# TOWARDS JOINT SCALING LAWS WITH OPTIMAL BATCH SIZE SCHEDULES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Modern deep learning typically keeps the batch size static throughout training, thus overlooking the joint effect of learning rate and batch size on the training dynamics. In this paper, we study the deep learning dynamics through the lens of convex optimization and derive a joint characterization of loss in terms of both schedules, applicable to general optimizers and model architectures. This characterization yields a closed-form optimal batch size schedule for any prescribed learning rate schedule, and further leads to joint scaling laws that consistently outperform static batch size baselines, highlighting the significance of dynamic batch size schedule in large language model training.

## 1 INTRODUCTION

Large-scale model training has been central to recent progress in artificial intelligence (Kaplan et al., 2020; Hoffmann et al., 2022a), but its effectiveness depends critically on robust choices of hyperparameters, particularly the learning rate and batch size (BS). The learning rate controls the magnitude of parameter updates: overly small values slow optimization, whereas overly large values can destabilize training and cause loss divergence (Yang et al., 2021; Li et al., 2025). The batch size controls a trade-off between model quality and systems efficiency: larger batches can improve throughput by exploiting data parallelism, amortizing communication, and increasing hardware utilization (Shallue et al., 2019; Goyal et al., 2017; Rajbhandari et al., 2020); however, beyond a critical regime, increasing the batch size can degrade model performance under fixed data or compute budgets, because it reduces the number of optimization steps and changes the gradient-noise profile (McCandlish et al., 2018; Golmant et al., 2018; Keskar et al., 2017). Consequently, hyperparameter selection is essential for efficient model training, yet exhaustive tuning is often prohibitively computationally expensive at frontier scale. This motivates principled methods for transferring or scaling hyperparameters across model sizes and training horizons, so that small-scale experiments can reliably inform large-scale ones (Kaplan et al., 2020; Li et al., 2025; Hu et al., 2024; Zhang et al.; Bergsma et al., 2025a; Shuai et al., 2024; Bi et al., 2024).

For the learning rate, the mainstream practice is to use a schedule $\{\eta_t\}_{t=1}^{T}$ rather than a fixed constant. Common schedules typically combine warmup, which mitigates early-training instability, with decay, which improves late-stage optimization (Vaswani et al., 2017; Godbole et al., 2023; Defazio et al., 2023; Schaipp et al., 2025). Under maximal update parametrization ($\mu$P), optimal learning rates can transfer across model sizes when the training horizon is fixed (Yang et al., 2021; 2024; Dey et al., 2026). Furthermore, when the training horizon $T$ changes, the optimal peak learning rate must be rescaled, often following a power law of the form $\eta^\star(T) \propto T^{-\alpha}$ (Bi et al., 2024; Bjorck et al., 2025; Bu et al., 2026).

In contrast, batch size has been less explored systematically as a scheduled hyperparameter. Existing approaches often use fixed batch sizes or hand-designed batch-size schedules coupled to the learning-rate schedule (Smith et al., 2018; Balles et al., 2017; Meterez et al., 2025). See Appendix E for a summarization of batch size schedules for recent LLMs. This is potentially suboptimal: a fixed batch size is only a constrained special case of a broader class of batch-size schedules $\{B_t\}_{t=1}^{T}$. This gap motivates a principled study of batch-size scheduling and its transfer across model sizes and training horizons.

In this work, we study the fundamental question:

**What is the optimal batch size schedule and which factors determine it?**



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We build on stochastic convex optimization theory to derive a sequence-to-sequence characterization of loss from learning-rate and batch-size schedules. This empirically accurate characterization yields a constrained optimization over batch-size schedules, whose closed-form solution is given in Theorem 3. The resulting schedule depends on the computational budget and the shape of the learning-rate schedule, but not on its peak value, weight decay, model size, architecture, or optimizer.

**Contributions:**

1. **Loss prediction by learning rate and batch size schedules.** We derive an any-iteration loss characterization in equation 2.5 for arbitrary learning-rate and batch-size schedules, empirically accurate across optimizers and model architectures.
2. **Closed-form optimal batch size schedule.** We derive the optimal batch-size schedule under data, compute, and general resource budgets (Theorem 3 and lemma 1). The schedule is independent of the peak learning rate and therefore requires no additional learning-rate tuning.
3. **Joint scaling laws and universal dynamics.** We build joint scaling laws that employ $1/\sqrt{T}$-scaling of learning rate and weight decay, as well as our scale-free batch size schedule, with an improvement on compute efficiency by $6 \sim 15\%$ and zero computational overhead. Crucially, we show that the optimal batch size schedule preserves universal training dynamics, also known as supercollapse, rendering a robust and high-performing recipe for scaling.
4. **Empirical ablations in various regimes.** We validate the schedule across pre-training and fine-tuning, Llama3, Qwen3 MoE, and vision-language models up to 7B parameters, using AdamW and Muon optimizers (see Appendix C).

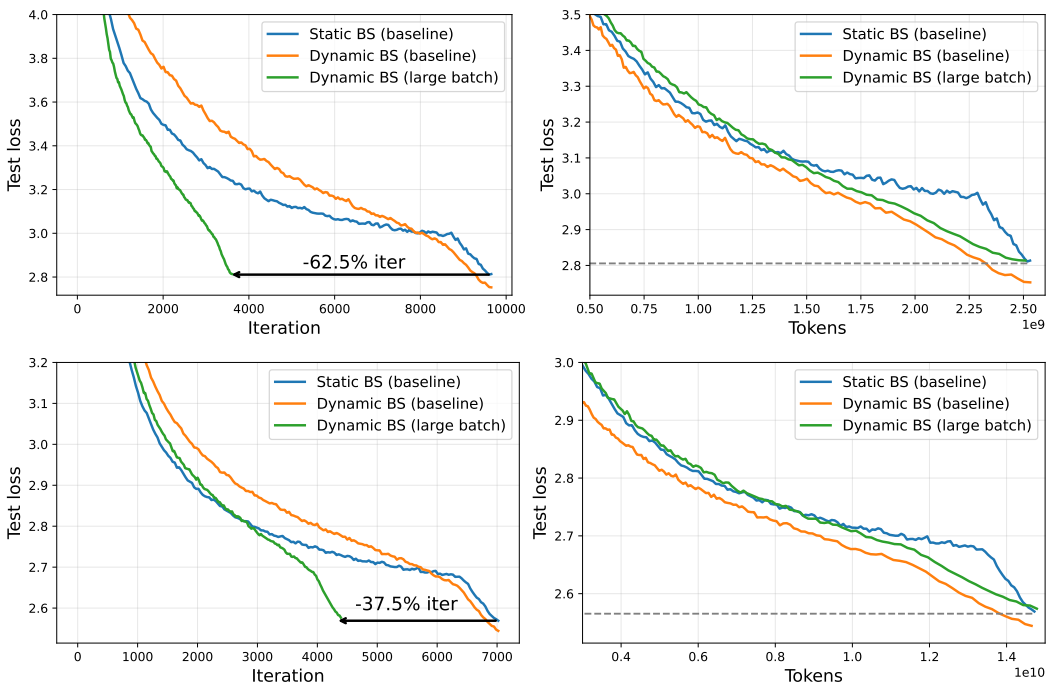



**Figure 1:** Dynamic batch size outperforms static batch size under fixed compute of 1e20 FLOPs on Fineweb-edu dataset: (I) for fixed training horizon, dynamic batch size schedule (same BS) achieves better loss than static baselines; (II) for the same loss, dynamic batch size schedule (larger BS) uses significantly fewer iterations than static baselines, with the same learning rate. Upper row: Llama3-7B (dense). Lower row: Qwen3-A1B (MoE). See the evaluation in Table 1 and Figure 9.

**Table 1:** Dynamic batch size generally outperforms static batch size (marked in green) by lm-evaluation-harness v0.4 on the last-iterate. See Figure 9 for the evaluation on intermediate iterates.

| Model | Batch Size | LAMBADA | HellaSwag | ARC-Easy | ARC-Challenge | WinoGrande | PIQA | OpenBookQA | BoolQ |
|---|---|---|---|---|---|---|---|---|---|
| **Llama3 (dense)** | static | 32.23 | 42.93 | 56.22 | 29.26 | 53.03 | 67.89 | 32.40 | 61.89 |
| | dynamic | 35.36 | 44.33 | 56.36 | 30.46 | 53.99 | 68.34 | 33.20 | 62.26 |
| **Qwen3 (MoE)** | static | 36.06 | 45.61 | 57.83 | 32.59 | 52.49 | 68.77 | 34.00 | 61.47 |
| | dynamic | 37.71 | 46.80 | 59.09 | 32.25 | 55.33 | 69.97 | 36.20 | 61.59 |

## 2 Loss prediction by batch size and learning rate sequences

We start with the stochastic gradient descent (SGD) as $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t$, where $\mathbf{w}_t$ is the parameters, $\eta_t$ is the learning rate, $\mathbf{g}_t = \mathbf{g}(\mathbf{w}_t)$ is the mini-batch gradient with batch size $B_t$, and $1 \le t \le T$ with $T$ total iterations. In what follows, we denote $\mathbf{w}_*$ as the minimizer of loss $L$, and $D = \|\mathbf{w}_t - \mathbf{w}_*\|$.

### 2.1 From learning rate sequence to loss sequence

The prediction from learning rate to loss value has been investigated under convexity or strong convexity loss, Lipschitz continuity or smoothness, and averaged or current iterate. We follow the work of Defazio et al. (2023); Bu et al. (2026); Schaipp et al. (2025) and study the sequence-to-sequence prediction under convex loss and SGD in Theorem 1 (see proof in Appendix A.1).

**Theorem 1.** *For convex loss and any learning rate sequence $\eta_t$, under SGD optimizer, we have*

$$\mathbb{E}L(\mathbf{w}_\tau; \eta_t) \le L_* + \frac{D^2}{2\sum_{t=1}^{\tau}\eta_t} + \frac{1}{2}\sum_{t=1}^{\tau-1}\frac{\eta_t^2 \mathbb{E}\|\mathbf{g}_t\|^2}{\sum_{k=t+1}^{\tau}\eta_k} + \frac{1}{2}\eta_\tau \mathbb{E}\|\mathbf{g}_\tau\|^2 \tag{2.1}$$

We emphasize that although the theoretical basis for Theorem 1 is limited to convex loss and SGD optimizer, the empirical applicability generalizes to non-convex neural networks (e.g. LLM and VLM) and adaptive optimizers (e.g. AdamW and Muon) as extensively investigated in Bu et al. (2026).

### 2.2 From batch size sequence to loss sequence

To understand the effect of batch size on the loss, we simply assume

$$\mathbb{E}\|\mathbf{g}_t\|^2 \le G^2 + X/B_t \tag{2.2}$$

where $G$ upper bounds the norm of expected gradient $\|\mathbb{E}\mathbf{g}_t\|$, and $X$ upper bounds the trace of per-example gradient covariance matrix. We derive equation 2.2 and extend it to adaptive optimizers in Appendix A.2.

