On the Role of Batch Size in Stochastic Conditional Gradient Methods

Rustem Islamov

University of Basel

Based on: R. Islamov, R. Machacek, A. Lucchi, A. Silveti-Falls, E. Gorbunov, V. Cevher, On the Role of Batch Size in Stochastic Conditional Gradient Methods, ICML 2026

Why Batch Size Matters

The Setting

Large-scale LLM training is constrained by a token budget T rather than optimization steps
With batch size B and sequence length S, the connection is $K = T/(BS)$

What We Know

- Increasing batch size B: improves hardware utilization [Goyal et al., 2017] and reduces variance per step
- Beyond critical threshold: benefits saturate, performance degrades, hurts generalization

Key Question: How should (B, S) and stepsize be chosen under fixed T?

Background and Related Work

Works on Batch Size

- ***Critical batch size*** derivations are based on auxiliary function (variance of the stochastic gradient) [McCandlish et al., 2018, Shallue et al., 2019]
- Critical batch sizes ***scale primarily with the effective data size*** and only weakly with model size under a fixed token budget [Zhang et al., 2025, Bergsma et al., 2025]
- Increase batch size stage-wise [Goyal et al., 2017]
- Batch size scheduling: instead of decaying LR ***increase batch size*** [Smith et al., 2018]

Hyperparameter Transfer

- μP framework: appropriate parameterization and initialization keeps gradient magnitudes $\Theta(1)$ ***across model widths*** [Yang & Hu, 2020, [Yang et al., 2022]
- CompleteP [Dey et al., 2025] extends μP framework across changes in ***model depth***
- Derivations require a ***fixed noise level*** (batch size)

Stochastic Conditional Gradient Methods

Algorithm 1 Stochastic Conditional Gradient (SCG) [Pethick et al., 2025]

Input: $x_0, m_0 \in \mathcal{X}$, parameters $\alpha, \beta \in (0,1), \eta > 0$
for $k = 0, \ldots, K-1$ **do**
  sample $\xi_k \sim \mathcal{D}$
  compute $m_{k+1} = (1-\alpha)m_k + \alpha g(x_k; \xi_k)$
  compute $d_{k+1} = \arg\min_{d \in \mathcal{X}} \langle m_{k+1}, d \rangle$ s.t. $\|d\| \leq 1$
  compute $x_{k+1} = (1-\beta)x_k + \beta\eta d_{k+1}$
end for

- Explicitly controls the weights through Frank-Wolfe step
- Covers modern spectral optimizers such as Muon and Scion [Jordan et al., 2024, Pethick et al., 2025]

Problem Formulation

$$\min_{x \in \mathcal{X}} f(x)$$

Euclidean norm: $\|x\|_2$ **Arbitrary norm:** $\|x\|$

Associated dual norm: $\|x\|_* := \min_{\|x'\| \leq 1} \langle x, x' \rangle$

A1 **L-Smoothness:** $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$

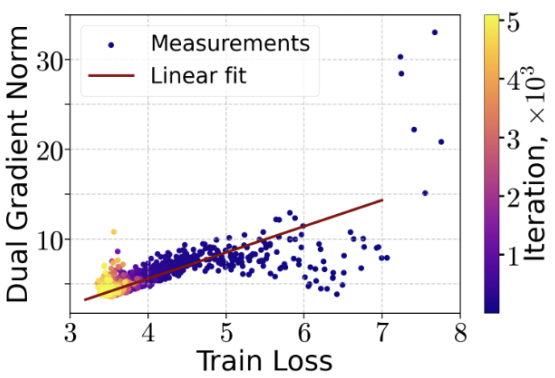
A2 **Norm Equivalence:** $\|x\| \leq \rho \|x\|_2$

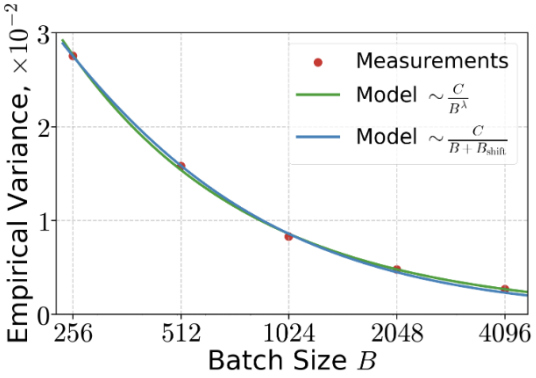
A3 ***μ*-Kurdyka-Lojasiewicz:** $\|\nabla f(x)\|_* \geq \mu(f(x) - f^\star)$ [Bolte et al., 2007], [Karimi et al., 2016]

