

Hyperparameter Scaling Laws from First Principles

Optimization Reading Group at Mistral

Rustem Islamov
PhD Student, University of Basel

Introduction



Ioannis Mavrothalassitis
PhD student, EPFL



Valentyn Boreiko
Researcher, Amazon



Moritz Haas
Researcher, Amazon



Roman Machacek
PhD Student, University of Bern



Eduard Gorbunov
Assistant Professor, MBZUAI



Volkan Cevher
Professor, EPFL



Aurelien Lucchi
Associate Professor, University of Basel

Background

Tuning hyperparameters at scale is *infeasible* due to compute and time constraints. Therefore, we try to approximate them by

Background

Tuning hyperparameters at scale is *infeasible* due to compute and time constraints. Therefore, we try to approximate them by

- Training models of multiple sizes d on several token horizons T , and then fitting power laws for hyperparameters as a function of d and T

Background

Tuning hyperparameters at scale is **infeasible** due to compute and time constraints. Therefore, we try to approximate them by:

- Training models of multiple sizes d on several token horizons T , and then fitting power laws for hyperparameters as a function of d and T
Still **expensive**: deriving Step Law [Li et al., 2025] scaling law required training 3,700 models from scratch, consuming **one million** H800 GPU-hours

Background

Rule	Optimizer	Learning Rate	Batch Size	Weight Decay	Mom.	Model Scaling	Type
Bi et al. (2024) ^(a)	AdamW	$C^{-0.125}$	$C^{0.327}$	–	–	Tied to T	Em
Bjorck et al. (2025) ^(b)		$d^{-0.23}T^{-0.32}$	Const.	–	–	Separate	Em
Li et al. (2025a)		$d_{\text{ne}}^{-0.713}T^{0.307}$	$T^{0.571}$	–	–	Separate	Em
Bergsma et al. (2025) ^(c)		Const.	$T^{0.383}$	$d^{-0.527}T^{-0.09}$	–	Separate	Em
Mlodozieniec et al. (2026) ^(d)		$(B/T)^{1/2}$	Free	$(B/T)^{1/2}$	B/T	Separate	Th
Schaipp (2026) ^(e)		–	$T^{0.566}$	–	–	Separate	Em
Ajroldi et al. (2026)		$d^{-0.417}T^{0.086}$	$d^{0.145}T^{0.460}$	–	–	Separate	Em

^(a) One can also fit the compute exponents along the compute-optimal path of the same paper, where $T \propto C^{0.4757}$, obtaining $\eta \propto T^{-0.263}$ and $B \propto T^{0.688}$ at a fixed model size.

^(b) In this study, they arrived at the scaling law using a fixed batch size. The fit is restricted to models with at least 760M parameters.

^(c) Converted from the fitted timescale $B/(\eta\lambda T) \propto (T/d)^{-0.527}$ at a constant learning rate, which the authors report to be more efficient than adjusting the learning rate.

^(d) The scaling law for the batch size is not specified and treated as a user choice. The momentum column refers to $1 - \beta_{1,2}$, and width and depth have separate multipliers. Bethune et al. (2026) replace B/T by B/T^γ and fit $\gamma = 0.44$ on masked diffusion models, so that the learning rate and the momentum stay constant for $B \propto T^{0.44}$.

^(e) Fitted on the training runs released by Li et al. (2025a).

^(f) Stated in the number of iterations T/B for a static batch size. The authors also derive a batch-size schedule within a run.

^(g) Scaling laws derived for discrete diffusion language models. The learning-rate exponent combines $B \propto T^{0.8225}$ with $\eta \propto B^{0.3412}$.

Background

Tuning hyperparameters at scale is *infeasible* due to compute and time constraints. Therefore, we try to approximate them by:

- Training models of multiple sizes d on several token horizons T , and then fitting power laws for hyperparameters as a function of d and T
Still *expensive*: deriving Step Law [Li et al., 2025] scaling law required training 3,700 models from scratch, consuming H800 GPU-hours
- Stable feature learning through maximal-update parameterization (μ P) [Yang et al., 2021] and completeP [Dey et al., 2025]

Background

Tuning hyperparameters at scale is *infeasible* due to compute and time constraints. Therefore, we try to approximate them by:

- Training models of multiple sizes d on several token horizons T , and then fitting power laws for hyperparameters as a function of d and T
Still *expensive*: deriving Step Law [Li et al., 2025] scaling law required training 3,700 models from scratch, consuming H800 GPU-hours
- Stable feature learning through maximal-update parameterization (μ P) [Yang et al., 2021] and completeP [Dey et al., 2025]

Typically requires taking a width to infinity at a *fixed* training horizon. It does not tell how to choose batch size.