**Theorem 2.** *Assuming equation 2.2, for convex loss and any learning rate sequence $\eta_t$, we have*

$$\mathbb{E}L(\mathbf{w}_\tau; \eta_t, B_t) \le L_* + \frac{D^2}{2\sum_{t=1}^{\tau}\eta_t} + \frac{1}{2}\sum_{t=1}^{\tau-1}\frac{\eta_t^2(G^2 + X/B_t)}{\sum_{k=t+1}^{\tau}\eta_k} + \frac{\eta_\tau}{2}(G^2 + X/B_\tau) \tag{2.3}$$

*Proof of Theorem 2.* Substituting equation 2.2 to Theorem 1. □

### 2.3 Precise characterization under linear regression

We validate Theorem 2 and thus justify the assumptions equation 2.1 and equation 2.2 on the various sequences of learning rate and batch size in Figure 2, where we construct three variables

$$x_{1,\tau} = \frac{1}{2\sum_{t=1}^{\tau}\eta_t}, x_{2,\tau} = \frac{1}{2}\sum_{t=1}^{\tau-1}\frac{\eta_t^2}{\sum_{k=t+1}^{\tau}\eta_k} + \frac{\eta_\tau}{2}, x_{3,\tau} = \frac{1}{2}\sum_{t=1}^{\tau-1}\frac{\eta_t^2/B_t}{\sum_{k=t+1}^{\tau}\eta_k} + \frac{\eta_\tau/B_\tau}{2}$$

then solve a linear regression[1] with $\boldsymbol{L} = [L_1, ..., L_T]$:

$$\min_{L_*, D, G, X} \left\| L_* + D^2\boldsymbol{x}_1 + G^2\boldsymbol{x}_2 + X\boldsymbol{x}_3 - \boldsymbol{L} \right\|_2^2 \tag{2.4}$$

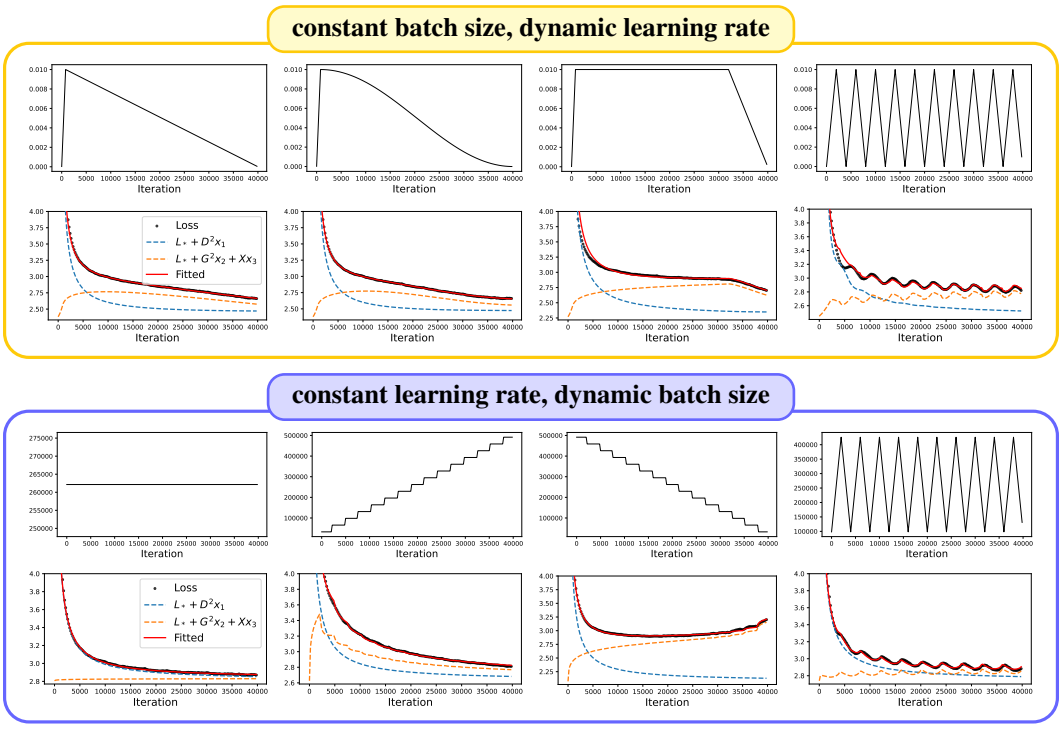



**Figure 2:** Sequence-to-sequence prediction by Theorem 2 for Llama3-1B model on Fineweb-edu data, trained by Muon-NSGD optimizer. Top two rows: $B_t$ is fixed ($B = 256 * 1024$) and $\eta_t$ is dynamic. Bottom two rows: $\eta_t$ is fixed ($\eta = 0.01$) and $B_t$ is dynamic. Similar patterns for AdamW optimizer can be observed in Figure 10.

As shown in Figure 2, for all cases, a precise characterization emerges after an initial transient phase. This indicates that the training dynamics become dominated by convexity, and that the inequality in Theorem 2 may be replaced by equality in practice[2]:

$$\mathbb{E}L(\mathbf{w}_\tau) \approx L_* + \frac{D^2}{2\sum_{t=1}^{\tau}\eta_t} + \frac{1}{2}\sum_{t=1}^{\tau-1}\frac{\eta_t^2(G^2 + X/B_t)}{\sum_{k=t+1}^{\tau}\eta_k} + \frac{\eta_\tau}{2}(G^2 + X/B_\tau) \tag{2.5}$$

Specifically, if the learning rate decays to zero ($\eta_T = 0$), then for the last iterate,

$$\mathbb{E}L(\mathbf{w}_T) \approx L_* + \frac{D^2}{2\sum_{t=1}^{T}\eta_t} + \frac{1}{2}\sum_{t=1}^{T-1}\frac{\eta_t^2(G^2 + X/B_t)}{\sum_{k=t+1}^{T}\eta_k} \tag{2.6}$$

## 3 Continuous loss prediction and optimal batch size schedule

Now we optimize the batch size with respect to the last-iterate loss in equation 2.6, subject to some constraints on the data size, compute budget, wall-clock training time, etc. To derive closed-form solutions, we will work in the continuous regime and replace the summations in equation 2.6 by integrals:

$$L_* + \frac{D^2}{2\int_0^T \eta_t dt} + \frac{G^2}{2}\int_0^T \left(\frac{\eta_t^2}{\int_t^T \eta_k dk}\right) dt + \frac{X}{2}\int_0^T \left(\frac{\eta_t^2/B_t}{\int_t^T \eta_k dk}\right) dt \tag{3.1}$$

[1] If $B_t$ is constant, then the features $(\boldsymbol{x}_2, \boldsymbol{x}_3)$ are collinear and perfectly correlated, rendering the coefficients $(G, X)$ indistinguishable, although the linear regression is solvable.

[2] $X/G^2$ is essentially the critical batch size $B_{\text{simple}}$ defined in McCandlish et al. (2018), which we treat as time-independent in line with Merrill et al.; Golmant et al. (2018).

For static batch size $B_t = B_{\text{static}}$, we get

$$L_{\text{static}}(T) := L_* + \frac{D^2}{2\int_0^T \eta_t dt} + \frac{G^2}{2}\int_0^T \left(\frac{\eta_t^2}{\int_t^T \eta_k dk}\right) dt + \frac{X}{2B_{\text{static}}}\int_0^T \left(\frac{\eta_t^2}{\int_t^T \eta_k dk}\right) dt \quad (3.2)$$

### 3.1 Data-optimal and compute-optimal batch size schedule

We optimize the batch size given a fixed amount of data budget $K = B_{\text{static}}T$, while the learning rate schedule and the number of total iterations $T$ are fixed. This is equivalent to optimizing the batch size given a fixed compute budget $C = 6NK$ (FLOPs), where $N$ is model size.

It suffices to only minimize the last term of equation 3.1 with respect to $B_t$:

$$\min_{B_t} \int_0^T \left(\frac{1}{B_t}\frac{\eta_t^2}{\int_t^T \eta_k dk}\right) dt, \quad \text{s.t.} \int_0^T B_t dt = K \quad (3.3)$$

We solve this problem in Theorem 3 with a proof in Appendix A.3.

**Theorem 3.** *The closed-form solution to equation 3.3 is*

$$B_t^{\text{optim}} = K \cdot \frac{b_t}{\int_0^T b_t dt} = \frac{K}{2}\frac{\eta_t}{\sqrt{\int_0^T \eta_k dk \int_t^T \eta_k dk}} \text{ where } b_t := \frac{\eta_t}{\sqrt{\int_t^T \eta_k dk}} \quad (3.4)$$

*Importantly, the optimal batch size schedule only depends on the shape of the learning rate schedule, not the peak learning rate.* ***Thus,*** $B_t^{\text{optim}}$ ***is fully decoupled from learning rate tuning and scale-free in terms of model size*** *(see Table 2 for the independence on $\eta_t$, and Figure 11 for ablations on peak learning rate).*

We summarize the optimal batch size schedules corresponding to different specific learning rate schedules, which are visualized in Figure 3 [3]. The endpoint behavior of $B_t^{\text{optim}}$ is governed by the *order* at which the learning rate approaches zero: if $\eta_t \sim (T-t)^p$ as $t \to T$, then $b_t = \eta_t/\sqrt{\int_t^T \eta_k\, dk} \sim (T-t)^{(p-1)/2}$, so schedules with a nonzero terminal slope ($p = 1$, e.g. linear and the WSD cooldown) yield a constant terminal batch size, whereas schedules that flatten to zero ($p = 2$, e.g. cosine with $\eta_t \propto (T-t)^2$) drive $B_t^{\text{optim}} \to 0$.

**Remark 1.** *Our characterization treats learning rate and batch size separately, unlike approaches based only on the ratio $\eta_t/B_t$ (Goyal et al., 2017; Smith et al., 2018; McCandlish et al., 2018). With matched $\eta_t/B_t$ trajectories and identical token budgets, learning-rate scheduling consistently outperforms the corresponding batch-size scheduling; see Appendix C.5. Thus, $\eta_t/B_t$ is not a sufficient statistic for training dynamics.*

**Table 2:** Optimal batch size schedules for different learning rate schedules.

| learning rate | $\eta_t$ formula | $b_t$ formula | $B_t^{\text{optim}}$ formula |
|---|---|---|---|
| constant | $\eta$ | $\sqrt{\frac{\eta}{T-t}}$ | $\frac{B_{\text{static}}}{2}\sqrt{\frac{T}{T-t}}$ |
| cosine | $\frac{\eta}{2}(\cos(\frac{\pi t}{T})+1)$ | $\sqrt{\frac{\eta}{2}}\frac{1+\cos(\pi t/T)}{\sqrt{T-t-T\sin(\pi t/T)/\pi}}$ | $\frac{B_{\text{static}}\cos^2\left(\frac{\pi t}{2T}\right)}{\sqrt{1-\frac{t}{T}-\frac{1}{\pi}\sin\frac{\pi t}{T}}}$ |
| linear | $\eta(1-t/T)$ | $\sqrt{\frac{2\eta}{T}}$ | $B_{\text{static}}$ |
| WSD | $\begin{cases}\eta & \text{if } t \le cT \\ \eta\frac{T-t}{T-cT} & \text{if } t > cT\end{cases}$ | $\begin{cases}\sqrt{\frac{2\eta}{T+cT-2t}} & \text{if } t \le cT \\ \sqrt{\frac{2\eta}{T-cT}} & \text{if } t > cT\end{cases}$ | $\begin{cases}\frac{B_{\text{static}}T}{\sqrt{(T+cT-2t)(T+cT)}} & \text{if } t \le cT \\ \frac{B_{\text{static}}}{\sqrt{1-c^2}} & \text{if } t > cT\end{cases}$ |

### 3.2 Optimal batch size leverages variance reduction

To demonstrate the advantage of the optimal batch size schedule, we first derive the last-iterate loss.

[3] We observe no empirical difference whether the learning rate warmup is calculated in $B_t^{\text{optim}}$ or not.

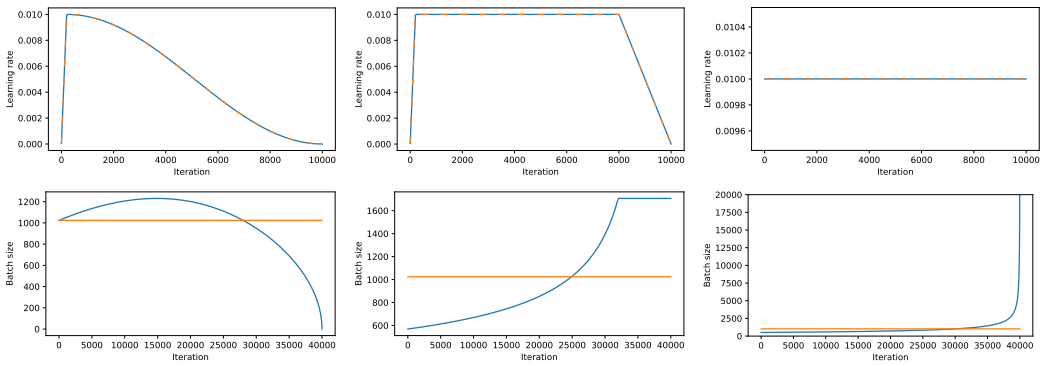



**Figure 3:** Optimal batch size schedules (blue) by Theorem 3 and static schedules (orange) for cosine, WSD, and constant learning rate schedules.

**Corollary 1.** *Substituting equation 3.4 back to equation 3.1, we have*

$$L_{\text{dynamic}}(T) := L_* + \frac{D^2}{2\int_0^T \eta_t dt} + \frac{G^2}{2}\int_0^T \left(\frac{\eta_t^2}{\int_t^T \eta_k dk}\right) dt + \frac{X}{2B_{\text{static}}T}\left(\int_0^T \frac{\eta_t}{\sqrt{\int_t^T \eta_k dk}} dt\right)^2$$

Secondly, we compare $L_{\text{dynamic}}$ in Corollary 1 to $L_{\text{static}}$ in equation 3.2, where the only difference is in the last terms. This difference explains the advantage of optimal batch size through the lens of variance reduction, as we show in Corollary 2.

**Corollary 2.** *The difference between dynamic and static batch size losses is always non-positive* $L_{\text{dynamic}}(T) - L_{\text{static}}(T) \leq 0$.