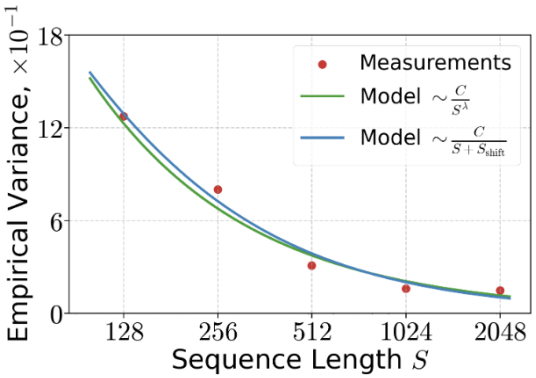
A4 **Bounded Variance:** $\mathbb{E}_\xi[g(x;\xi)] = \nabla f(x)$ and $\mathbb{E}_\xi[\|g(x;\xi) - \nabla f(x)\|_2^2] \leq \sigma^2$

where the variance scales as $\sigma^2 = \frac{\sigma_0^2}{BS}$

Empirical Verification of Assumptions







Empirical verification of Assumptions (A3) and (A4) during the training of 124M NanoGPT model on FineWeb [Penedo et al., 2024] with Scion optimizer under the experimental setup of [Pethick et al., 2025] with batch size 512, sequence length 1024, Frank-Wolfe stepsize 3.6e-4, under token budget 2.7B.

(**Left**): the measurements (dual gradient norm vs train loss) fit a linear function when the loss is below 5 with a slope μ.

(**Central and Right**): measurements (empirical variance vs batch size/sequence length a power low well. For the central figure: $\lambda \approx 0.9, B_{\text{shift}} \approx 90$; for the right figure: $\lambda \approx 1.1, S_{\text{shift}} \approx 35$

Main Result: Convergence Guarantees

Main Convergence Theorem

Let Assumptions (A1)-(A4) hold. Let $m_0 = g(x_0, \xi_0)$. Let the parameters of SCG and initialization x_0 are chosen such that

$$\beta = \mathcal{O}(1/K), \qquad \eta = \tilde{\mathcal{O}}(1/\mu), \qquad \alpha = \min\left\{1, \mathcal{O}\left(\frac{(\varepsilon\mu)^2}{(\rho\sigma)^2}\right)\right\}, \qquad 2\|x_0\| \leq \eta,$$

$$K = \max\left\{\tilde{\mathcal{O}}(1), \tilde{\mathcal{O}}\left(\frac{L}{\mu^2\varepsilon}, \frac{\rho\sigma}{\mu\varepsilon}, \frac{L(\rho\sigma)^2}{\mu^4\varepsilon^3}, \frac{(\rho\sigma)^3}{(\mu\varepsilon)^3}\right)\right\}$$

Then, the output of SCG satisfies $\mathbb{E}[f(x_K) - f^\star] \leq \varepsilon$

In practice, the training is conducted under a fixed token budget T, so we translate the requirement on K to

$$T = \max\left\{\tilde{\mathcal{O}}(BS), \tilde{\mathcal{O}}\left(\frac{LBS}{\mu^2\varepsilon}, \frac{\rho\sigma_0\sqrt{BS}}{\mu\varepsilon}, \frac{L(\rho\sigma_0)^2}{\mu^4\varepsilon^3}, \frac{(\rho\sigma)^3}{(\mu\varepsilon)^3\sqrt{BS}}\right)\right\}$$

BST Error and Scaling Rule

Under a fixed token budget T, the lowest achievable optimization error is

$$\varepsilon = \max\left\{\frac{LBS}{\mu^2 T}, \left(\frac{L\rho^2\sigma_0^2}{\mu^4 T}\right)^{1/3}, \frac{\rho\sigma_0}{\mu(T^2BS)^{1/3}}\right\}$$