Background

Tuning hyperparameters at scale is *infeasible* due to compute and time constraints. Therefore, we try to approximate them by:

- Using SDE approximations [Malladi et al., 2022; Compagnoni et al., 2025; Mlodozeneic et al., 2026] to derive scaling laws to preserve the trajectory or the asymptotic loss

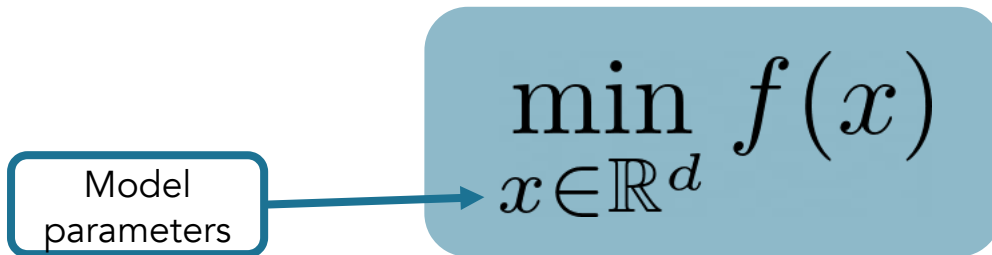
Background

Tuning hyperparameters at scale is *infeasible* due to compute and time constraints. Therefore, we try to approximate them by:

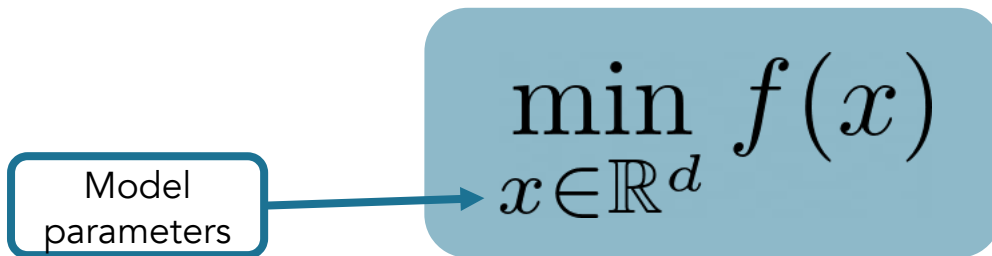
- Using SDE approximations [Malladi et al., 2022; Compagnoni et al., 2025; Mlodozeneic et al., 2026] to derive scaling laws to preserve the trajectory or the asymptotic loss
- Studying convergence of algorithms and using the upper bounds to scale the hyperparameters

Caveat: there is some uncertainty since the optimization bounds depend on the assumptions that one uses

Problem Formulation



Problem Formulation



Assumptions.

1.

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^d$$

Smoothness

Problem Formulation

Model parameters

$$\min_{x \in \mathbb{R}^d} f(x)$$

Assumptions.

1. $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^d$

Smoothness

2.

$$\|\nabla f(x)\|_2 \geq \mu(f(x) - f^*), \quad \forall x \in \mathcal{X}$$

μ -KL

Iterates stay in the set during training

Problem Formulation

Model parameters

$$\min_{x \in \mathbb{R}^d} f(x)$$

Assumptions.

1. $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^d$

Smoothness

2. $\|\nabla f(x)\|_2 \geq \mu(f(x) - f^*), \quad \forall x \in \mathcal{X}$

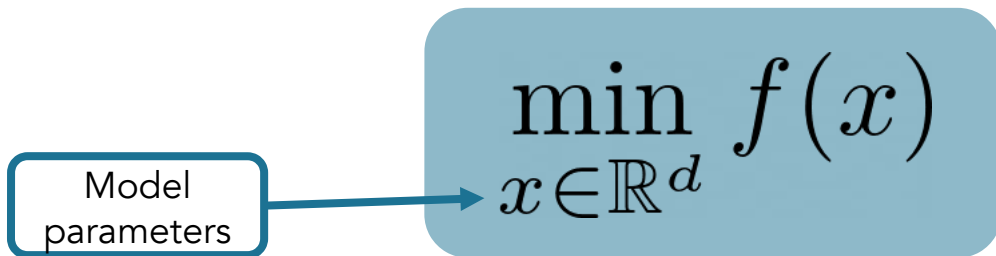
μ -KL

3. $\mathbb{E}_\xi[g(x, \xi)] = \nabla f(x), \quad \mathbb{E}_\xi[\|g(x, \xi) - \nabla f(x)\|_2^2] \leq \sigma_\star^2/B$

Bounded
Variance

Decays inversely
with batch size

Problem Formulation



Assumptions.

1. $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^d$
2. $\|\nabla f(x)\|_2 \geq \mu(f(x) - f^*), \quad \forall x \in \mathcal{X}$
3. $\mathbb{E}_\xi[g(x, \xi)] = \nabla f(x), \quad \mathbb{E}_\xi[\|g(x, \xi) - \nabla f(x)\|_2^2] \leq \sigma_\star^2/B$
4. $g_k := g(x_k, \xi_k), \quad |g_k^i| \leq G$

Smoothness

μ -KL

Bounded
Variance

Bounded
Gradient

Problem Formulation

Consider a model with training loss f satisfying Assumptions, and let $\mathcal{A}$ be an optimizer with hyperparameters θ (e.g., batch size, learning rate, etc). We write $x_{\mathcal{A}}(\theta, T)$ for the output of $\mathcal{A}$ after processing T .