Additionally, we derive three facts from Table 3.

**Variance reduction:** Our batch size schedule reduces the gradient covariance term with $\frac{X}{B_{\text{static}}}$. For example, dynamic batch size improves $1.061\frac{X}{B_{\text{static}}} \to \frac{X}{B_{\text{static}}}$ for cosine learning rate, and $2.47\frac{X}{B_{\text{static}}} \to 1.9\frac{X}{B_{\text{static}}}$ for WSD learning rate with 10% cooldown.

**Scaled learning rate:** The optimal peak learning rate for our batch size schedule is $1/\sqrt{T}$-scaled for qualified learning rate, by minimizing $L(T) - L_*$ over $\eta$ in the third column.

**Optimal loss convergence rate:** Our batch size schedule consequently achieves $O(1/\sqrt{T})$ loss convergence in the last column.

**Table 3:** Loss prediction by dynamic/static batch size schedules for different learning rate schedules.

| learning rate | batch size | $L(T) - L_*$ | optimal $L(T) - L_*$ |
|---|---|---|---|
| constant | dynamic | $\frac{D^2}{2T\eta} + \frac{\eta}{2}G^2 \ln T + \frac{2\eta X}{B_{\text{static}}}$ | $D\sqrt{G^2\frac{\ln T}{T} + 4X/B_{\text{static}}\frac{1}{T}}$ |
| | static | $\frac{D^2}{2T\eta} + \frac{\eta}{2}G^2 \ln T + \frac{\eta X}{2B_{\text{static}}} \ln T$ | $D\sqrt{G^2\frac{\ln T}{T} + X/B_{\text{static}}\frac{\ln T}{T}}$ |
| cosine | dynamic | $\frac{D^2}{T\eta} + 1.061\eta G^2 + \frac{X\eta}{B_{\text{static}}}$ | $2D\sqrt{1.061G^2 + X/B_{\text{static}}}\sqrt{\frac{1}{T}}$ |
| | static | $\frac{D^2}{T\eta} + 1.061\eta G^2 + 1.061\frac{X\eta}{B_{\text{static}}}$ | $2D\sqrt{1.061G^2 + 1.061X/B_{\text{static}}}\sqrt{\frac{1}{T}}$ |
| WSD | dynamic | $\frac{D^2}{(1+c)T\eta} + \eta G^2\left[1 + \frac{1}{2}\ln\frac{1+c}{1-c}\right] + \eta X/B_{\text{static}}(1+c)$ | $\frac{2D}{\sqrt{(1+c)T}}\sqrt{\left[1 + \frac{1}{2}\ln\frac{1+c}{1-c}\right]G^2 + (1+c)\frac{X}{B_{\text{static}}}}$ |
| | static | $\frac{D^2}{(1+c)T\eta} + \eta(G^2 + X/B_{\text{static}})\left[1 + \frac{1}{2}\ln\left(\frac{1+c}{1-c}\right)\right]$ | $\frac{2D}{\sqrt{(1+c)T}}\sqrt{G^2 + \frac{X}{B_{\text{static}}}}\sqrt{1 + \frac{1}{2}\ln\left(\frac{1+c}{1-c}\right)}$ |

### 3.3 General computational cost-optimal batch size schedule

Theoretically, we can derive the optimal batch size for any general computational cost function $f$, as an extension of equation 3.3, e.g. the wall-clock training time.

$$\min_B \int_0^T \left(\frac{\eta_t^2}{B_t}\frac{1}{\int_t^T \eta_s ds}\right) dt, \quad s.t. \int_0^T f(B_t)dt = K \tag{3.5}$$

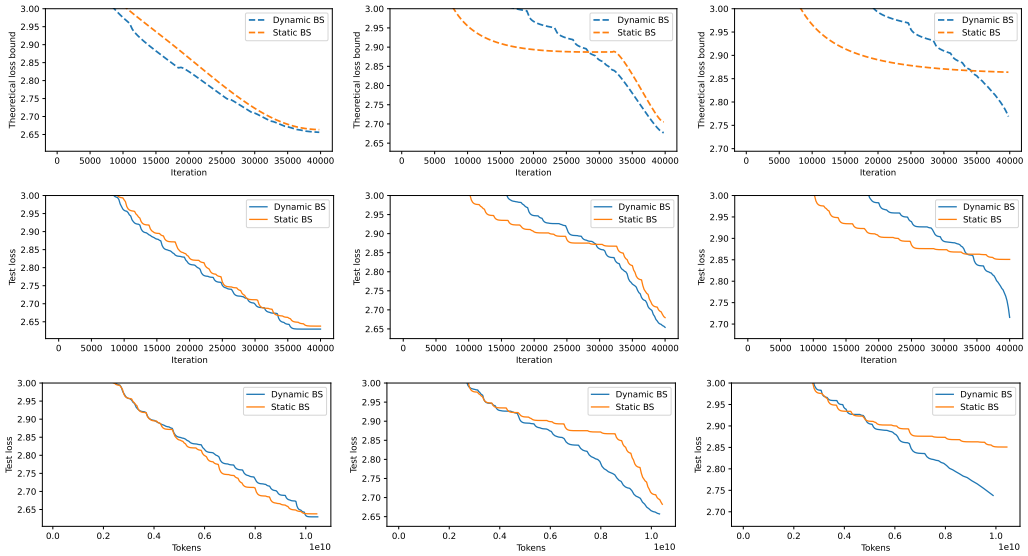



**Figure 4:** Dynamic batch size consistently outperforms static batch size for Llama3-1B model on Fineweb-edu data, trained by Muon-NSGD optimizer. Left to right: cosine, WSD, and constant learning rate. Relative improvement of best perplexity is 0.8%, 2.5%, and 14.6%, respectively.

Notice that when $f(B) = B$, the following reduces to the data-optimal case in Theorem 3. We obtain the closed form solution to equation 3.5 as $B_t^*(\lambda)$ in Lemma 1 (see proof in Appendix A.6).

**Lemma 1.** *Denote $g^{-1}(x) = x^2 f'(x)$, then we can determine $\lambda$ and $B_t^*$ through*

$$\int_0^T f\left(g\left(\frac{\eta_t^2}{\int_t^T \eta_s ds}/\lambda\right)\right) dt = K, \quad B_t^* = g\left(\frac{\eta_t^2}{\int_t^T \eta_s ds}/\lambda\right)$$

## 4 Joint Scaling Laws and Universal Dynamics

### 4.1 Hyperparameter Scaling

With our batch size schedule in Theorem 3, we present the joint scaling laws of multiple hyperparameters, which extends the single scaling law of peak learning rate in Bu et al. (2026): with $\eta^*(N,T)$ and $\kappa^*(N,T)$ being the optimal peak learning rate and weight decay for model size $N$ and horizon $T$,

$$\begin{aligned} \text{peak learning rate: } & \eta^*(N,T) = \eta^*(N_{\text{small}}, T_{\text{small}})/\sqrt{T/T_{\text{small}}} \\ \text{optimal batch size: } & B_t^{\text{optim}}(T) = \frac{\text{total data}}{2} \frac{\eta_t^*}{\sqrt{\int_0^T \eta_k^* dk \int_t^T \eta_k^* dk}} \\ \text{weight decay: } & \kappa(N,T) = \kappa^*(N_{\text{small}}, T_{\text{small}})/\sqrt{T/T_{\text{small}}} \end{aligned} \tag{4.1}$$

We theoretically validate equation 4.1 in Theorem 4, showing that $1/\sqrt{T}$ scaling of peak learning rate and weight decay leads to asymptotically optimal loss convergence.

**Theorem 4.** *Assuming equation 2.2, convex loss and bounded parameters, for SGD with weight decay $\mathbf{w}_{t+1} = (1-\eta_t\kappa)\mathbf{w}_t - \eta_t \mathbf{g}_t$ and for qualified schedule with $\eta_{ref}/\sqrt{T}$ peak learning rate, we have*

$$\mathbb{E}L(\mathbf{w}_T) - L_* = O(1/\sqrt{T})$$

*under static batch size and optimal batch size schedule if $\kappa = O(1/\sqrt{T})$.*

We demonstrate on the Llama3 and Qwen3 MoE families that the optimal batch size schedule is highly beneficial, on a wide range of model sizes (from 0.1B to 7B) and compute budgets (up to

1e21 FLOPs). We visualize two experiments in Figure 5, one with fixed compute and the other with fixed TPP. For the static batch size, we determine the optimal learning rate by following Bu et al. (2026): the peak learning rate is scaled as $\eta_{\text{ref}}/\sqrt{T}$, independent of model size, where $\eta_{\text{ref}}$ is tuned on small-scale runs. Weight decay is determined in the same manner. We emphasize that our batch size schedule directly uses the same learning rate and weight decay tuned for the static batch size, rendering the comparison fair but conservative. Nevertheless, in both experiments, the optimal batch size significantly outperforms the static batch size, with 15% improvement of compute efficiency on Llama3 as well as 6% improvement on Qwen3 MoE (see the middle column). Especially, the performance is consistent for most iterations during training.

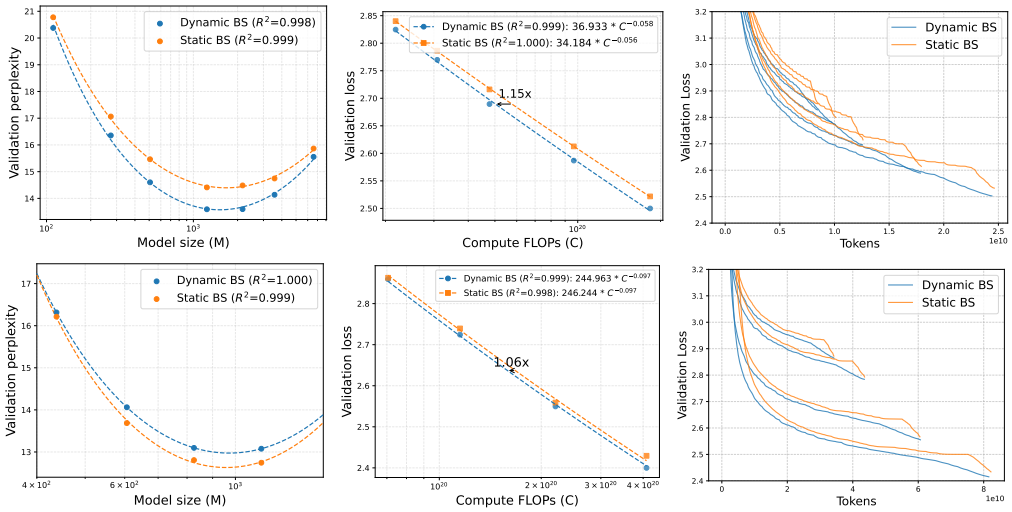



**Figure 5:** Dynamic batch size significantly outperforms static batch size for Llama3 (top row) and Qwen3 MoE (bottom row). Left: perplexity under fixed compute of 1e20 FLOPs. Quadratic functions are fitted for validation loss and log(model size) following Hoffmann et al. (2022b). Middle & Right: TPP=50 for Llama3 and TPP=100 for Qwen3 MoE. Power laws are fitted for validation loss and compute FLOPs.

### 4.2 Universal Dynamics

Crucially, our joint scaling laws lead to the universal dynamics (Qiu et al., 2025; Bergsma et al., 2025b), where different models share the same normalized loss curve when

(I) TPP is fixed (governing depth, width, and training horizon).

(II) Hyperparameters are scaled (governing learning rate and weight decay).

In Figure 6, each loss curve is normalized on the training horizon $x = t/T$ and on the validation loss $F(x) = \frac{L(\mathbf{w}_{xT}) - L_*}{L(\mathbf{w}_T) - L_*}$. In addition, we observe another universal dynamics when the loss is normalized with respect to tokens $x = \frac{\int_0^t B_k dk}{\int_0^T B_k dk}$, besides to iterations. Notice that these two types of normalization are equivalent for static batch size but different for dynamic batch size. As a result, our joint scaling laws enable the robust prediction of future scaling and demonstrate the advantage of optimal batch size.

## 5 Post-Training Experiments

We further test our batch size schedule on two post-training experiments—vision-language model (VLM) and math-domain fine-tuning. See additional experiment details in Appendix D.

### 5.1 Vision-Language Model

We train vision-language models (VLM) with SmoLM2-360M-Instruct as language backbone and Siglip2-large as vision backbone, following nanoVLM codebase (Wiedmann et al., 2025) on the Cauldron dataset (Laurençon et al., 2024). We train with Muon-NSGD optimizer. The static batch size is 0.26M tokens for text and 512*512 pixels for image.