This rule suggests the existence of the largest useful batch size:

- Below the threshold noise dominates
- Above the threshold the training is iteration-starved
- Critical batch [McCandlish et al., 2018] is where regimes switch

BST Scaling Rules

By balancing the first (optimization error) and second (noise error) terms, we derive BST scaling rule

$$BS \propto T^{2/3}, \qquad \beta \propto \frac{1}{K}, \qquad \alpha = \text{Const}$$

Comparison Against Prior Works

- BST scaling rule is derived **directly from optimization bounds on the loss**, not auxiliary quantity (e.g., gradient noise variance line in [McCandlish et al., 2018])
- μP framework [Yand and Hu, 2020] **does not determine the batch size**: it transfers hyperparameters given a training regime, but in practice you need to know the critical batch size of the larger model to transfer.
- BST scaling rule is not a curve fit, that can be architecture dependent [McLeish et al. 2025]

BST Scaling Rules

By balancing the first (optimization error) and second (noise error) terms, we derive BST scaling rule

$$BS \propto T^{2/3}, \qquad \beta \propto \frac{1}{K}, \qquad \alpha = \text{Const}$$

HP Transfer Strategies

- Assume that we have tuned batch size $B_0^\star$, sequence length $S_0^\star$, Frank-Wolfe stepsize $\beta_0^\star$, and momentum α parameters for a small model of size D_0.
- For a larger model of size D_1, we balance the deterministic (first) and stochastic (second) terms, and use batch size B_1, sequence length S_1, Frank-Wolfe stepsize β_1, and momentum α such that

$$B_1 S_1 = B_0^\star S_0^\star \left(\frac{T_1}{T_0} \frac{\mu_1}{\mu_0} \frac{\rho_1}{\rho_0} \frac{L_0}{L_1} \right)^{2/3}, \quad \beta_1 = \beta_0^\star \left(\frac{\sqrt{T_0}}{\sqrt{T_1}} \frac{\mu_1}{\mu_0} \frac{\rho_1}{\rho_0} \frac{L_0}{L_1} \right)^{2/3}$$

BST Scaling Rules

By balancing the first (optimization error) and second (noise error) terms, we derive BST scaling rule

$$BS \propto T^{2/3}, \qquad \beta \propto \frac{1}{K}, \qquad \alpha = \text{Const}$$

Estimating Problem-Dependent Constants

The BST scaling rule requires estimations of problem-dependent constants L, μ, ρ.

- To estimate the smoothness constant, we measure $\frac{\|g(x_k;\xi_k) - g(x_{k-1};\xi_{k-1})\|_*}{\|x_k - x_{k-1}\|}$
- To estimate μ KL constant, we fit a linear function into pairs of $\|g(x_k;\xi_k)\|_*, f(x_k;\xi_k)$
- To estimate the norm equivalence constant ρ, we track the ratio $\frac{\|g(x_k;\xi_k) - g(x_k;\Xi_k)\|_*}{\|g(x_k;\xi_k) - g(x_k;\Xi_k)\|_2}$

Then, we fit power laws when changing the number of layers p, embedding size w, and batch size B