Problem Formulation

Consider a model with training loss f satisfying Assumptions, and let $\mathcal{A}$ be an optimizer with hyperparameters θ (e.g., batch size, learning rate, etc). We write $x_{\mathcal{A}}(\theta, T)$ for the output of $\mathcal{A}$ after processing T .

$$\theta_{\mathcal{A}}^*(T) := \arg \min_{\theta} \mathbb{E}_{\xi} [f(x_{\mathcal{A}}(\theta, T)) - f^*]$$

Problem Formulation

$$\theta_{\mathcal{A}}^*(T) := \arg \min_{\theta} \mathbb{E}_{\xi} [f(x_{\mathcal{A}}(\theta, T)) - f^*]$$

Using optimization bounds, we approximate the solution by

$$\min_{\theta} \mathbb{E}_{\xi} [f(x_{\mathcal{A}}(\theta, T)) - f^*] \leq \varepsilon_{\mathcal{A}}(T, \theta^{\text{theory}})$$

Upper bound from
convergence analysis

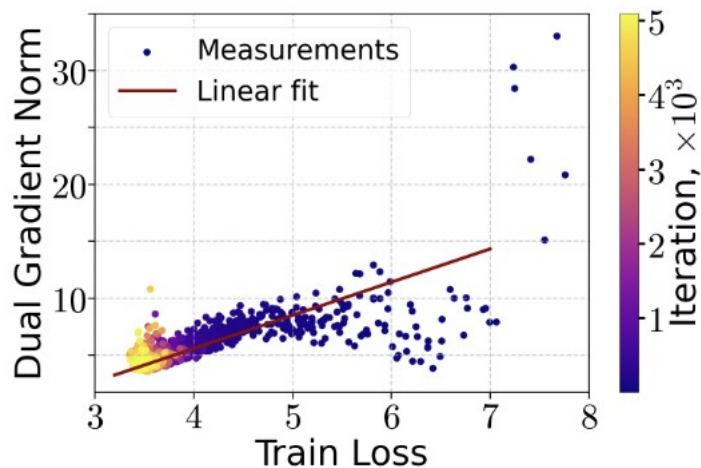
Hyperparameters
suggested by the analysis

BST Scaling Rule

This approach heavily relies *on the assumptions* used to derive the bounds.

BST Scaling Rule

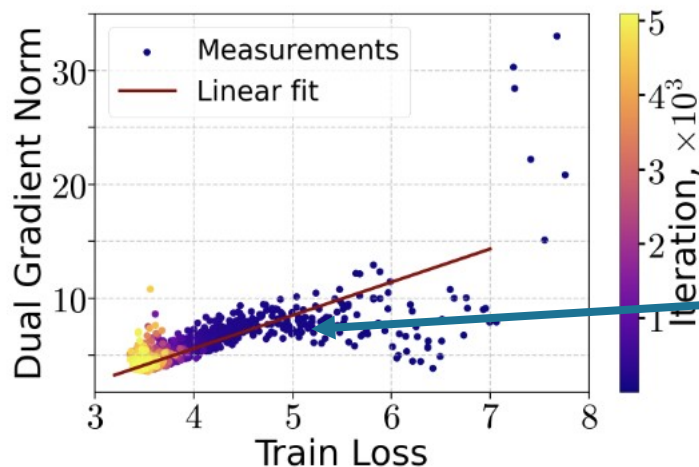
This approach heavily relies *on the assumptions* used to derive the bounds.



124M NanoGPT model on FineWeb.
[Islamov et al., 2026]

BST Scaling Rule

This approach heavily relies *on the assumptions* used to derive the bounds.

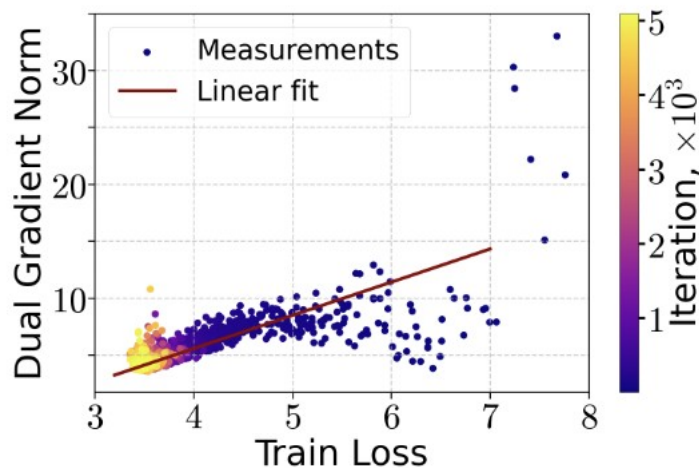


$$\|\nabla f(x)\|_2 \geq \mu(f(x) - f^*), \quad \forall x \in \mathcal{X}$$

124M NanoGPT model on FineWeb.
[Islamov et al., 2026]

BST Scaling Rule

This approach heavily relies on *the assumptions* used to derive the bounds.



124M NanoGPT model on FineWeb.
[Islamov et al., 2026]

1. Two-layer linear network trained by gradient flow with MSE loss
2. Linear readout with bounded features and readout
3. Polyak-Lojasiewicz (PL) condition might fail for simple neural networks without over-parametrization

BST Scaling Rule

This approach heavily relies on the assumptions used to derive the bounds.
the tightness of the analysis.