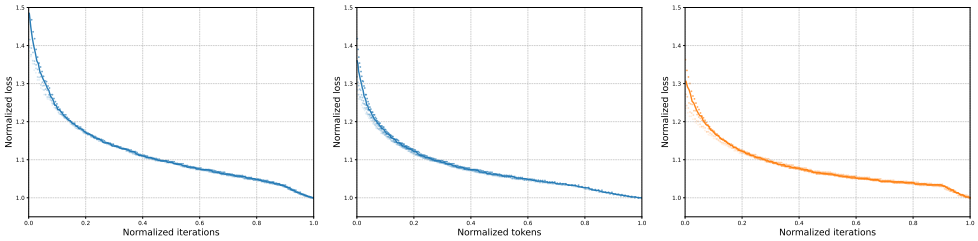



**Figure 6:** Dynamic batch size has universal dynamics with respect to normalized iterations and normalize tokens, similar to the static batch size for which two types of normalization are equivalent. Each plot shows 4 runs of Qwen3 MoE with TPP=100 from 0.4 to 2B parameters, with lighter color being smaller model size .

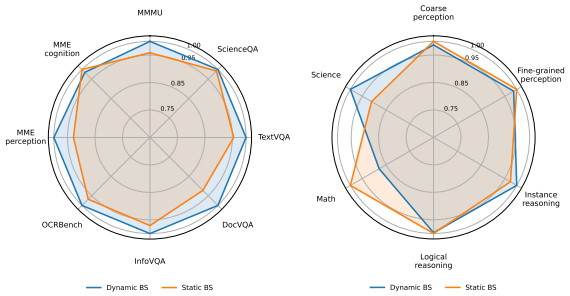



**Figure 7:** Normalized radar comparison of dynamic vs. static batch size across VLM benchmarks. Left: MMStar subtasks. Right: other benchmarks (MMMU, ScienceQA, TextVQA, DocVQA, InfoVQA, OCRBench, and MME). Each axis is normalized by $\max(\text{Dynamic BS}, \text{Static BS})$ so that mixed-scale metrics are visually comparable. See Table 6 for absolute scores.

Figure 7 and Table 6 compare the dynamic and static batch sizes across various VLM evaluation tasks, where the dynamic batch size outperforms the static baseline on 7 of 9 tasks, while the static baseline can be slightly better on some MMStar subtasks. Overall, we see that dynamic batch size is competitive or superior for VLM tasks, with no significant regression on any benchmark.

### 5.2 Math-domain fine-tuning

We fine-tune Qwen3-0.6B (Team, 2025) on `orca_math` dataset (Mitra et al., 2024) with AdamW optimizer and WSD learning rate schedule. The peak learning rate is selected via a sweep with a constant batch size. In Figure 8, dynamic batch size shows better overall trajectories on both validation loss and GSM8K accuracy than static batch size.

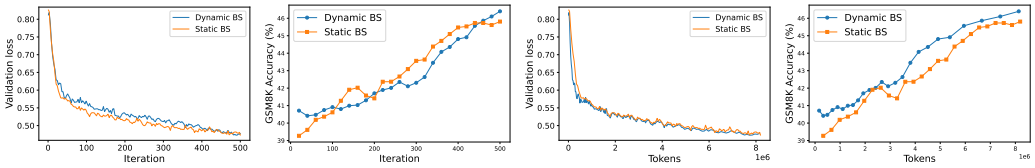



**Figure 8:** Dynamic batch size consistently outperforms static batch size on math-domain fine-tuning. Left: validation loss and 5-shot GSM8K accuracy over iterations. Right: the same metrics plotted against total tokens consumed.

## 6 Conclusion

We characterize deep-learning training dynamics jointly through learning-rate and batch-size schedules and use this characterization to derive a closed-form optimal batch-size schedule that is decoupled from peak learning-rate tuning. Combined with $1/\sqrt{T}$ scaling of peak learning rate and weight decay, the resulting joint scaling laws improve compute efficiency by 6–15% and preserve universal training dynamics across scales. Experiments across model families, optimizers, learning-rate schedules, and post-training settings further validate dynamic batch sizing as a principled scaling dimension.

## AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for [assisting in the writing of proofs, designing or providing feedback on research methodology or experiments, implementing methods, interpreting results]. We have not used generative AI tools for [generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, proposing or refining hypotheses, cleaning and reformatting dataset, supporting qualitative and thematic data analysis], and [generating synthetic data sets, assisting with translation] are not applicable to this work. Additionally, we used generative AI tools for [creating or modifying scientific figures or images, drafting parts of a research paper, summarizing or analysing existing literature, editing a research paper to improve readability, identifying relevant literature, suggesting a structure for a research paper]. We have reviewed all AI-assisted work. LLM-generated content was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Lucas Balles, Javier Romero, and Philipp Hennig. Coupling adaptive batch sizes with learning rates. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017. URL http://auai.org/uai2017/proceedings/papers/141.pdf. 1

Shane Bergsma, Nolan Simran Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Power lines: Scaling laws for weight decay and batch size in LLM pre-training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=bFXbLQzRoZ. 1

Shane Bergsma, Bin Claire Zhang, Nolan Dey, Shaheer Muhammad, Gurpreet Gosal, and Joel Hestness. Scaling with collapse: Efficient and predictable training of llm families. *arXiv preprint arXiv:2509.25087*, 2025b. 8

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024. 1

Johan Bjorck, Alon Benhaim, Vishrav Chaudhary, Furu Wei, and Xia Song. Scaling optimal lr across token horizons. In *International Conference on Learning Representations*, volume 2025, pp. 83640–83657, 2025. 1

Valentyn Boreiko, Zhiqi Bu, and Sheng Zha. Towards understanding of orthogonalization in muon. In *Tiny Titans: The next wave of On-Device Learning for Foundational Models (TTODLer-FM)*. 20

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html. 27

Zhiqi Bu, Shiyun Xu, and Jialin Mao. Convex dominance in deep learning i: A scaling law of loss and learning rate. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dSdLqg02tx. 1, 3, 7, 8

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam

Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling Language Modeling with Pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. URL https://jmlr.org/papers/v24/22-1144.html. 27

DeepSeek-AI. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf. Technical report. 27

Aaron Defazio, Ashok Cutkosky, Harsh Mehta, and Konstantin Mishchenko. Optimal linear decay learning rate schedules and further refinements. *arXiv preprint arXiv:2310.07831*, 2023. 1, 3, 14

Nolan Dey, Bin Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: Completep enables compute-efficient deep transformers. *Advances in Neural Information Processing Systems*, 38:137707–137739, 2026. 1

Varun Godbole, George E. Dahl, Justin Gilmer, Christopher J. Shallue, and Zachary Nado. Deep learning tuning playbook, 2023. URL http://github.com/google-research/tuning_playbook. Version 1.0. 1

Noah Golmant, Nikita Vemuri, Zhewei Yao, Vladimir Feinberg, Amir Gholami, Kai Rothauge, Michael W Mahoney, and Joseph Gonzalez. On the computational inefficiency of large batch sizes for stochastic gradient descent. *arXiv preprint arXiv:1811.12941*, 2018. 1, 4

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 1, 5, 21

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 27

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022a. 1

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training Compute-Optimal Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35, 2022b. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf. 8, 27

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024. 1

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 1

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017. 1

Kimi Team. Kimi K2: Open Agentic Intelligence, 2025. URL https://arxiv.org/abs/2507.20534. 27

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. What matters when building vision-language models?, 2024. 8

Houyi Li, Wenzhen Zheng, Jingcheng Hu, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Shuigeng Zhou, Xiangyu Zhang, et al. Predictable scale: Part i–optimal hyperparameter scaling law in large language model pretraining. *arXiv e-prints*, pp. arXiv–2503, 2025. 1

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*. 20

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018. 1, 4, 5, 21

William Merrill, Shane Arora, Dirk Groeneveld, and Hannaneh Hajishirzi. Critical batch size revisited: A simple empirical approach to large-batch language model training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. 4

Alexandru Meterez, Depen Morwani, Jingfeng Wu, Costin-Andrei Oncescu, Cengiz Pehlevan, and Sham M. Kakade. Seesaw: Accelerating training by balancing learning rate and batch size scheduling, 2025. 1

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math, 2024. 9

Shikai Qiu, Lechao Xiao, Andrew Gordon Wilson, Jeffrey Pennington, and Atish Agarwala. Scaling collapse reveals universal dynamics in compute-optimally trained neural networks. In *International Conference on Machine Learning*, pp. 50697–50720. PMLR, 2025. 8

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling Language Models: Methods, Analysis & Insights from Training Gopher, 2021. URL https://arxiv.org/abs/2112.11446. 27

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16, 2020. 1

Fabian Schaipp, Alexander Hägele, Adrien Taylor, Umut Simsekli, and Francis Bach. The surprising agreement between convex optimization theory and learning-rate scheduling for large model training. In *International Conference on Machine Learning*, 2025. 1, 3

Christopher J Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20(112):1–49, 2019. 1

Xian Shuai, Yiding Wang, Yimeng Wu, Xin Jiang, and Xiaozhe Ren. Scaling law for language models training considering batch size. *arXiv preprint arXiv:2412.01505*, 2024. 1

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1Yy1BxCZ. 1, 5, 21

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022. 27

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388. 9

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

Luis Wiedmann, Aritra Roy Gosthipaty, and Andrés Marafioti. nanovlm. `https://github.com/huggingface/nanoVLM`, 2025. 8

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 27

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097, 2021. 1

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite depth neural networks. In *International Conference on Learning Representations*, volume 2024, pp. 55099–55150, 2024. 1

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Peng Zhang, Yuxiao Dong, and Jie Tang. GLM-130B: An Open Bilingual Pre-trained Model. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=-Aw0rrrPUF`. 27

Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean Foster, and Sham M Kakade. How does critical batch size scale in pre-training? In *The Thirteenth International Conference on Learning Representations*. 1

# A THEORETICAL RESULTS

## A.1 THEOREM 1

*Proof of Theorem 1.* Theorem 10 in Defazio et al. (2023) gives an upper bound of the loss

$$\mathbb{E}L(\mathbf{w}_\tau) \leq L_* + \frac{D^2}{2\sum_{t=1}^{T}\eta_t} + \frac{1}{2}\left(\frac{\sum_{t=1}^{\tau}\eta_t^2\mathbb{E}\|\mathbf{g}_t\|^2}{\sum_{t=1}^{\tau}\eta_t} + \sum_{k=1}^{\tau-1}\frac{\eta_k}{\sum_{t=k+1}^{\tau}\eta_t}\frac{\sum_{t=k}^{\tau}\eta_t^2\mathbb{E}\|\mathbf{g}_t\|^2}{\sum_{t=k}^{\tau}\eta_t}\right) \tag{A.1}$$

To simplify the notation, we write

$$x_t = \eta_t^2\mathbb{E}\|\mathbf{g}_t\|^2, A_k = \frac{\eta_k}{\sum_{t=k+1}^{\tau}\eta_t\sum_{t=k}^{\tau}\eta_t} = \frac{1}{\sum_{t=k+1}^{\tau}\eta_t} - \frac{1}{\sum_{t=k}^{\tau}\eta_t}$$

so that equation A.1 becomes

$$\begin{aligned}
&L_* + \frac{D^2}{2\sum_{t=1}^{\tau}\eta_t} + \frac{1}{2}\frac{1}{\sum_{t=1}^{\tau}\eta_t}\sum_{t=1}^{\tau}x_t + \frac{1}{2}\sum_{k=1}^{\tau-1}\left(A_k\sum_{t=k}^{\tau}x_t\right)\\
=&L_* + \frac{D^2}{2\sum_{t=1}^{\tau}\eta_t} + \frac{1}{2}\frac{1}{\sum_{t=1}^{\tau}\eta_t}\sum_{t=1}^{\tau-1}x_t + \frac{1}{2}\frac{x_\tau}{\sum_{t=1}^{\tau}\eta_t} + \frac{1}{2}\sum_{k=1}^{\tau-1}\left(A_k\sum_{t=k}^{\tau}x_t\right)\\
=&L_* + \frac{D^2}{2\sum_{t=1}^{\tau}\eta_t} + \frac{1}{2}\frac{1}{\sum_{t=1}^{\tau}\eta_t}\sum_{t=1}^{\tau-1}x_t + \frac{1}{2}x_\tau\frac{1}{\sum_{t=1}^{\tau}\eta_t} + \frac{1}{2}x_\tau\sum_{k=1}^{\tau-1}A_k + \frac{1}{2}\sum_{k=1}^{\tau-1}\left(A_k\sum_{t=k}^{\tau-1}x_t\right)
\end{aligned}$$