$$\mu(p,w) = 5.2(p+1.7)^{-0.2} \qquad L(p,w) = 0.4(p+0.7)^{0.2}(w+126)^{0.35}$$

$$\rho(p,w,B) = 4.1(p-2.7)^{0.25}(w-250.8)^{0.3}(B-9.4)^{0.1}$$

Experiments

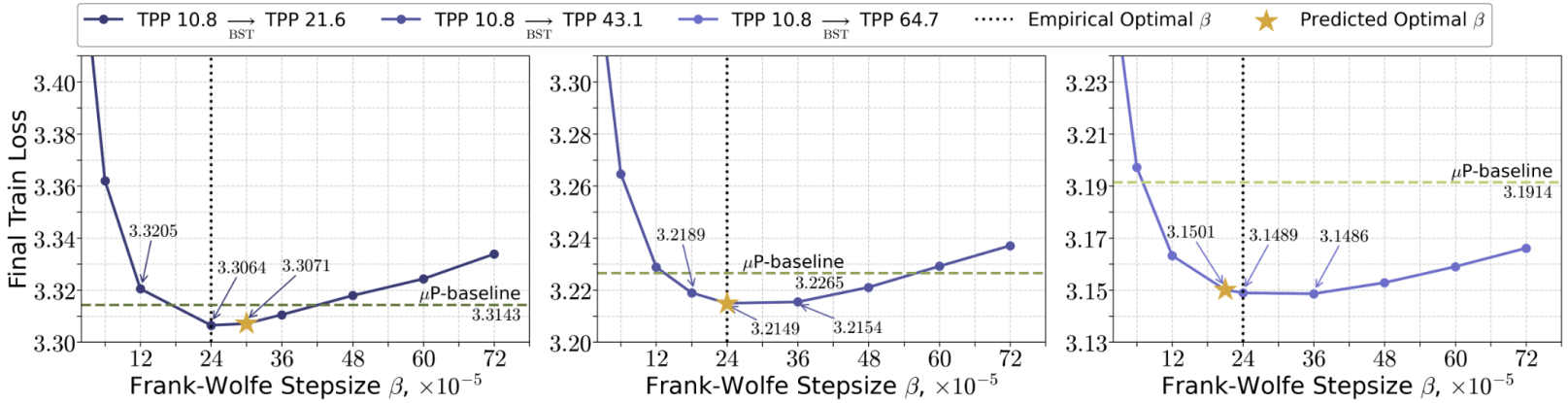


Figure 5: The final performance of the 124M model when varying the Frank–Wolfe stepsize β under different token budgets (**left:** 2.7B, **center:** 5.3B, **right:** 8.0B). We average the train loss over 3 random seeds and report the moving average in the window of size 500. We observe that the BST scaling rule predicts a good estimate for the optimal β when increasing the token budget. Moreover, the difference in performance between BST and μP baselines grows with a token budget.

Experiments

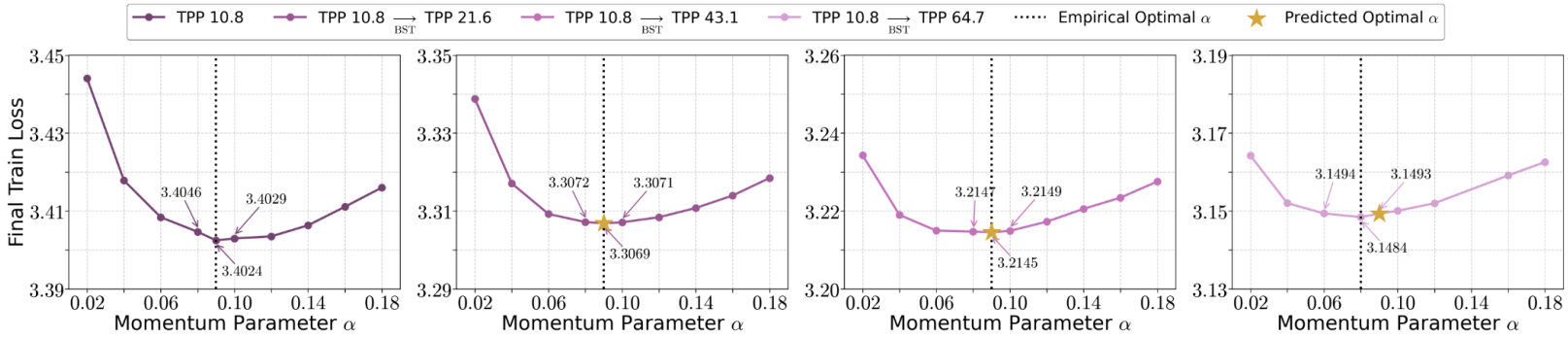


Figure 6: The final performance of the 124M model when varying the momentum α under different token budgets (**left:** 1.3B, **center left:** 2.7B, **center right:** 5.3B, **right:** 8.0B). We average the train loss over 3 random seeds and report the moving average in the window of size 500. We observe that rule momentum parameter α transfers under BST scaling.