BST Scaling Rule

This approach heavily relies on the assumptions used to derive the bounds.
the tightness of the analysis.

In our analysis, we derive the convergence upper bounds that *match lower bounds* with respect to problem constants and convergence speed of stochastic and deterministic terms.

If you want $\mathbb{E}_\xi[f(\hat{x}_K) - f^*] \leq \varepsilon$, you need at least $K = \Omega\left(\frac{L}{\varepsilon} + \frac{L\sigma^2}{\varepsilon^3}\right)$

Convergence Analysis for AdamW

We consider AdamW with decoupled weight decay [Loshchilov and Hutter, 2019]

$$m_k = (1 - \beta_1)m_{k-1} + \beta_1 g_k$$

$$v_k = (1 - \beta_2)v_{k-1} + \beta_2(g_k \odot g_k)$$

$$x_{k+1} = (1 - \eta\lambda)x_k - \eta \frac{m_k}{\sqrt{v_k} + \nu}$$

Convergence Analysis for AdamW

Theorem. There is a choice of hyperparameters for AdamW, such that large enough token horizon T , the optimization error is

$$\varepsilon_{\text{AdamW}}(\theta, T) \approx \max \left\{ \frac{B}{T}, \frac{1}{T^{1/3}} \right\}$$

Convergence Analysis for AdamW

Theorem. There is a choice of hyperparameters for AdamW, such that large enough token horizon T , the optimization error is

$$\varepsilon_{\text{AdamW}}(\theta, T) \approx \max \left\{ \frac{B}{T}, \frac{1}{T^{1/3}} \right\}$$

Given tuned hyperparameters for the base model, BST scaling rule for AdamW prescribes to change hyperparameters as

$$B \sim T^{2/3}, \quad \eta, \beta_1 \sim \text{Const}, \quad \lambda \sim \frac{1}{\sqrt{d}T^{1/3}}$$

to decay the optimization error as $\varepsilon \sim T^{-1/3}$

BST Scaling Law

Rule	Optimizer	Learning Rate	Batch Size	Weight Decay	Mom.	Model Scaling	Type
Bi et al. (2024) ^(a)	AdamW	$C^{-0.125}$	$C^{0.327}$	–	–	Tied to T	Em
Bjorck et al. (2025) ^(b)		$d^{-0.23}T^{-0.32}$	Const.	–	–	Separate	Em
Li et al. (2025a)		$d_{\text{ne}}^{-0.713}T^{0.307}$	$T^{0.571}$	–	–	Separate	Em
Bergsma et al. (2025) ^(c)		Const.	$T^{0.383}$	$d^{-0.527}T^{-0.09}$	–	Separate	Em
Mlodozeniec et al. (2026) ^(d)		$(B/T)^{1/2}$	Free	$(B/T)^{1/2}$	B/T	Separate	Th
Schaipp (2026) ^(e)		–	$T^{0.566}$	–	–	Separate	Em
Ajroldi et al. (2026)		$d^{-0.417}T^{0.086}$	$d^{0.145}T^{0.460}$	–	–	Separate	Em

Given tuned hyperparameters for the base model, BST scaling rule for AdamW prescribes to change hyperparameters as

$$B \sim T^{2/3}, \quad \eta, \beta_1 \sim \text{Const}, \quad \lambda \sim \frac{1}{\sqrt{d}T^{1/3}}$$

to decay the optimization error as $\varepsilon \sim T^{-1/3}$

BST Scaling Law

Rule	Optimizer	Learning Rate	Batch Size	Weight Decay	Mom.	Model Scaling	Type
Bi et al. (2024) ^(a)		$C^{-0.125}$	$C^{0.327}$	–	–	Tied to T	Em
Bjorck et al. (2025) ^(b)		$d^{-0.23}T^{-0.32}$	Const.	–	–	Separate	Em
Li et al. (2025a)		$d^{-0.713}T^{0.307}$	$T^{0.571}$			Separate	Em
Bergsma et al. (2025) ^(c)	AdamW	Const.	$T^{0.383}$	$d^{-0.527}T^{-0.09}$	–	Separate	Em
Moudozenec et al. (2026) ^(d)		$(B/T)^{1/3}$	Rec	$(B/T)^{1/3}$	B/T	Separate	Th
Schaipp (2026) ^(e)		–	$T^{0.566}$	–	–	Separate	Em
Ajroldi et al. (2026)		$d^{-0.417}T^{0.086}$	$d^{0.145}T^{0.460}$	–	–	Separate	Em

Given tuned hyperparameters for the base model, BST scaling rule for AdamW prescribes to change hyperparameters as

$$B \sim T^{2/3}, \quad \eta, \beta_1 \sim \text{Const}, \quad \lambda \sim \frac{1}{\sqrt{d}T^{1/3}}$$

to decay the optimization error as $\varepsilon \sim T^{-1/3}$

Experiments on NanoGPT

We tune AdamW hyperparameters for 124M model on 1.3B tokens, then we transfer to larger token horizons using BST rule.