We exchange the double sum to yield $\sum_{k=1}^{\tau-1}\left(A_k\sum_{t=k}^{\tau-1}x_t\right) = \sum_{t=1}^{\tau-1}\left(x_t\sum_{k=1}^{t}A_k\right)$, and obtain

$$L_* + \frac{D^2}{2\sum_{t=1}^{\tau}\eta_t} + \frac{1}{2}x_\tau\left(\frac{1}{\sum_{s=1}^{\tau}\eta_s} + \sum_{k=1}^{\tau-1}A_k\right) + \frac{1}{2}\sum_{t=1}^{\tau-1}x_t\left(\frac{1}{\sum_{s=1}^{\tau}\eta_s} + \sum_{k=1}^{t}A_k\right)$$

which can be further simplified to

$$L_* + \frac{D^2}{2\sum_{t=1}^{\tau}\eta_t} + \frac{1}{2}\frac{x_\tau}{\eta_\tau} + \frac{1}{2}\sum_{t=1}^{\tau-1}x_t\frac{1}{\sum_{s=t+1}^{\tau}\eta_s}$$

because the telescoping sum gives $\sum_{k=1}^{t}A_k = \frac{1}{\sum_{s=t+1}^{\tau}\eta_s} - \frac{1}{\sum_{s=1}^{\tau}\eta_s}$. □

## A.2 EQUATION (2.2)

According to the mean-covariance decomposition,

$$\mathbb{E}\|\mathbf{g}_t\|^2 = \|\mathbb{E}\mathbf{g}_t\|^2 + \text{tr}(\text{Cov}(\mathbf{g}_t)). \tag{A.2}$$

Let $G$ be the upper bound of $\|\mathbb{E}\mathbf{g}_t\|$ and $X$ be the upper bound of trace of $\Sigma_t := Cov_\xi(g(\mathbf{w}_t;\xi))$ which is the per-example stochastic gradient at iteration $t$, where $\xi$ is a data sample drawn from the data distribution. Since

$$Cov(\mathbf{g}_t) = Cov\left(\frac{1}{B_t}\sum_{i=1}^{B_t}g(w_t;\xi_{t,i})\right) = \frac{1}{B_t}\Sigma_t,$$

it gives

$$tr(Cov(\mathbf{g}_t)) = tr(\frac{1}{B_t}\Sigma_t) \leq \frac{X}{B_t}.$$

Therefore,

$$\mathbb{E}\|\mathbf{g}_t\|^2 = \|\mathbb{E}\mathbf{g}_t\|^2 + tr(Cov(\mathbf{g}_t)) \leq G^2 + \frac{X}{B_t}. \tag{A.3}$$

**Remark 2.** *For adaptive gradient descent $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t\phi(\mathbf{g}_t)$, suppose some upper bounds $G, X$, then we obtain*

$$\mathbb{E}\|\phi(\mathbf{g}_t)\|^2 \leq G^2 + X/B_t + O_p(1/B_t^2)$$

### A.3 THEOREM 3

*Proof of Theorem 3.* Using the notation of $b_t$, we rewrite the minimization problem equation 3.3 as

$$\min_{B_t} \int_0^T \frac{b_t^2}{B_t}\, dt \text{ s.t.} \int_0^T B_t\, dt = K.$$

The Lagrangian functional associated with the equality constraint is, with $\lambda \in \mathbb{R}$,

$$\int_0^T \frac{b_t^2}{B_t} dt + \lambda \left( \int_0^T B_t dt - K \right) = \int_0^T \left( \frac{b_t^2}{B_t} + \lambda B_t \right) dt - \lambda K.$$

Denote the integrand as $\ell_t(B_t) := \frac{b_t^2}{B_t} + \lambda B_t$. The first-order condition gives

$$\frac{\partial \ell_t}{\partial B_t} = -\frac{b_t^2}{B_t^2} + \lambda = 0 \Longrightarrow B_t^{\text{optim}} = \frac{b_t}{\sqrt{\lambda}},$$

Next, we determine $\lambda$ from the constraint:

$$\int_0^T B_t^{\text{optim}}\, dt = \frac{1}{\sqrt{\lambda}} \int_0^T b_t\, dt = K \Longrightarrow \frac{1}{\sqrt{\lambda}} = \frac{K}{\int_0^T b_t\, dt} \Longrightarrow B_t^{\text{optim}} = K \cdot \frac{b_t}{\int_0^T b_t\, dt}.$$

To make $B_t^{\text{optim}}$ explicit on $\eta_t$, notice that

$$\int_0^T b_t dt = 2\sqrt{\int_0^T \eta_k dk}.$$

□

### A.4 COROLLARY 2

*Proof of Corollary 2.* Denoting $Z_t := \frac{\eta_t}{\sqrt{\int_t^T \eta_k dk}}$, plain Calculation shows:

$$L_{\text{dynamic}}(T) - L_{\text{static}}(T) = \frac{XT}{2B_{\text{static}}} \left[ \left( \frac{\int_0^T Z_t dt}{T} \right)^2 - \frac{\int_0^T Z_t^2 dt}{T} \right].$$

Thus $L_{\text{dynamic}}(T) - L_{\text{static}}(T) \leq 0$ via Jensen's inequality, or Cauchy-Schwarz inequality applied to $Z_t$ and 1. □

### A.5 TABLE 2 AND TABLE 3

Here we derive $b_t$, $B_t^{\text{optim}}$ and $L(T)$. We omit the optimal $L(T)$ because it is trivial to minimize the formula $L_* + a/\eta + b \cdot \eta$.

**constant learning rate** We start from $\eta_t = \eta$. Firstly, we write

$$b_t := \frac{\eta_t}{\sqrt{\int_t^T \eta_s ds}} = \sqrt{\frac{\eta}{T-t}}$$

Secondly, we get

$$\int_0^T b_t dt = 2\sqrt{T\eta}$$

Thirdly, we get

$$B_t = BT \cdot \frac{b_t}{\int_0^T b_t dt} = \frac{BT}{2\sqrt{T(T-t)}}$$

Lastly, substituting $B_t$ and $\eta_t$ into the loss bound:

$$L(T) = L_* + \frac{D^2}{2T\eta} + \frac{\eta}{2} G^2 \ln T + \frac{2\eta X}{B}$$

**linear learning rate** We start from $\eta_t = \eta(1 - t/T)$. Firstly, we write

$$b_t := \frac{\eta_t}{\sqrt{\int_t^T \eta_s ds}} = \frac{\eta(1 - \frac{t}{T})}{\sqrt{\frac{(T-t)^2}{2T}}} = \sqrt{\frac{2}{T}}$$

Secondly, we get

$$B_t = B$$

Lastly, substituting $B_t$ and $\eta_t$ into the loss bound:

$$L_* + \frac{D^2}{T\eta} + \eta(G^2 + X/B)$$

**WSD** We start from $\eta_t = \begin{cases} \eta & \text{if } t \leq cT \\ \eta \frac{T-t}{T-cT} & \text{if } t > cT \end{cases}$.

Firstly, we write

$$b_t := \frac{\eta_t}{\sqrt{\int_t^T \eta_s ds}}$$

where

$$\int_t^T \eta_s ds = \begin{cases} \frac{\eta}{2}(T + cT - 2t) & \text{if } t \leq cT \\ \frac{\eta}{2}\frac{(T-t)^2}{(T-cT)} & \text{if } t > cT \end{cases}$$

so that

$$b_t = \begin{cases} \sqrt{\frac{2\eta}{T+cT-2t}} & \text{if } t \leq cT \\ \sqrt{\frac{2\eta}{T-cT}} & \text{if } t > cT \end{cases}$$

Secondly, we get

$$\int_0^T b_t dt = \sqrt{2\eta(T + cT)}$$

Thirdly, we get

$$B_t = BT \cdot \frac{b_t^*}{\int_0^T b_t^* dt} = \begin{cases} \frac{BT}{\sqrt{(T+cT-2t)(T+cT)}} & \text{if } t \leq cT \\ \frac{BT}{\sqrt{(T-cT)(T+cT)}} = \frac{B}{\sqrt{1-c^2}} & \text{if } t > cT \end{cases}$$

Lastly, substituting $B_t$ and $\eta_t$ into the loss bound:

$$L_* + \frac{D^2}{\eta(1+c)T} + \frac{\eta}{2}\left[2 + \ln\frac{1+c}{1-c}\right]G^2 + \eta(1+c)\frac{X}{B}.$$

**cosine learning rate** We start from $\eta_t = \frac{\eta}{2}(1 + \cos(\frac{\pi t}{T}))$.

Firstly, we write

$$b_t := \frac{\eta_t}{\sqrt{\int_t^T \eta_s ds}} = \cos^2\left(\frac{\pi t}{2T}\right)\sqrt{\frac{\eta}{\frac{T}{2}\left(1 - \frac{t}{T} - \frac{1}{\pi}\sin\frac{\pi t}{T}\right)}}.$$

Secondly, we get

$$B_t = BT \cdot \frac{b_t}{\int_0^T b_t dt} = \frac{B\cos^2\left(\frac{\pi t}{2T}\right)}{\sqrt{1 - \frac{t}{T} - \frac{1}{\pi}\sin\frac{\pi t}{T}}},$$

Thirdly, substituting $B_t$ and $\eta_t$ into the loss bound:

$$L_* + \frac{D^2}{T\eta} + 1.061\eta G^2 + \frac{\eta X}{B}$$

### A.6 LEMMA 1

*Proof of Lemma 1.* Similar to Theorem 3, using the notation of $b_t$, we rewrite the minimization problem equation 3.5 as

$$\min_{B_t} \int_0^T \frac{b_t^2}{B_t}\, dt \text{ s.t.} \int_0^T f(B_t)\, dt = K.$$

The Lagrangian functional associated with the equality constraint is, with $\lambda \in \mathbb{R}$,

$$\int_0^T \frac{b_t^2}{B_t} dt + \lambda \left( \int_0^T f(B_t) dt - K \right) = \int_0^T \left( \frac{b_t^2}{B_t} + \lambda f(B_t) \right) dt - \lambda K.$$

Denote the integrand as $\ell_t(B_t) := \frac{b_t^2}{B_t} + \lambda f(B_t)$. The first-order condition gives

$$\frac{\partial \ell_t}{\partial B_t} = -\frac{b_t^2}{B_t^2} + \lambda f'(B_t) = 0 \Longrightarrow g^{-1}(B_t) = B_t^2 f'(B_t) = \frac{b_t^2}{\lambda} \Longrightarrow B_t^* = g\left(\frac{b_t^2}{\lambda}\right),$$

Next, we determine $\lambda$ from the constraint:

$$\int_0^T f(B_t^*)\, dt = K \Longrightarrow \int_0^T f\left( g\left( \frac{\eta_t^2}{\int_t^T \eta_s ds} / \lambda \right) \right) dt = K \Longrightarrow B_t^* = g\left( \frac{\eta_t^2}{\int_t^T \eta_s ds} / \lambda \right)$$

□

### A.7 THEOREM 4

*Proof of Theorem 4.* Denote $L_\kappa(\mathbf{w}) = L(\mathbf{w}) + \frac{\kappa}{2}\|\mathbf{w}\|^2$, then SGD with weight decay on $L$ is equivalent to SGD without weight decay on $L_\kappa$, where the bounded gradient condition becomes $\nabla L_\kappa = \nabla L + \kappa \mathbf{w} \Longrightarrow \|\nabla L_\kappa\| = \|\nabla L\| + \kappa \|\mathbf{w}\| \le G + \kappa W$.

It is not hard to obtain (c.f. equation 2.6)

$$\mathbb{E} L_\kappa(\mathbf{w}_T) - L_{\kappa,*} \le \frac{D^2}{2\sum_{t=1}^T \eta_t} + \frac{1}{2} \sum_{t=1}^{T-1} \frac{\eta_t^2((G+\kappa W)^2 + X/B_t)}{\sum_{k=t+1}^T \eta_k}$$

as long as the learning rate decays to zero for any fixed $T$.

Notice that weight decay does not impact the last term $X/B_t$, rendering the same minimization problem as equation 3.3 and hence the same solution $B_t^{\text{optim}}$, i.e. our optimal batch size schedule.