Experiments

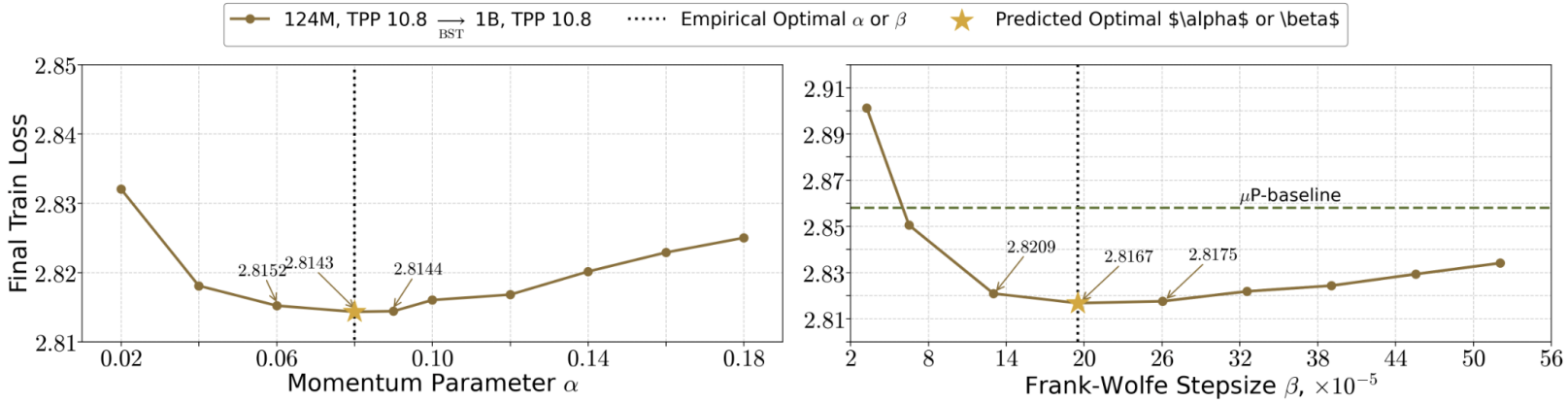


Figure 7: The final performance of the 1B model when varying the Frank–Wolfe stepsize β (**left**) and momentum parameter (**right**) under different token budget 10.6 TPP. We report the final train loss, smoothed in the window of size 500. We observe that the BST scaling rule predicts a good estimate for both optimal α and β when transferring from a smaller 124M model to a larger 1B model.