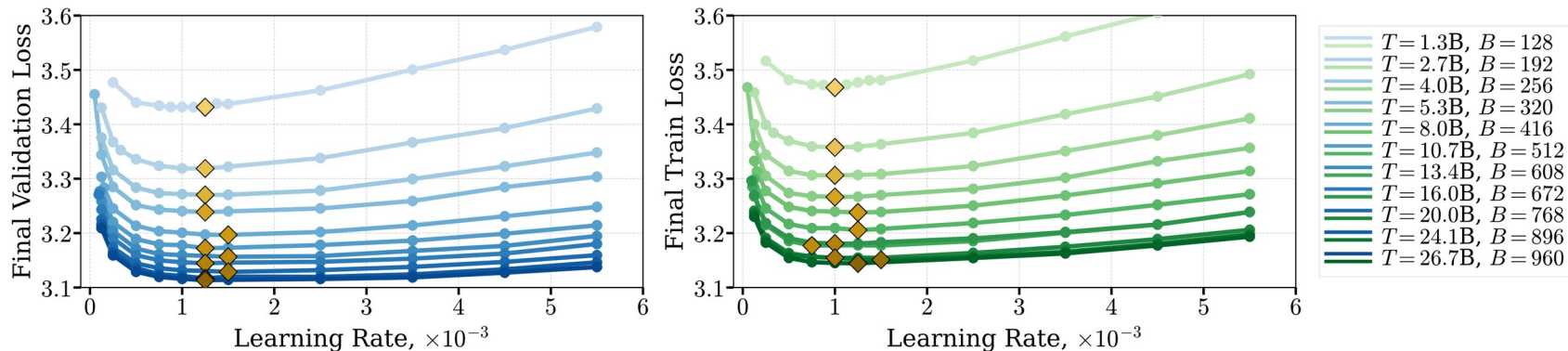


Figure: 124M NanoGPT model trained with AdamW, whose hyperparameters are tuned on 1.3B tokens and transferred to larger token horizons.

Experiments on NanoGPT

We tune AdamW hyperparameters for 124M model on 1.3B tokens, then we transfer to larger token horizons using BST rule.

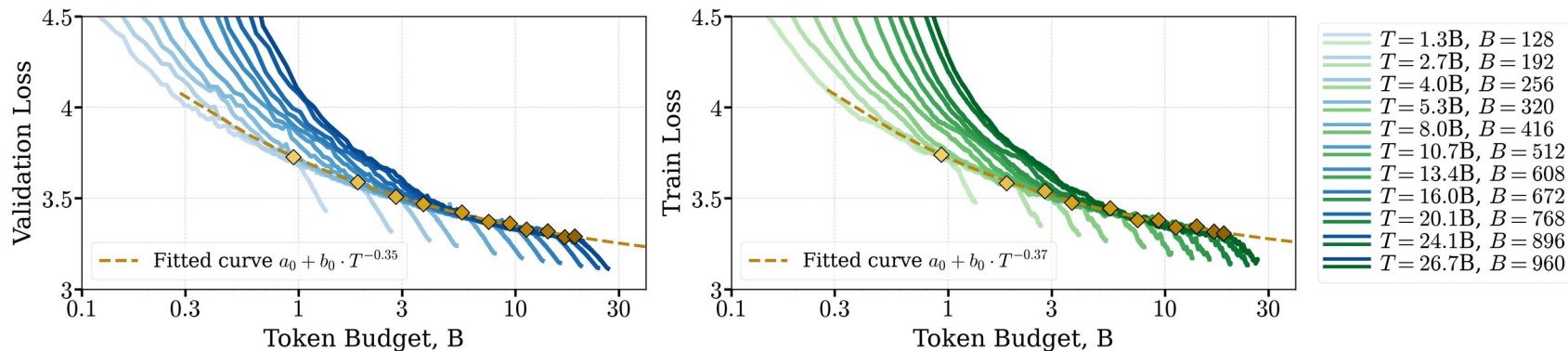
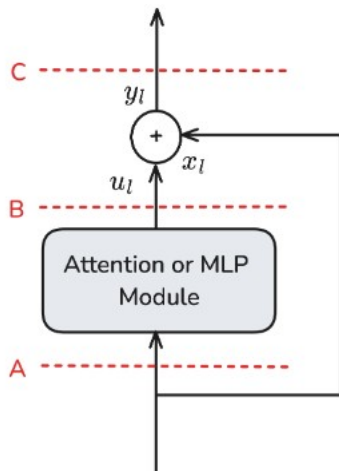


Figure: 124M NanoGPT model trained with AdamW, whose hyperparameters are tuned on 1.3B tokens and transferred to larger token horizons.

