On the Role of Batch Size in Stochastic Conditional Gradient Methods

Rustem Islamov[1,4] Roman Machacek[2] Aurelien Lucchi[1] Antony Silveti-Falls[3]
Eduard Gorbunov[4,⋆] Volkan Cevher[5,⋆]

[1]University of Basel [2]University of Bern [3]CentraleSupélec [4]MBZUAI [5]EPFL

Problem Formulation

We want to solve a finite-sum problem

$$\min_{x\in\mathcal{X}} \left[f(x) = \mathbb{E}_{\xi\sim\mathcal{D}} f(x;\xi) \right]$$

$f^\star = \min_{x\in\mathcal{X}} f(x)$ — non-convex — data points — # model parameters — Empirical risk/loss — Dataset

$\mathcal{X}$ is equipped with

- a standard Euclidean norm $\|\cdot\|_2$ induced by the inner product $\langle\cdot,\cdot\rangle$: $\|x\|_2 = \langle x, x\rangle$;
- another norm $\|\cdot\|$ with the associated dual norm $\|x\|_* := \sup_{\|x'\|\le 1} \langle x, x'\rangle$.

Training LLMs deviates from classical optimization and exhibits unique characteristics.

- the training is conducated under a **fixed token budget** T, rather than a fixed number of optimization steps;
- Through the link $K := \frac{T}{BS}$, (B, S) and the stepsize together determine how efficiently the token budget translates into optimization progress.

How should (B, S) and the stepsize be chosen, and adapted, to optimize performance under a fixed token budget T?

Stochastic Conditional Gradient

Algorithm 1: SCG

Input: $x^0 \in \mathcal{X}$, $\alpha, \beta \in [0,1]$, $\eta > 0$
For $k = 0, \dots, K-1$ **do**
  Sample $\xi_k \in \mathcal{D}$
  Compute $m_{k+1} = (1-\alpha)m_k + \alpha g(x_k;\xi_k)$
  Compute $d_{k+1} = \arg\min_{d\in\mathcal{X}} \langle d, m_{k+1}\rangle$ s.t. $\|d\| \le 1$
  Update $x_{k+1} = (1-\beta)x_k + \beta\eta d_{k+1}$
End For

Assumptions

(A1) Smoothness. The gradient $\nabla f(\cdot)$ is L-Lipschitz with respect to the norm $\|\cdot\|$:

$$\|\nabla f(x) - \nabla f(x')\|_* \le L\|x - x'\| \quad \forall\, x, x' \in \mathcal{X}.$$

(A2) Norm Equivalence. There exists a constant $\rho > 0$ such that

$$\|x\|_* \le \rho\|x\|_2 \quad \forall\, x \in \mathcal{X}.$$

(A3) μ-Kurdyka-Łojasiewicz. The objective function f satisfies for some $\mu > 0$:

$$\|\nabla f(x)\|_* \ge \mu(f(x) - f^\star).$$

(A4) Bounded Variance. The stochastic gradient estimator $g(x;\xi)$ is unbiased and has σ^2-bounded variance:

$$\mathbb{E}[g(x;\xi)] = \nabla f(x), \quad \mathbb{E}[\|g(x;\xi) - \nabla f(x)\|_2^2] \le \sigma^2.$$

Empirical Verification of Assumptions

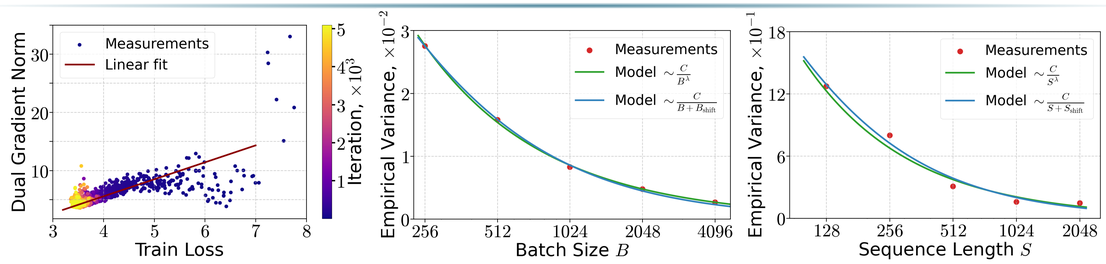

Figure 1: Empirical verification of the validity of the assumptions during the training of a 124M NanoGPT model FineWeb dataset under a fixed token budget $T = 2.7$B. Left: Assumption (A3). The points with a loss below 5 fit a linear function well, with a slope equal to μ. (Central and Right) Assumption (A4). For the central figure, the sequence length $S = 1024$, and the estimated scaling constants are $\lambda \approx 0.9$ and $B_{\text{shift}} \approx 90$. For the right figure, the batch size $B = 512$, and the estimated scaling constants are $\lambda \approx 1.1$ and $S_{\text{shift}} \approx 35$.