Experiments

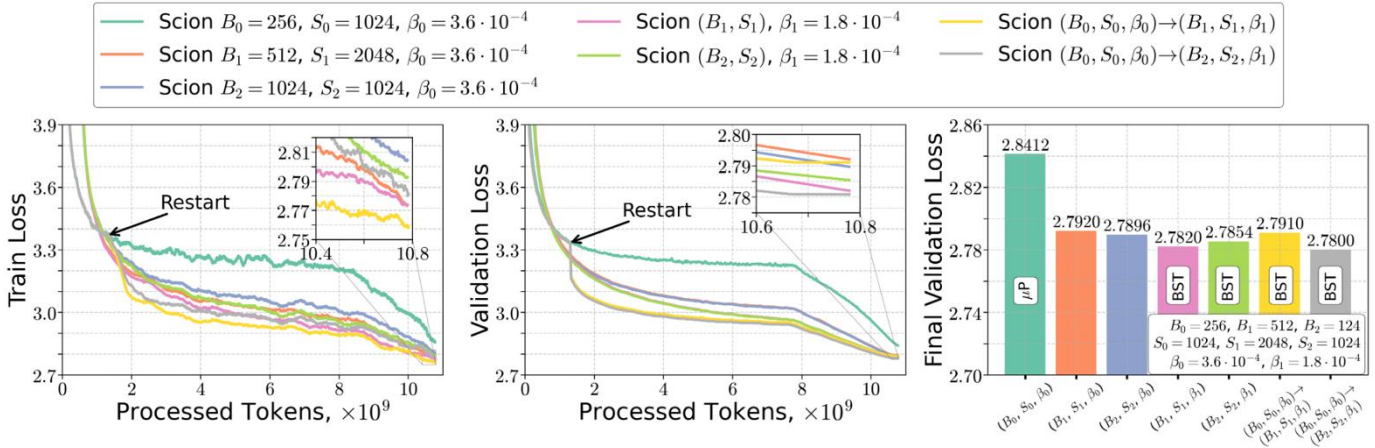


Figure 3: Comparison of batch size and sequence length scheduling strategies when training a 1B model. The restarting schemes (in yellow and gray) are compared against fixed schedules. The validation loss is evaluated with a smaller sequence length of 1024. The values of batch sizes $B_{0,1,2}$, sequence lengths $S_{0,1}$, and Frank–Wolfe stepsizes $\beta_{0,1}$ are given in the legends. The notation $(B_{0,1,2}, S_{0,1,2}, \beta_{0,1})$ characterizes which batch size, sequence length, and Frank–Wolfe stepsize are used for the particular setup. The notation $(B_0, S_0, \beta_0) \to (B_{1,2}, S_{1,2}, \beta_1)$ characterizes how parameters of Scion change after restart (e.g., batch size increases from B_0 to $B_{1,2}$), respectively. The notation μP or BST indicates the rule used to select B, S, and β.

Experiments

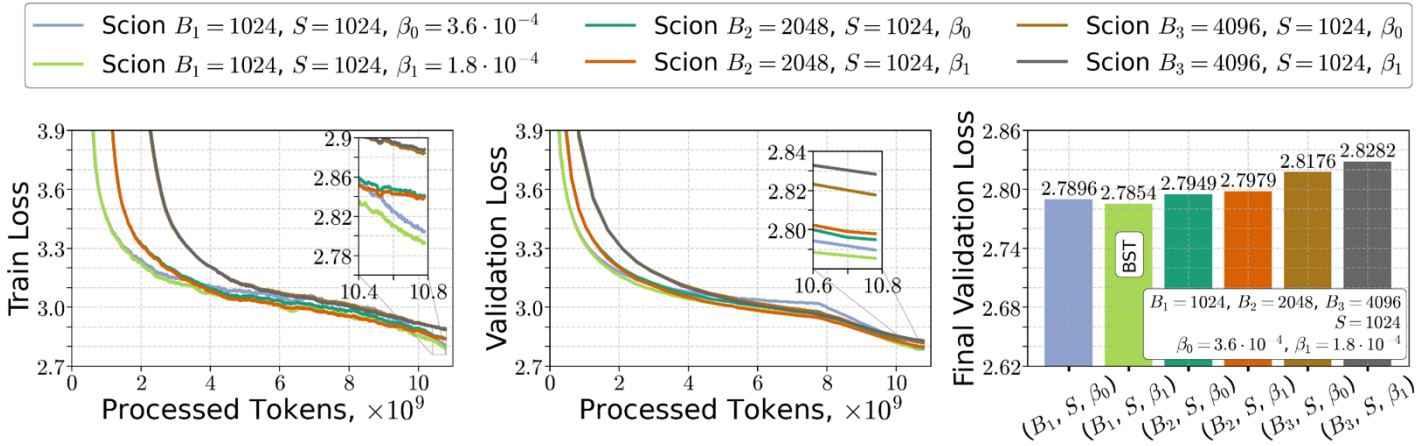


Figure 4: Comparison of fixed large batch size strategies when training a 1B model. The validation loss is evaluated with a smaller sequence length 1024. Scion with a batch size of 1024 suggested by our BST scaling rule achieves the best performance compared to other baselines with batch sizes 2048 and 4096. The values of batch sizes $B_{1,2,3}$, sequence lengths S, and Frank–Wolfe stepsizes $\beta_{0,1}$ are given in the legends. The notation $(B_{1,2,3}, S, \beta_{0,1})$ characterizes which batch size, sequence length, and Frank–Wolfe stepsize are used for the particular setup, respectively. The notation BST indicates the rule used to select the B, S, and β.

Conclusion

1. **Batch size should be treated as a dynamic optimization variable**, not a fixed hyperparameter. Optimal batch size depends on the token budget.

2. The BST scaling rule **balances noise reduction versus iteration count**.

3. Large batch sizes do not inherently hurt performance. When paired with proper stepsize scaling**, large-batch training is efficient**.

4. **There is a trade-off between batch size and sequence length**. Increasing sequence length improves model capability, but naïve context extension can reduce training efficiency. Therefore, capability (through sequence length) and efficiency (through batch size) must be co-designed, not tuned independently.