Experiments on Peri-LN Models

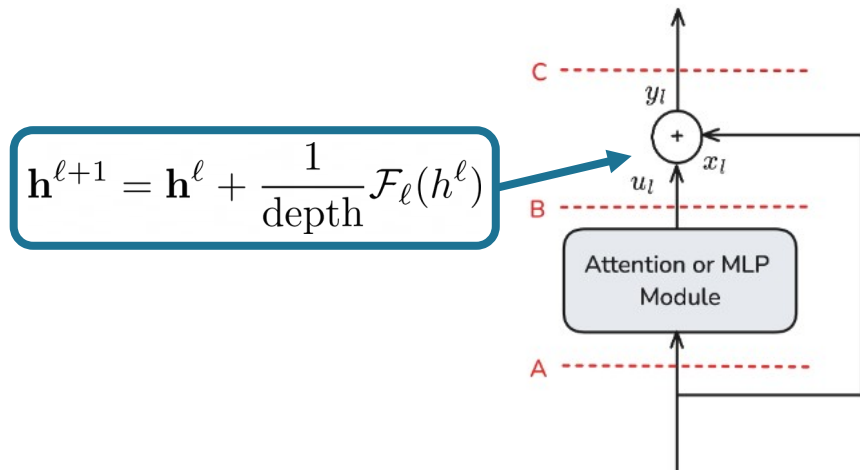
For model and token horizon co-scaling experiments, we use Peri-LN model [Kim et al., 2025], which adds an additional normalization layer.



	A	B	C
Post-LN	×	×	✓
Pre-LN	✓	×	×
Peri-LN	✓	✓	×

Experiments on Peri-LN Models

For model and token horizon co-scaling experiments, we use Peri-LN model [Kim et al., 2025], which adds an additional normalization layer.



	A	B	C
Post-LN	×	×	✓
Pre-LN	✓	×	×
Peri-LN	✓	✓	×

+ CompleteP [Dey et al., 2025] model scaling

Experiments on Peri-LN Models

For model and token horizon co-scaling experiments, we use Peri-LN model [Kim et al., 2025], which adds an additional normalization layer.

We tune AdamW hyperparameters for the base 162M model on 1.6B tokens, and test the performance for several model sizes.

Experiments on Peri-LN Models

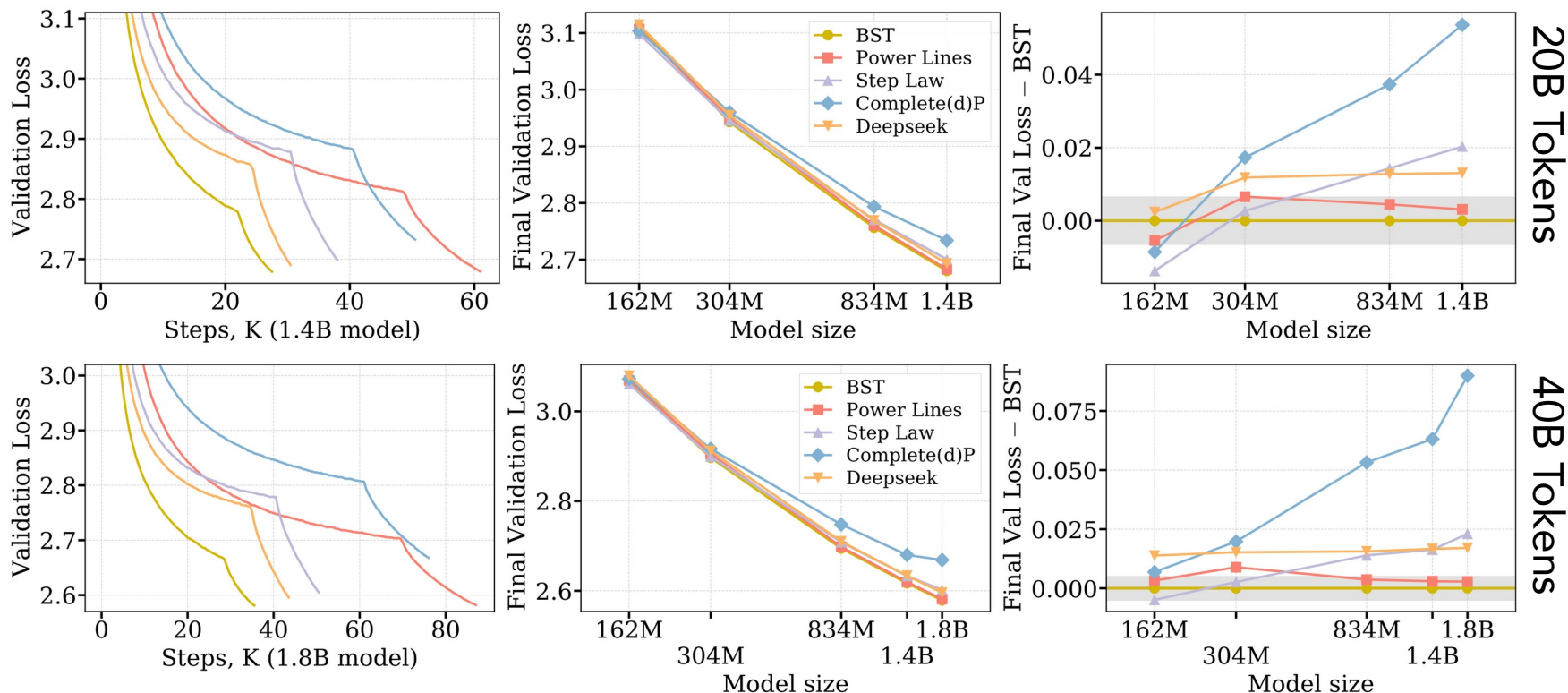


Figure: Training of Peri-LN models at 20B and 40B token horizons

Convergence Analysis for Scion/Muon

We consider Scion (Muon with decoupled weight decay) [Pethick et al., 2025]

$$m_k = (1 - \beta_1)m_{k-1} + \beta_1 g_k$$

$$d_k = \arg \min_d \langle d, m_k \rangle \quad \text{s.t.} \quad \|d\| \leq 1$$