For qualified learning rate schedule and scaled peak learning rate, we get $\Theta(1/\sqrt{T})$ loss by Table 3, where the only difference is on $G$. Therefore, under either static batch size or optimal batch size, we get

$$\mathbb{E} L_\kappa(\mathbf{w}_T) - L_{\kappa,*} = O(1/\sqrt{T})$$

To translate the gap between regularized loss $L_\kappa$ to the gap between vanilla loss $L$, we need

$$\mathbb{E} L(\mathbf{w}_T) - L_* \le \mathbb{E} L_\kappa(\mathbf{w}_T) - L_* \le \mathbb{E} L_\kappa(\mathbf{w}_T) - L_{\kappa,*} + L_{\kappa,*} - L_*.$$

The first difference has already been derived. The second difference is obvious by contradiction:

$$L_{\kappa,*} - L_* \le \frac{\kappa}{2}\|\mathbf{w}_*\|^2 \le \frac{\kappa}{2} W^2$$

All in all, we need

$$\mathbb{E} L(\mathbf{w}_T) - L_* \le O(1/\sqrt{T}) + \frac{\kappa}{2} W^2$$

which requires $\kappa = O(1/\sqrt{T})$. □

# B EXPERIMENT DETAILS

## B.1 FIGURE 1, TABLE 1

We train with WSD learning rate schedule and cooldown at 0.9; peak learning rate is 0.02. The optimizer is Muon-NSGD. Default initialization in Huggingface.

Llama3 model has 48 layer, hidden size 3072, intermediate size 3072*4, batch size 256, and sequence length 1024. Qwen3 MoE model has 3B total parameters, 32 layers, hidden size 2048, intermediate size 2048*3, moe intermediate size 2048/4, 16 experts, batch size 512, and sequence length 4096.

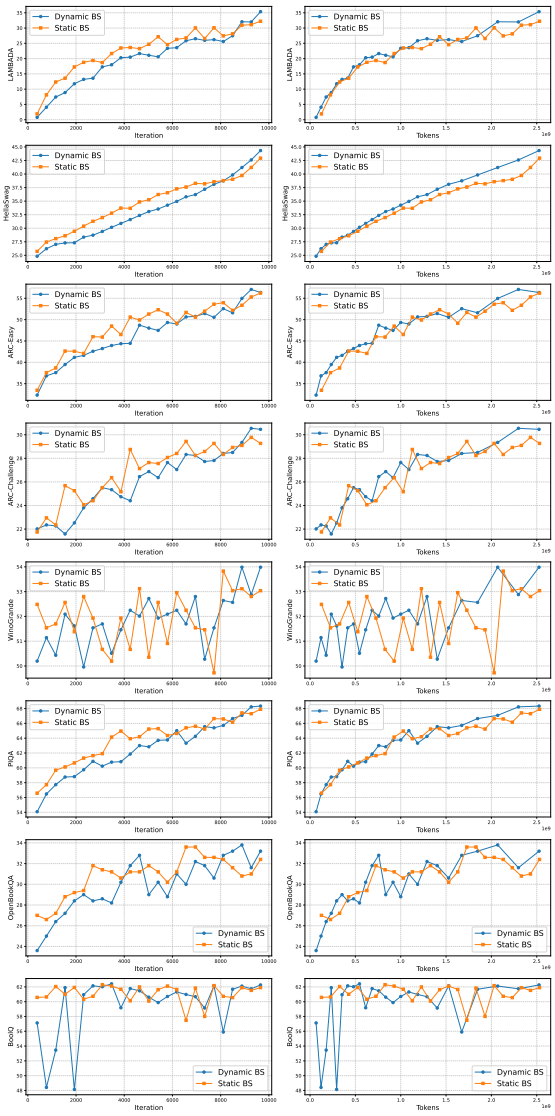



**Figure 9:** Evaluation of intermediate iterates for Llama3-8B with dynamic or static batch size.

## B.2 FIGURE 2, FIGURE 4

Llama3 model has 28 layer, hidden size 1024, intermediate size 3072, batch size 256, and sequence length 1024. Figure 10 is analogous to Figure 2 but uses AdamW optimizer. The losses are smoothed by time weighted EMA.

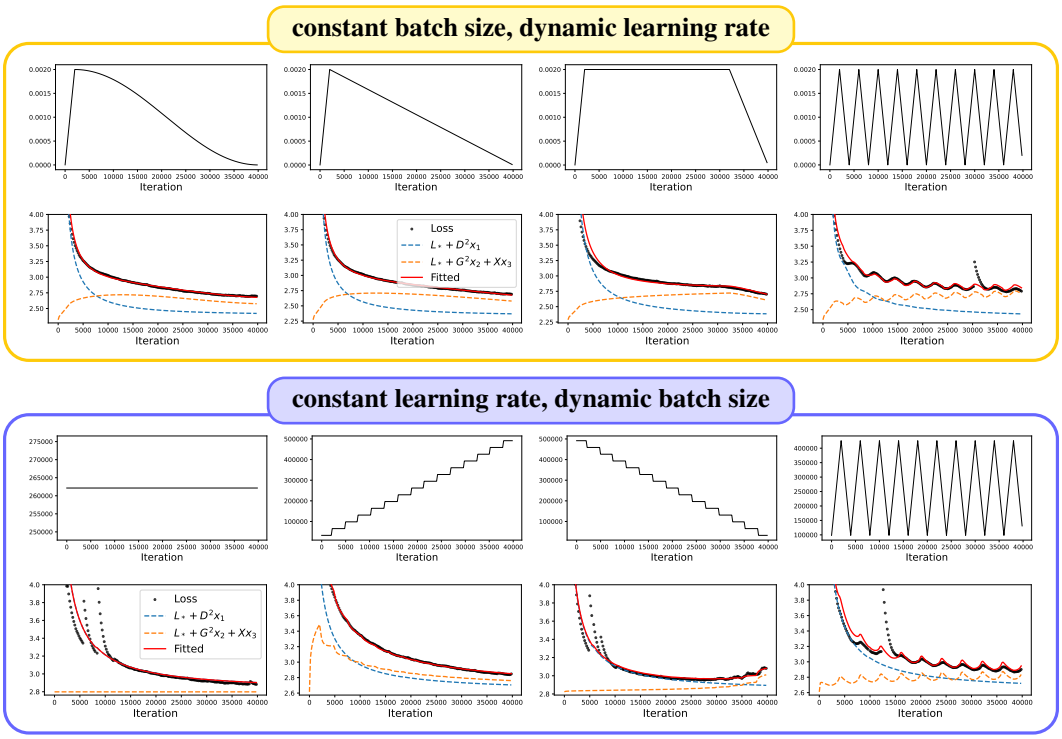



**Figure 10:** Sequence-to-sequence prediction by Theorem 2 for Llama3-1B model on Fineweb-edu data, trained by AdamW optimizer. Top two rows: $B_t$ is fixed ($B = 256 * 1024$) and $\eta_t$ is dynamic. Bottom two rows: $\eta_t$ is fixed ($\eta = 0.002$) and $B_t$ is dynamic.

### B.3 Figure 5, Figure 6

We use WSD learning rate schedule with 10% cooldown and Muon-NSGD optimizer. For Figure 5 (left column), the peak learning rate is 0.01; for other figures, the peak learning rate is scaled as $2/\sqrt{T}$. TPP is based on active parameters for MoE and total parameters for dense models.

For Llama3 models, we experiment with

- total parameters 111M, 8 layers, 512 hidden size
- total parameters 274M, 8 layers, 1024 hidden size
- total parameters 426M, 10 layers, 1280 hidden size
- total parameters 507M, 10 layers, 1440 hidden size
- total parameters 631M, 12 layers, 1536 hidden size
- total parameters 898M, 14 layers, 1792 hidden size
- total parameters 1.236B, 16 layers, 2048 hidden size
- total parameters 2.163B, 20 layers, 2560 hidden size
- total parameters 3.569B, 28 layers, 2880 hidden size
- total parameters 6.585B, 48 layers, 3072 hidden size

The intermediate size is set to 4 times hidden size, batch size 256, and sequence length 1024.

For Qwen3 MoE models, we experiment with

- active parameters 438M, 8 layers, 1280 hidden size
- active parameters 606M, 16 layers, 1536 hidden size

- active parameters 826M, 24 layers, 1792 hidden size
- active parameters 1.127B, 32 layers, 2048 hidden size

The intermediate size is set to 3 times hidden size, MoE intermediate size is 1/4 hidden size, 16 experts, batch size 512, and sequence length 4096.

# C Ablation results

## C.1 Optimizers

Our main optimizer is Muon-NSGD (Boreiko et al.). We additionally experiment on AdamW optimizer (Loshchilov & Hutter) to confirm the applicability of Theorem 2 in Figure 10. We also trained VLM in Section 5.1 with AdamW.

## C.2 Peak learning rate

We have ablated $O(1/\sqrt{T})$ learning rate in Section 4, which is empirically the optimal learning rate. Furthermore, we show in Figure 11 that optimal batch size ***always outperforms*** static batch size for any learning rate, larger or smaller than the optimal learning rate (which is 0.01 in this case). This holds because our batch size derived in Theorem 3 is ***decoupled*** from the peak learning rate.

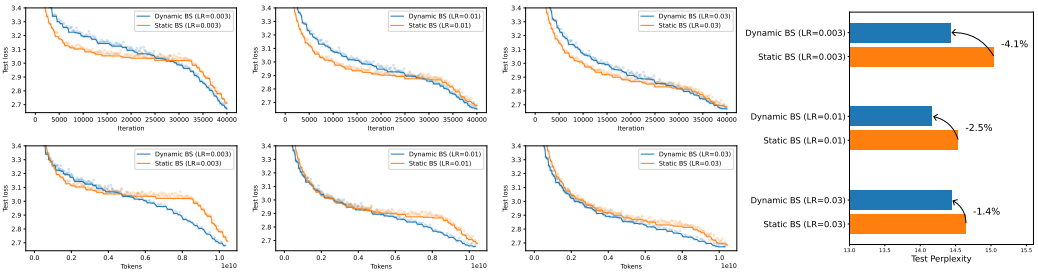



**Figure 11:** Impact of batch size schedules for Llama3 1B model on Fineweb-edu data. Left to right: peak learning rate=0.003, 0.01 (optimal), 0.03.

## C.3 Training precision

By default, we use mixed-precision training. We also observe strong performance of optimal batch size schedule when using full-precision training. However, with half-precision training, the advantage of our batch size vanishes in comparison to the static batch size.

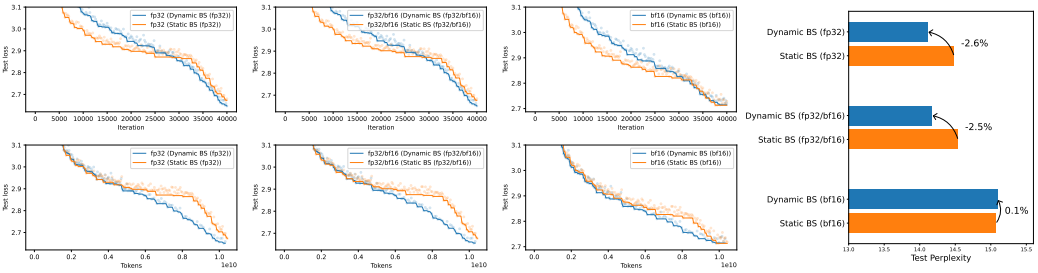



**Figure 12:** Impact of batch size schedules for Llama3 1B model on Fineweb-edu data. Left to right: full-precision (fp32), mixed precision (fp32/bf16), and half-precision (bf16).

## C.4 Base batch size

We show in Figure 13 and Figure 14 that optimal batch size ***always outperforms*** static batch size for any base batch size, under fixed compute and fixed iterations. Note that the advantage vanishes

as $B$ approaches infinity, because the advantage is inversely proportional to base batch size by Corollary 2. Here $B$=1X is 128k tokens.

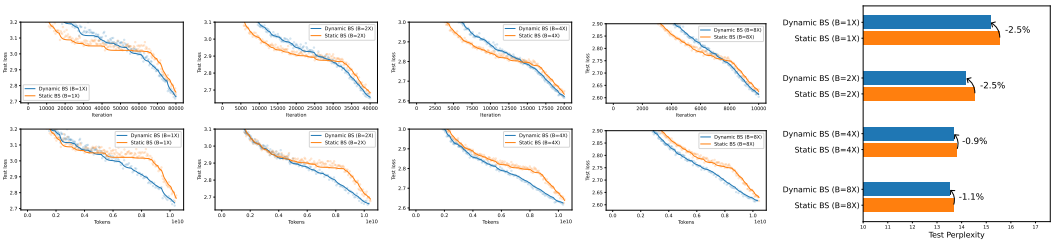



**Figure 13:** Impact of batch size schedules for Llama3 1B model on Fineweb-edu data, under fixed compute (larger batch uses fewer iterations).