Thank You

Questions?

Definition of the Norms

In our work, individual and product norms follow the setup of [Pethick et al., 2025]

$$\|W\| = \max_{1 \le \ell \le p} \|W_\ell\|_{(\ell)}, \quad \|W\|_* = \sum_{\ell=1}^{p} \|W_\ell\|_{(*,\ell)}$$

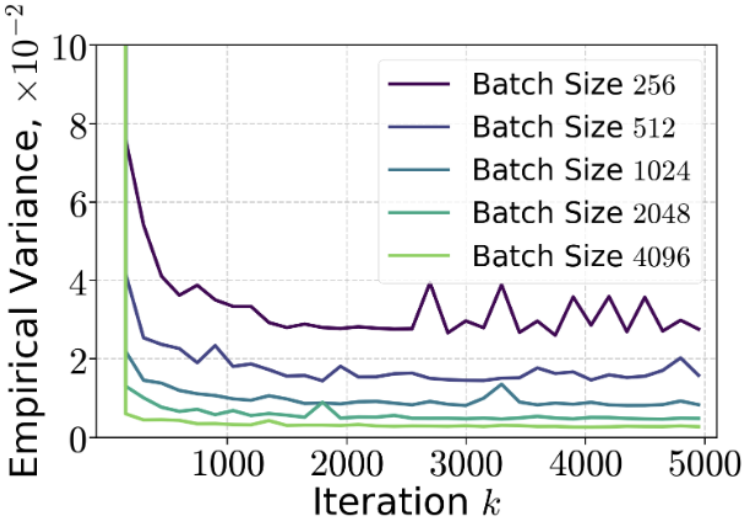
For the embedding and head layers, we use

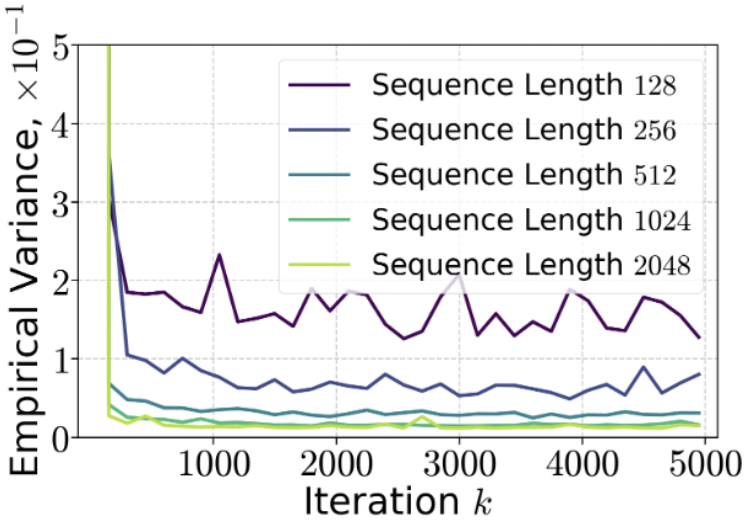
$$\|W\|_{(\ell)} = \max_{i,j} |W_{ij}|$$

For the rest of the layers, we use

$$\|W\|_{(\ell)} = \sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}} \|W_\ell\|_{\mathcal{S}_\infty}, \quad W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$$

Appendix: Dynamics of Empirical Variance





Estimations of Problem Constants

Table B.2: Estimated L constant for various model configurations.

n_embd \ n_layer	3	6	9	12	15	18	21	24	27	30
384	–	5.3	6.2	6.5	6.2	18	7.6	–	27	7.84
576	–	6.4	7.7	7.0	7.4	–	7.5	8.5	7.9	8.2
768	6.1	6.5	6.9	7.6	8.3	18	9.9	8.5	8.7	10.0
1152	–	–	–	8.8	9.4	10.8	9.5	–	–	–
1536	3	7.9	9.2	12	9.9	–	–	–	–	–
2304	–	9.9	–	–	–	–	–	–	–	–

Table B.3: Estimated ρ constant for various model configurations.