Orthogonalization
(done via Newton-
Schulz steps)



$$x_{k+1} = (1 - \lambda)x_k + \lambda\eta d_k$$

For Euclidean norm: $d_k = -\frac{g_k}{\|g_k\|_2}$

For spectral norm: $d_k = -UV^\top, \quad g_k = U\Sigma V^\top$

Convergence Analysis for Scion/Muon

Theorem. There is a choice of hyperparameters for Scion, such that the optimization error is

$$\varepsilon_{\text{Scion}}(\theta, T) \approx \max \left\{ \frac{B}{T}, \frac{1}{T^{1/3}}, \frac{1}{B^{1/6}T^{1/3}} \right\}$$

Given tuned hyperparameters for the base model, BST scaling rule for AdamW prescribes to change hyperparameters as

$$B \sim T^{2/3}, \quad \lambda \sim T^{-1/3}, \quad \beta_1, \eta \sim \text{Const}$$

to decay the optimization error as $\varepsilon \sim T^{-1/3}$

Experiments on NanoGPT

We tune Scion hyperparameters for 124M model on 1.3B tokens, then we transfer to larger token horizons using BST rule.

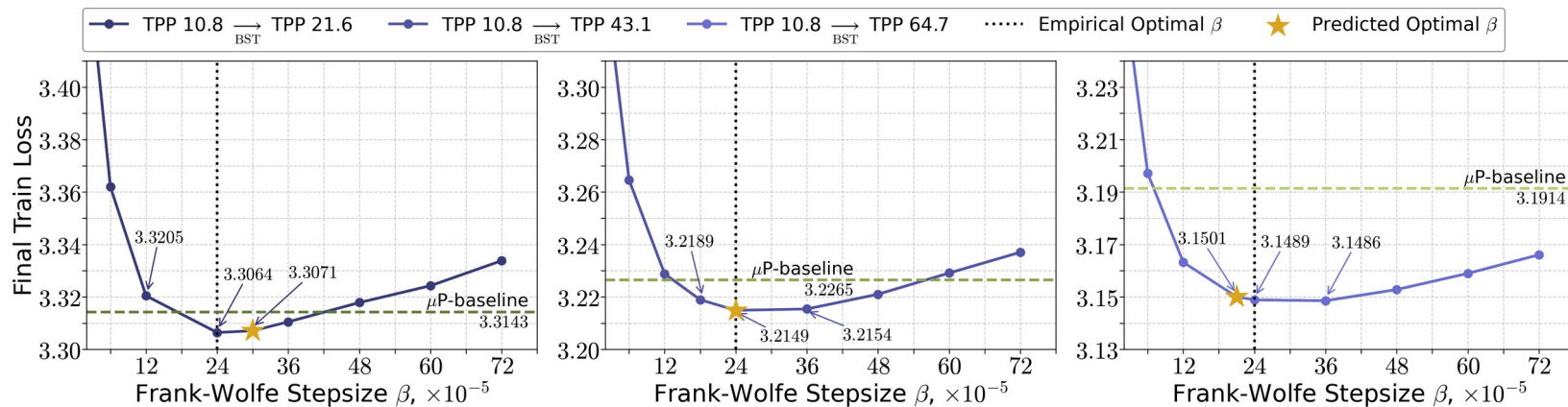


Figure: Training of 124M NanoGPT model at different TPP

Frank-Wolfe
stepsize=Weight Decay

Experiments on NanoGPT

We tune Scion hyperparameters for 124M model on 1.3B tokens, then we transfer to larger token horizons using BST rule.