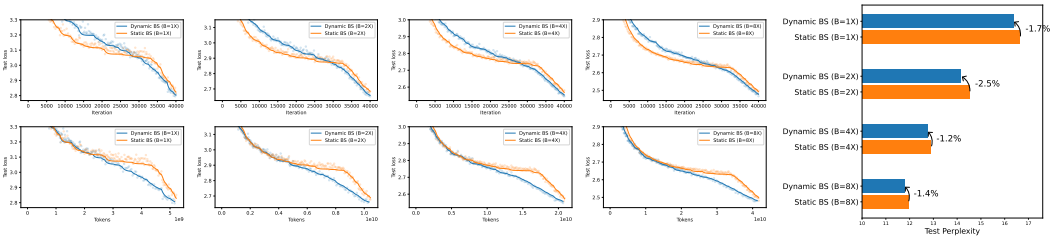



**Figure 14:** Impact of batch size schedules for Llama3 1B model on Fineweb-edu data, under fixed iterations of 40000.

### C.5 The $\eta/B$ ratio does not determine training dynamics

A natural hypothesis, suggested by the linear scaling rule (Goyal et al., 2017; Smith et al., 2018) and the analysis in McCandlish et al. (2018), is that training dynamics are governed by the ratio $\eta/B$ rather than by $\eta$ and $B$ individually. If true, one could freely trade learning rate schedules for batch size schedules as long as $\eta(t)/B(t)$ remains unchanged. We design a controlled experiment to test this hypothesis directly.

**Setup.** We train Qwen3-0.6B from scratch on FineWeb-EDU using AdamW with batch size 256 and sequence length 1024. Each run trains for 10,000 steps, consuming 2.6B tokens. We sweep over three learning rate schedules (WSD, cosine, linear) and three base learning rates ($3 \times 10^{-4}$, $10^{-3}$, $3 \times 10^{-3}$), with $\eta_{\min} = 0.1\,\eta_{\max}$.

**Design.** For each (schedule, $\eta_{\max}$) pair, we run two configurations that produce the *same* $\eta/B$ trajectory at every step:

- **Type 1 (LR schedule):** $\eta(t)$ follows the schedule, $B = G$ is constant. The ratio is $\eta(t)/G$.
- **Type 2 (BS schedule):** $\eta = \eta_{\text{const}}$ is fixed, $B(t) \propto 1/\eta_{\text{ref}}(t)$ so that $\eta_{\text{const}}/B(t) = \eta_{\text{ref}}(t)/G$. Here $\eta_{\text{const}} = T/\sum_{t=0}^{T-1} 1/\eta_{\text{ref}}(t)$ is chosen to match the total token budget exactly.

Both configurations consume identical total tokens by construction. If $\eta/B$ is the governing quantity, Type 1 and Type 2 should yield the same loss.

**Results.** Figure 15 and Figure 16 show the validation loss curves versus iteration and total tokens, respectively. Across all 9 configurations, the LR-schedule run (Type 1, blue) consistently outperforms the BS-schedule run (Type 2, orange), often by a significant margin. Table 4 summarizes the final validation loss: the gap ranges from $+0.017$ to $+0.74$ in favor of Type 1. The effect is largest at small learning rates and for cosine/linear schedules, where the BS-schedule run must use a very small constant $\eta_{\text{const}}$ to satisfy the token budget constraint.

These results demonstrate that $\eta/B$ alone does not determine training dynamics. The learning rate schedule provides benefits beyond its effect on the $\eta/B$ ratio—likely through its interaction with the optimizer state and loss landscape curvature.

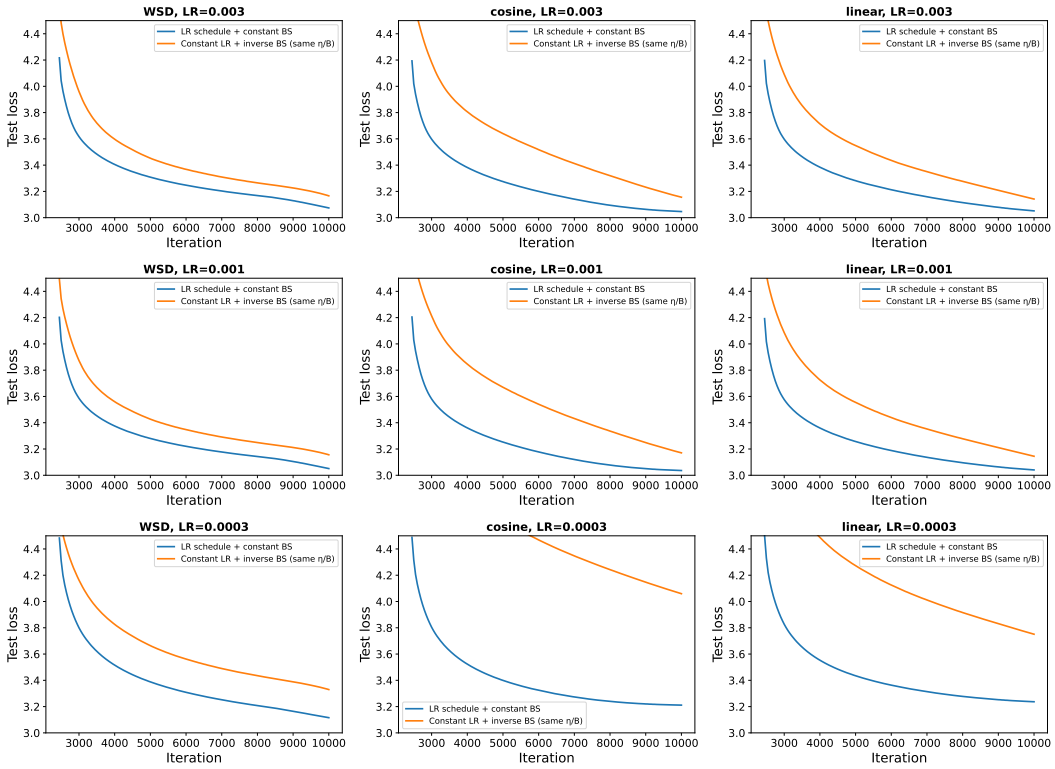



**Figure 15:** Validation loss vs. iteration for Type 1 (LR schedule + constant BS, blue) and Type 2 (constant LR + inverse BS, orange), which share the same $\eta/B$ trajectory. Columns: WSD, cosine, and linear schedules. Rows top to bottom: $\eta_{\max} = 3 \times 10^{-3}$, $10^{-3}$, $3 \times 10^{-4}$. The LR-schedule run consistently achieves lower loss despite identical $\eta/B$ at every step. Curves smoothed with a 50-iteration running average.

**Table 4:** Final validation loss for Type 1 (LR schedule) vs. Type 2 (BS schedule) across all configurations. $\Delta$ is Type 2 minus Type 1; positive values indicate the LR schedule wins.

| $\eta_{\max}$ | Schedule | Type 1 (LR sched.) | Type 2 (BS sched.) | $\Delta$ |
|---|---|---|---|---|
| $3 \times 10^{-3}$ | WSD | 3.015 | 3.033 | +0.019 |
| $3 \times 10^{-3}$ | Cosine | 3.050 | 3.082 | +0.031 |
| $3 \times 10^{-3}$ | Linear | 3.041 | 3.059 | +0.017 |
| $10^{-3}$ | WSD | 2.995 | 3.035 | +0.039 |
| $10^{-3}$ | Cosine | 3.042 | 3.093 | +0.051 |
| $10^{-3}$ | Linear | 3.035 | 3.063 | +0.028 |
| $3 \times 10^{-4}$ | WSD | 3.077 | 3.212 | +0.136 |
| $3 \times 10^{-4}$ | Cosine | 3.221 | 3.964 | +0.743 |
| $3 \times 10^{-4}$ | Linear | 3.239 | 3.655 | +0.416 |

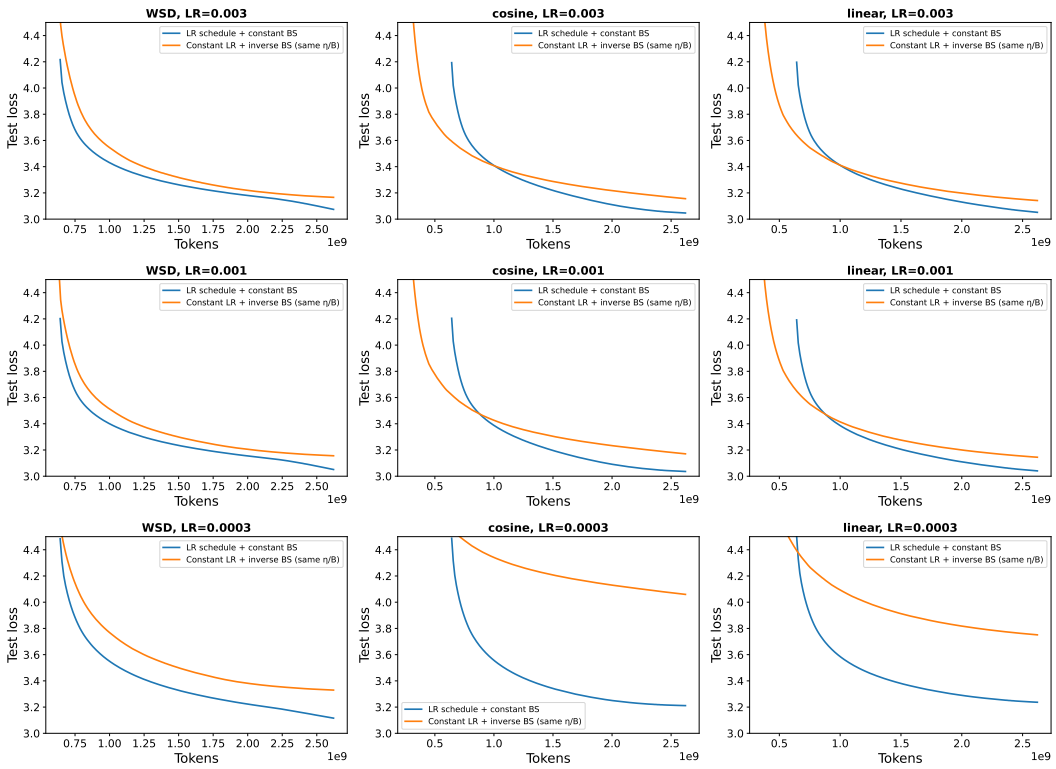



**Figure 16:** Same comparison as Figure 15, plotted against total tokens consumed. Since both types consume exactly the same number of tokens, the x-axes are aligned. The LR-schedule advantage persists when controlling for compute budget.

## D FURTHER EXPERIMENT DETAILS

Table 5 summarizes the hyperparameters used in the post-training experiments of Section 5.

**Table 5:** Hyperparameters for the VLM and math fine-tuning experiments in Section 5. For the LR sweep, we train with constant batch size and select the best LR by validation loss and downstream accuracy, then fix it for the dynamic-vs.-static comparison.

| Hyperparameter | VLM | Math |
|---|---|---|
| Model | SmoLM2+Siglip2 | Qwen3-0.6B |
| Dataset | Cauldron | OrcaMath |
| Optimizer | Muon-NSGD | AdamW |
| LR schedule | WSD | WSD |
| Peak LR | 2e-3 | 1e-5 |
| Min LR | 0 | 0 |
| Weight decay | 0.01 | 0.01 |
| Context length | 4096 | 512 |
| Base batch size | 64 | 32 |
| Total steps | 40000 | 500 |

Table 6 reports the absolute scores for VLM benchmark tasks under dynamic and static batch size schedules.

**Table 6:** Benchmark scores for dynamic and static batch sizes on VLM evaluation (§5.1). Bold text indicates the better value per benchmark.

| Benchmark | Dynamic | Static |
|---|---|---|
| MMMU (acc) | **0.2411** | 0.2311 |
| ScienceQA | **0.3954** | 0.3924 |
| TextVQA | **0.3556** | 0.3397 |
| DocVQA | **0.1847** | 0.1704 |
| InfoVQA | **0.1099** | 0.1067 |
| OCRBench | **0.2810** | 0.2720 |
| MME-Perception | **704.03** | 653.82 |
| MME-Cognition | 169.29 | **171.79** |
| MMStar (average) | 0.2361 | **0.2377** |
| Coarse Perception | 0.2782 | **0.2819** |
| Fine-grained Perception | 0.2287 | **0.2317** |
| Instance Reasoning | **0.2457** | 0.2392 |
| Logical Reasoning | 0.2335 | **0.2341** |
| Math | 0.2082 | **0.2369** |
| Science | **0.2222** | 0.2024 |

## E How existing paper use batch size?

In this section we summarize the learning rate and batch size schedules used in training recent open source models. We summarize the schedules each model uses in Table 7, also visualize them in Figure 17 and 18. It could be seen that, most of the recent models are using a non-constant batch-size schedule, however they mostly employ a step function batch size schedule heuristically.

