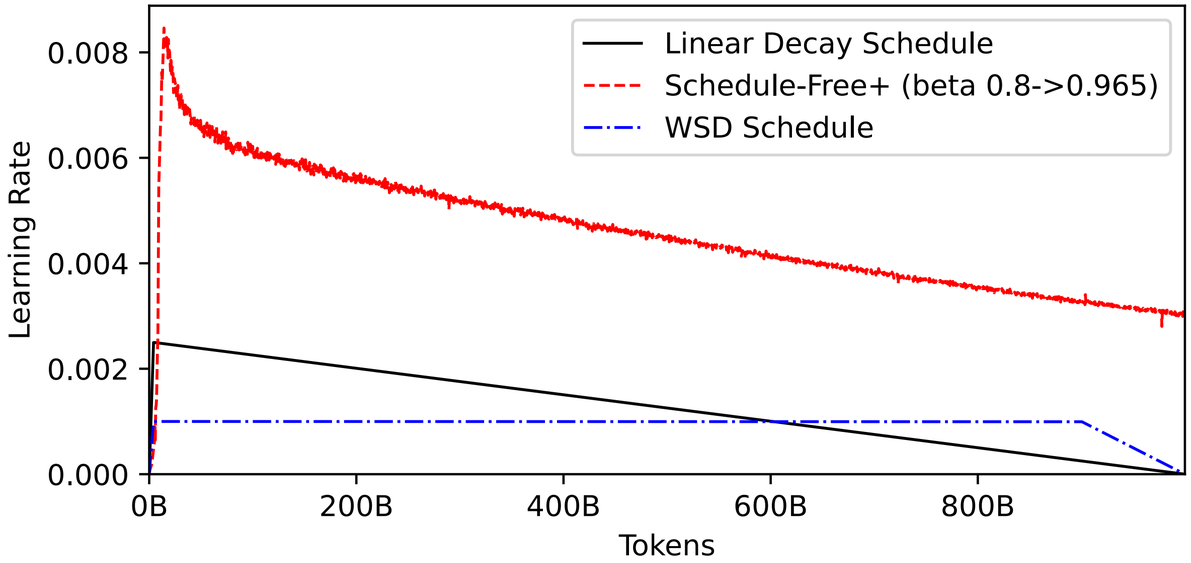



This is common wisdom, but appears more crucial for larger model training.

None of the 20+ experiments in the Schedule-Free paper required this…. But LLM training does.

# RESULTS: Very effective for long duration training runs!

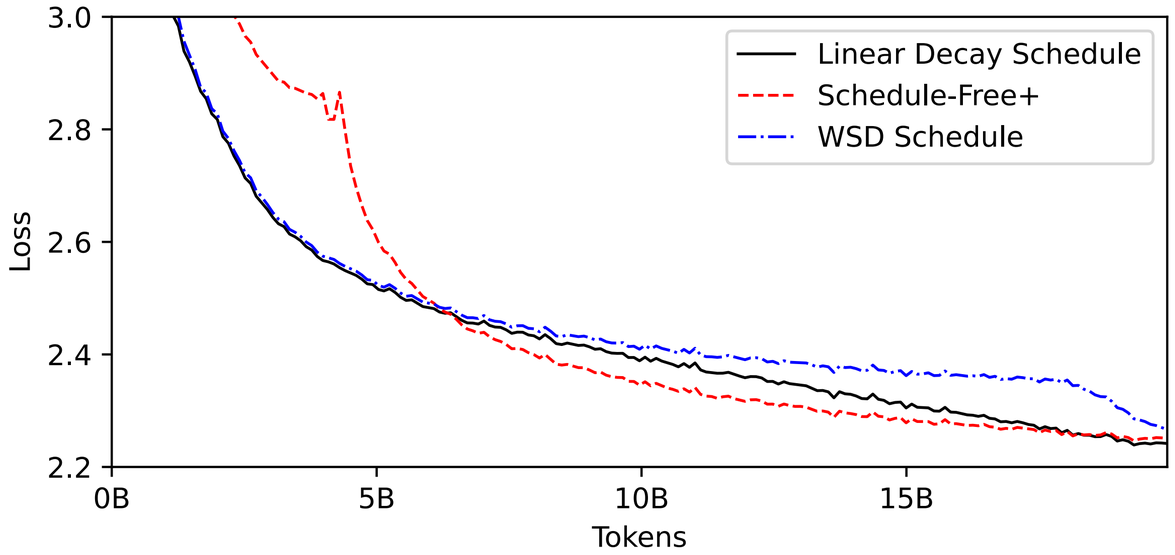


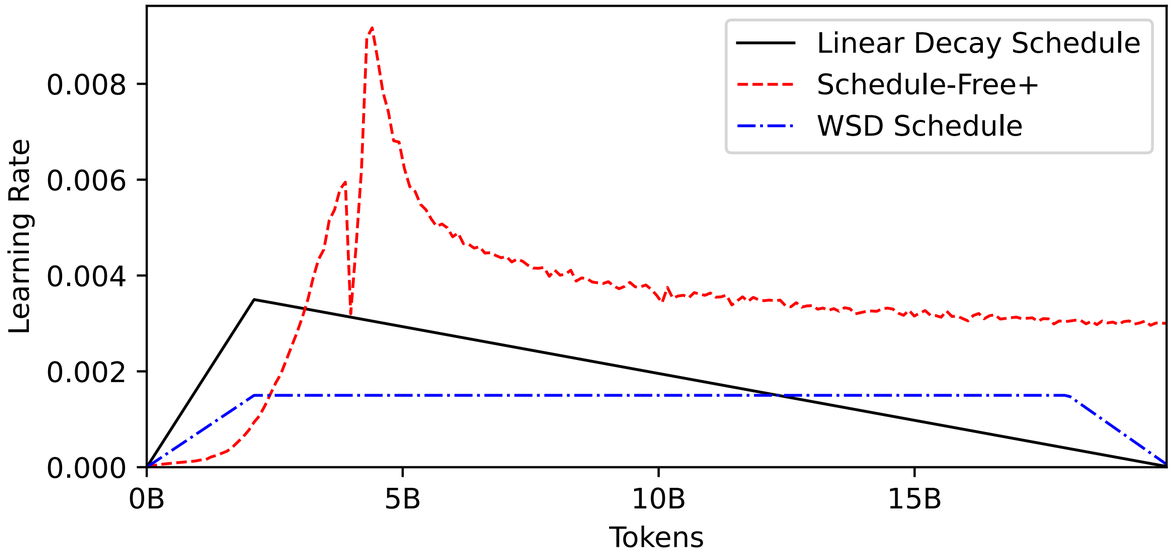




1B parameters, 1T tokens (1000 tokens-per-parameter)

Same training loss as a 45% longer training run!

# Failure cases: short training runs




When training the same 1B model for only 20B tokens, Schedule-Free has no advantage.

Speculation: Schedule-Free benefits from low loss it sees for intermediate iterates. Short runs don’t have significantly lower loss during the middle of training.

ALSO: Warmup takes a larger fraction of the run.

# Nice Properties of Schedule-Free

In classic theory, we expect the convergence of function value to follow a nice smooth 1/sqrt shape:

$$\mathbb{E}f(\bar{z}_t) - f_* \leq \frac{\|x_1 - x_*\| G}{\sqrt{t}}$$

Schedule-based approaches look nothing like this. Schedule-Free training curves do!

We fit the following curve $f(x_t) = \frac{a}{\sqrt{t+b}} + c$