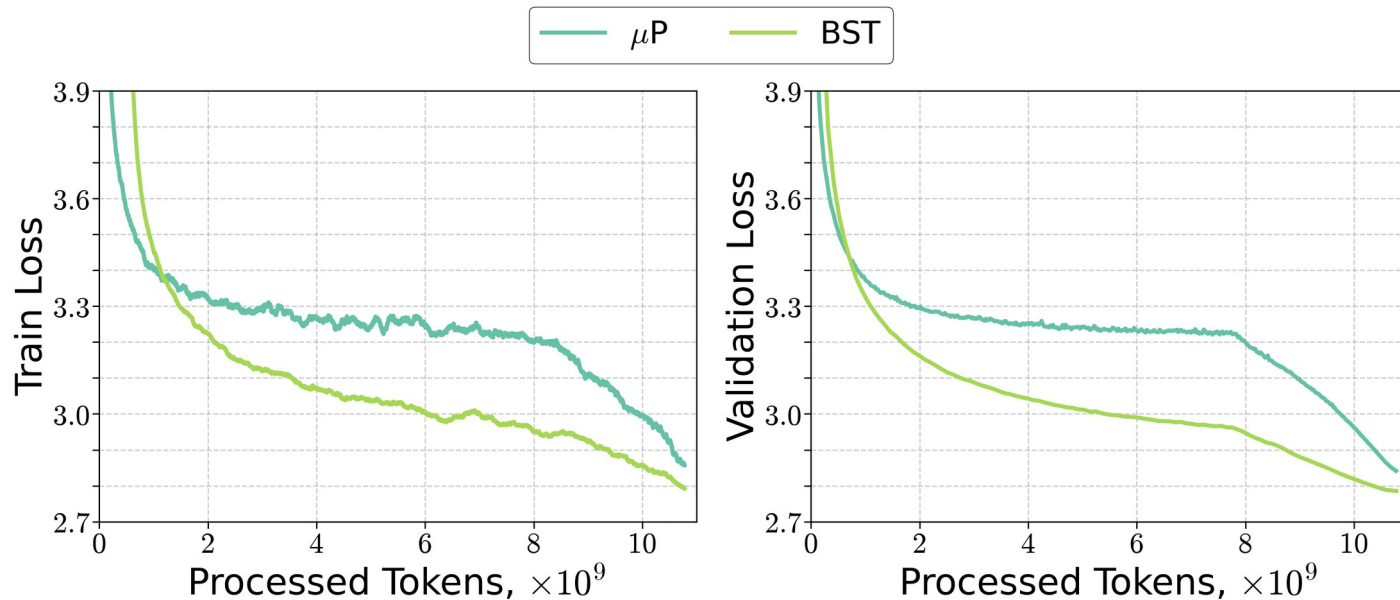


Figure: Training of 1B NanoGPT model at 10
TPP

Experiments on NanoGPT

We tune Scion hyperparameters for 124M model on 1.3B tokens, then we transfer to larger token horizons using BST rule.

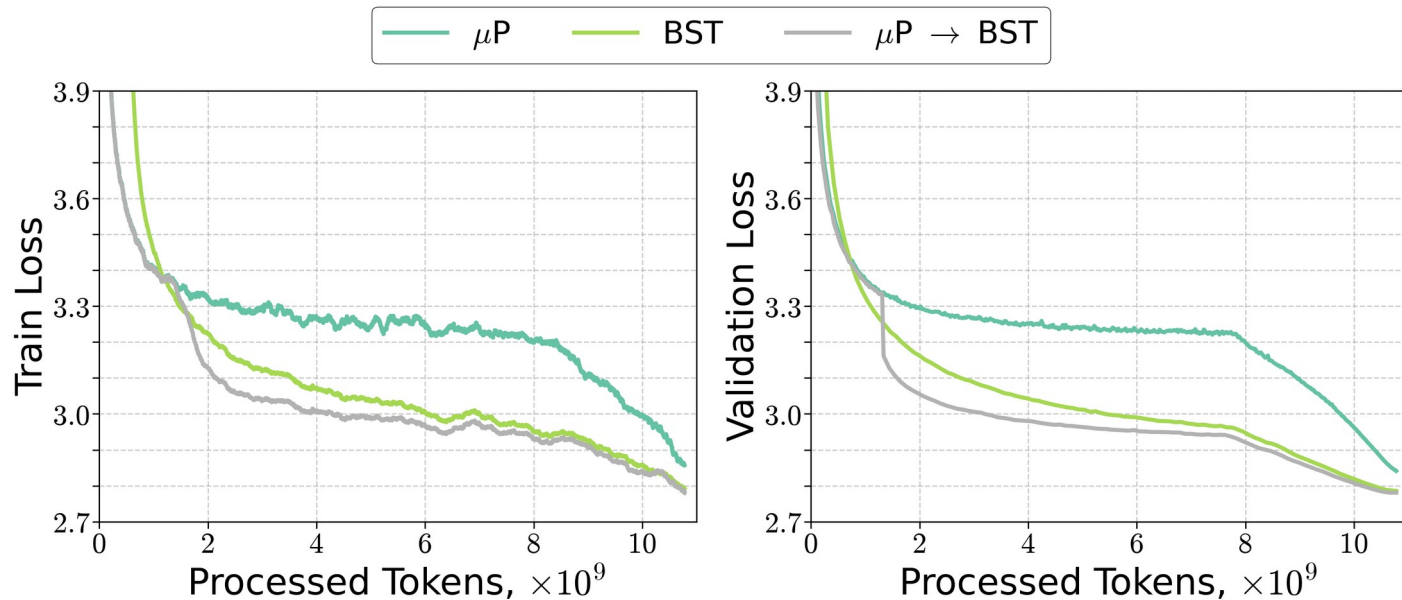


Figure: Training of 1B NanoGPT model at 10 TPP

Conclusion

How does the learning rate schedule affect the results? Which schedule should we use?

Does this approach extend to other architectures: MoE, Looped Transformers?

Does this approach extend to other settings: Distillation, QAT?

Conclusion

How does the learning rate schedule affect the results? Which schedule should we use?

Does this approach extend to other architectures: MoE, Looped Transformers?

Does this approach extend to other settings: Distillation, multimodal data?

What about compute efficiency? How to incorporate it into this setting?

Thank you!
Questions? Comments?

I am on a job market next year
Looking for internships and positions