n_embd \ n_layer	3	6	9	12	15	18	21	24	27	30
384	–	35.5	41.3	48.2	48.7	–	50.7	–	–	58.0
576	–	42.1	53.8	61.1	62.9	–	64.2	64.6	66.2	68.6
768	31.2	52.6	64.1	67.1	72.4	–	80.2	87.0	86.3	89.2
1152	–	–	–	76.6	81.1	87.3	—	–	–	–
1536	–	67.5	77.4	–	–	–	–	–	–	–
2304	–	–	–	–	–	–	–	–	–	–

Estimations of Problem Constants

Table B.4: Estimated ρ constant for a configuration with 6 layers and 768 embedding dimension when varying the batch size.

batch_size	256	512	1024	2048	4096
ρ	48.9	52.6	55.0	56.4	57.9

Table B.5: Estimated Kurdyka–Łojasiewicz constant μ for various model configurations.

n_embd \ n_layer	3	6	9	12	15	18	21	24	27	30
384	–	3.4	3.1	3.1	3.0	–	2.9	–	–	2.7
576	–	3.3	3.2	3.0	2.9	–	2.7	2.6	2.5	2.4
768	3.7	3.2	3.0	2.9	2.8	–	2.6	2.5	2.3	2.4
1152	–	–	–	2.7	2.7	2.8	2.6	–	2.5	–
1536	–	3.2	2.9	–	2.9	3.0	–	–	–	–
2304	–	3.6	–	–	–	–	–	–	–	–

Estimations of Problem Constants

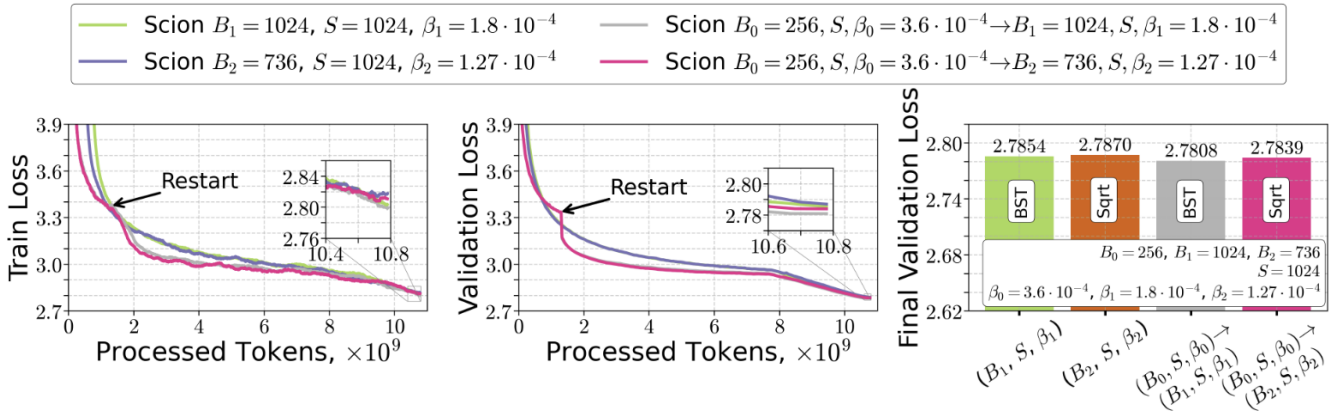


Figure C.1: Comparison of two strategies: BST scaling rule, where $BS \sim T^{2/3}$ (fixed 1024 batch size with $\beta_1 = 1.8 \cdot 10^{-4}$ in light green and restarted version in gray) and square-root rule (16), where $BS \sim T^{1/2}$ (fixed 736 batch size with $\beta_2 = 1.27 \cdot 10^{-4}$ in orange and restarted version in pink). The validation and train sequence lengths are fixed to 1024. (large batch size strategies when training a 1B model. Scion with a batch size of 1024 (fixed from the beginning or after a restart) achieves slightly better performance compared to the baselines, where the batch size is set according to the square-root rule (16). The notation $(B_{1,2}, S, \beta_{0,1})$ characterizes which batch size, sequence length, and Frank–Wolfe stepsize are used for the particular setup, respectively. The notation $(B_0, S, \beta_0) \to (B_{1,2}, S, \beta_{1,2})$ characterizes how parameters of Scion change after restart (e.g., batch size increases from B_0 to $B_{1,2}$), respectively.