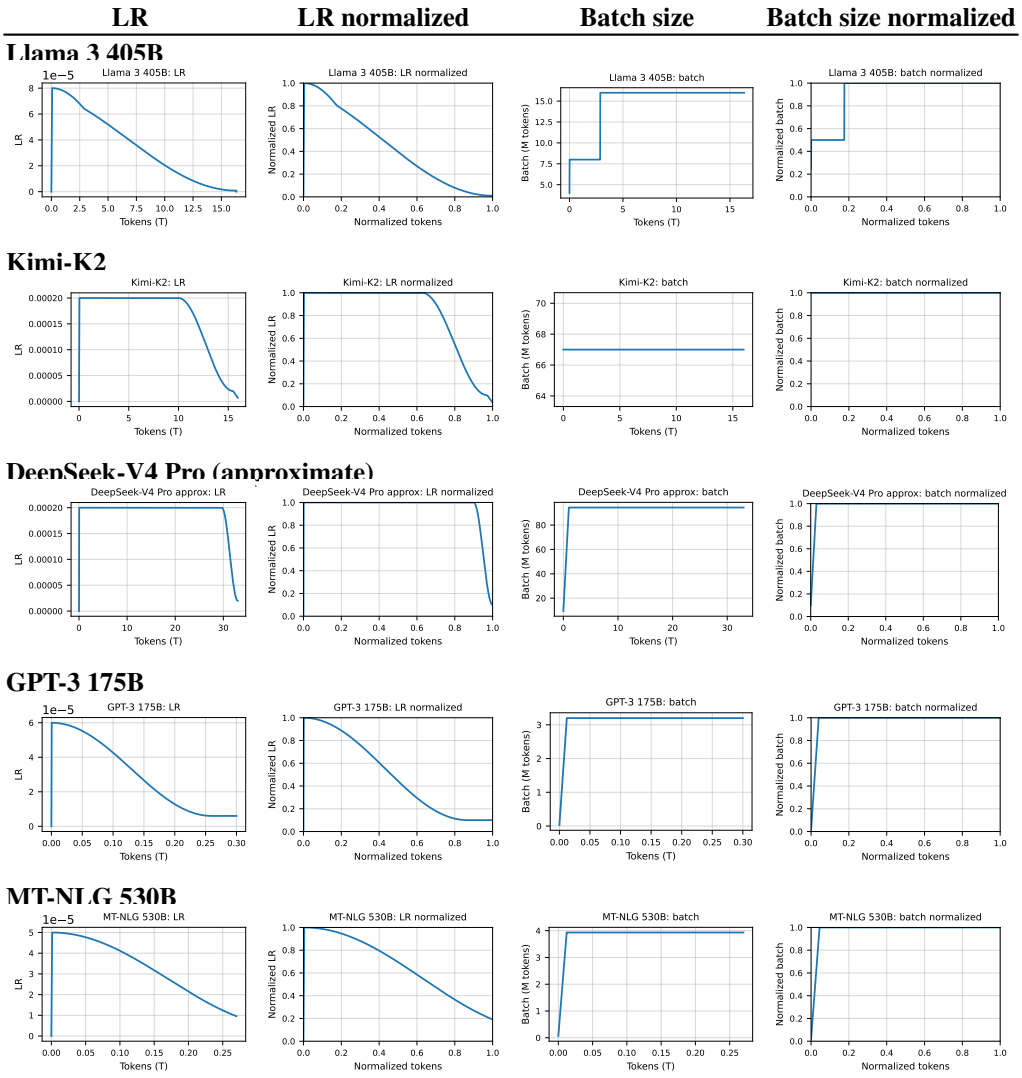



**Figure 17:** Learning-rate and batch-size schedules used in large-scale language model pre-training, part 1. The data are collected from the corresponding existing publications specified in Table 7. Each row corresponds to one model. The four columns show, from left to right: learning rate in original coordinates, learning rate in normalized coordinates, batch size in original coordinates, and batch size in normalized coordinates. In normalized plots, both axes are scaled to lie in $[0, 1]$.

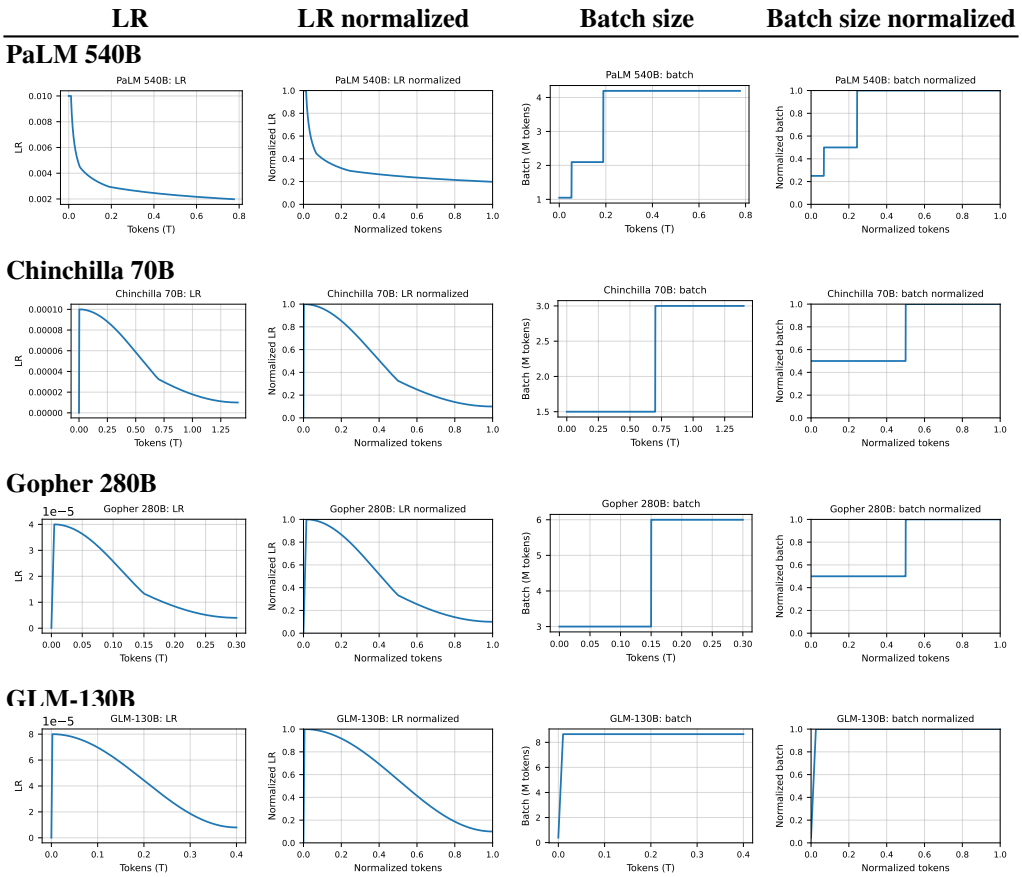



**Figure 18:** Learning-rate and batch-size schedules used in large-scale language model pre-training, part 2. The data are collected from the corresponding existing publications specified in Table 7. Each row corresponds to one model. The four columns show, from left to right: learning rate in original coordinates, learning rate in normalized coordinates, batch size in original coordinates, and batch size in normalized coordinates. In normalized plots, both axes are scaled to lie in $[0, 1]$.

| Model | Learning-rate schedule | Batch-size schedule | Where specified |
|---|---|---|---|
| Llama 3 405B | Peak LR $8 \times 10^{-5}$. Linear warmup for 8,000 steps; cosine decay to $8 \times 10^{-7}$ over 1.2M steps; final long-context annealing linearly decays LR to 0 over the last 40M tokens. | 4M tokens/update initially; 8M after 252M tokens; 16M after 2.87T tokens. | (Grattafiori et al., 2024, Sec. 3.4, pp. 14–15). |
| Qwen3 / Qwen3.5 / Qwen3.6 | Numeric LR schedule not disclosed in the public report/pages I found. Qwen3 states that predicted optimal LR/batch strategies are obtained by scaling laws and that LR decay is accelerated in Stage 2. | Numeric batch-size schedule not disclosed. Publicly stated stages: $>$ 30T tokens at 4K context, then $\sim$ 5T higher-quality tokens at 4K, then hundreds of billions of tokens at 32K. | (Yang et al., 2025, Sec. 3.2, p. 4). |
| DeepSeek-V4 Flash / Pro | Flash: 2,000-step linear warmup; LR $2.7 \times 10^{-4}$ for most training; near the end, cosine decay to $2.7 \times 10^{-5}$. Pro: same strategy, peak $2.0 \times 10^{-4}$, end $2.0 \times 10^{-5}$. Exact decay-start token count not disclosed. | Flash: batch size increases from a small size to 75.5M tokens/update, then stays there. Pro: maximum batch 94.4M tokens/update. Exact ramp endpoint not disclosed. | (DeepSeek-AI, 2026, Sec. 4.2.2, pp. 24–25). |
| Kimi-K2 | MuonClip with WSD. 500-step warmup to $2 \times 10^{-4}$, i.e. about $500 \times 67\text{M} = 33.5\text{B}$ tokens; constant $2 \times 10^{-4}$ until 10T tokens; cosine decay to $2 \times 10^{-5}$ over the next 5.5T tokens; terminal/long-context phase decays from $2 \times 10^{-5}$ to $7 \times 10^{-6}$, with decay shape not explicitly specified. | Constant 67M tokens/update. | (Kimi Team, 2025, Sec. 2.5, p. 9). |
| GPT-3 175B | Peak LR $0.6 \times 10^{-4} = 6 \times 10^{-5}$. Linear warmup for 375M tokens; cosine decay to 10% of peak over 260B tokens; constant at 10% of peak through 300B tokens. | Final batch 3.2M tokens/update. Batch linearly increases from 32K tokens/update to final size over the first 4–12B tokens, depending on model size; the 175B-specific ramp length is not separately stated. | (Brown et al., 2020, Table 2.1, p. 8; App. B, p. 42). |
| MT-NLG 530B | Peak LR $5 \times 10^{-5}$. Linear warmup for 1B tokens; cosine decay targeting 10% of peak over 340B tokens. Training stops at 270B tokens, so the LR does not reach the 10% target. | Batch starts at 32 sequences and increases by 32 sequences until 1,920 sequences over first 12B tokens. With sequence length 2,048, this is 65,536 $\rightarrow$ 3.93M tokens/update. | (Smith et al., 2022, Sec. 3.2, p. 10). |
| PaLM 540B | Adafactor. LR $10^{-2}$ for first 10,000 steps, then inverse-square-root decay proportional to $1/\sqrt{k}$, where $k$ is the step number. | Sequence length 2,048. Batch size 512 until step 50K, 1,024 until step 115K, and 2,048 until step 255K; equivalently about 1.05M, 2.10M, and 4.19M tokens/update. | (Chowdhery et al., 2023, Sec. 5, p. 10). |
| Chinchilla 70B | Peak LR $1 \times 10^{-4}$. Uses Gopher training setup except listed changes: warmup from $10^{-7}$ to peak over 1,500 steps, followed by $10\times$ cosine decay. | Batch 1.5M $\rightarrow$ 3M tokens/update, doubled midway. Since Chinchilla trains on 1.4T tokens, this is about 700B tokens at each batch size. | (Hoffmann et al., 2022b, Sec. 4.1/Table 4, p. 9); (Rae et al., 2021, Sec. 3.2, p. 6). |
| Gopher 280B | Peak LR $4 \times 10^{-5}$. Warmup from $10^{-7}$ to peak over first 1,500 steps, followed by $10\times$ cosine decay. | Batch 3M $\rightarrow$ 6M tokens/update. The Chinchilla paper states the batch is doubled midway; for 300B training tokens this gives roughly 150B tokens at each batch size. | (Rae et al., 2021, Sec. 3.1–3.2/Table 1, p. 6); (Hoffmann et al., 2022b, Table 4, p. 9). |
| GLM-130B | Peak LR $8 \times 10^{-5}$. Warmup from $10^{-7}$ to peak over first 0.5% of samples, then $10\times$ cosine decay to $8 \times 10^{-6}$. For 400B tokens, $0.5\% \approx 2$B tokens. | Sequence length 2,048. Batch warmup from 192 to 4,224 sequences over first 2.5% of samples, then constant; equivalently 0.393M $\rightarrow$ 8.65M tokens/update, with the ramp lasting about 10B tokens. | (Zeng et al., 2023, p. 5; Table 11, p. 48). |

**Table 7:** Pre-training learning-rate and batch-size schedules. Batch size is expressed as tokens per optimizer update unless otherwise noted.