Convergence Theory

Theorem. Under the assumptions (A1)-(A4), let $m_0 = g(x_0;\xi_0)$ and parameters of SCG are chosen such that

$$\beta = \mathcal{O}\left(\frac{1}{K}\right), \quad \eta = \tilde{\mathcal{O}}\left(\frac{1}{\mu}\right), \quad \alpha = \min\left\{1, \mathcal{O}\left(\frac{(\varepsilon\mu)^2}{(\rho\sigma)^2}\right)\right\}, \quad 2\|x_0\| \le \eta, \text{ and } K = \max\left[\tilde{\mathcal{O}}(1), \tilde{\mathcal{O}}\left(\max\left\{\frac{L}{\varepsilon\mu^2}, \frac{\rho\sigma}{\varepsilon\mu}, \frac{L(\rho\sigma)^2}{\mu(\varepsilon\mu)^3}, \frac{(\rho\sigma)^3}{(\varepsilon\mu)^3}\right\}\right)\right],$$

where $\mathcal{O}$ hides all numerical constants and $\tilde{\mathcal{O}}$ hides all numerical and logarithmic factors. Then, the output of SCG after K iterations satisfies $\mathbb{E}[f(x_K) - f^\star] \le \varepsilon$.

BST Error Floor

Under the setup of the Theorem, with the SCG algorithm, we achieve the optimization error

$$\varepsilon = \tilde{\mathcal{O}}\left(\max\left\{\frac{LBS}{\mu^2 T}, \left(\frac{L\rho^2\sigma_\star^2}{\mu^4 T}\right)^{1/3}, \frac{\rho\sigma_\star}{\mu(T^2BS)^{1/6}}\right\}\right),$$

BST Scaling Rules

Let $B_0^\star$, $S_0^\star$, and $\beta_0^\star$ be the batch size, sequence length, and Frank-Wolfe stepsize tuned for a small model, then for a large model, we should use parameters

$$B_1 S_1 = B_0^\star S_0^\star \chi_{0\to1}^{2/3} \left(\frac{T_1}{T_0}\right)^{2/3}, \beta_1 = \beta_0^\star \chi_{0\to1}^{2/3} \left(\frac{T_1}{T_0}\right)^{-1/3},$$

where $\chi_{0\to1} = \mu_1\rho_1 L_0/\mu_0\rho_0 L_1$.

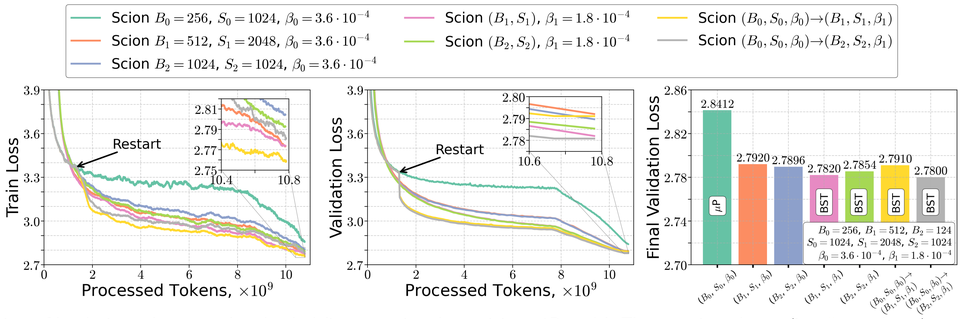

Figure 2: Comparison of batch size and sequence length scheduling strategies when training a 1B model. The restarting schemes (in yellow and gray) are compared against fixed schedules. The validation loss is evaluated with a smaller sequence length of 1024. Baselines following BST rules significantly outperform μP baseline.

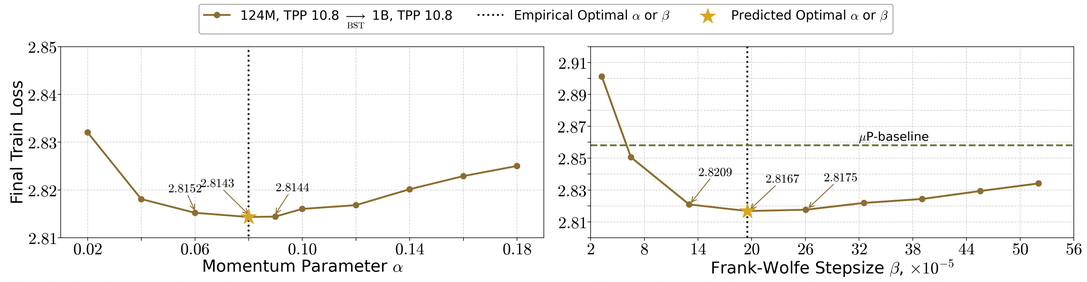

Figure 3: The final performance of the 1B model when varying the momentum parameter α (left) and Frank–Wolfe stepsize β (right) under different token budget 10.6 TPP. We observe that the BST scaling rule predicts a good estimate for both optimal α and β when transferring from a smaller 124M model to a larger 1B model.

Estimation of Problem-Dependent Constants

To estimate the change of problem-dependent constants ρ, L, μ, we fit power laws as a function of the number of layers n_l, embedding size n_e, and batch size B:

$$\mu(n_l, n_e) = 5.2(n_l + 1.7)^{-0.2};$$
$$L(n_l, n_e) = 0.4(n_l + 0.7)^{0.2}(n_e + 126)^{0.35};$$
$$\rho(n_l, n_e, B) = 4.1(n_l - 2.7)^{0.25}(n_e - 250.8)^{0.3}(B - 9.4)^{0.1}.$$

Table 1: Estimated problem-dependent constants according to the fitted laws. The estimations of the change for β and BS are based on BST scaling rules.

Model	L	μ	ρ	How to change β w.r.t. 124M model?	How to change BS w.r.t. 124M model?
124M	7.2	3.1	62.7	$1\times$	$1\times$
1B	10.6	2.9	111.9	$\searrow 0.5\times$ [(a)], $\searrow 0.54\times$ [(b)]	$\nearrow 4\times$ [(a)], $\nearrow 4.37\times$ [(b)]

[(a)] Taking into account the practical requirement that B and S should be powers of two. We increase the product BS, rounding to the closest power of two.
[(b)] Ignoring the practical requirement that B and S should be powers of two.

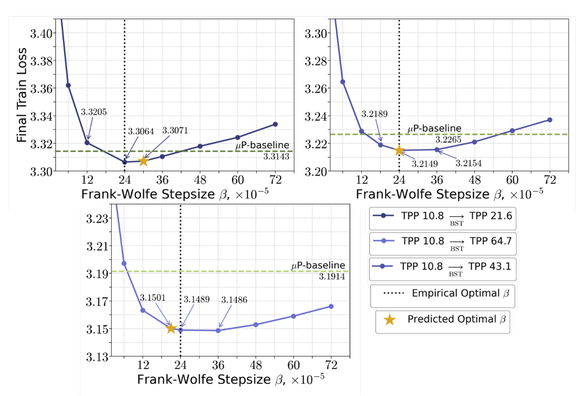

Figure 4: The final performance of the 124M model when varying the Frank–Wolfe stepsize β under different token budgets (left: 2.7B, right: 5.3B, bottom: 8.0B). We observe that the BST scaling rule predicts a good estimate for the optimal β when increasing the token budget. Moreover, the difference in performance between BST and μP baselines grows with a token budget.

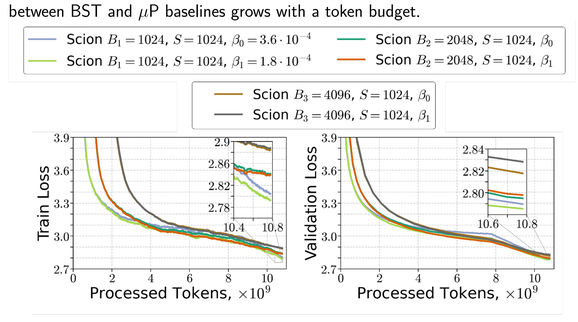

Figure 5: Comparison of fixed large batch size strategies when training a 1B model. The validation loss is evaluated with a smaller sequence length 1024. Scion with a batch size of 1024 suggested by our BST scaling rule achieves the best performance compared to other baselines with batch sizes 2048 and 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.