OPTIMAL SCALING NEEDS OPTIMAL NORM

Oleg Filatov, Jiangtao Wang, Jan Ebert, Stefan Kesselheim
Jülich Supercomputing Centre
Forschungszentrum Jülich
{o.filatov,jian.wang,ja.ebert,s.kesselheim}@fz-juelich.de

ABSTRACT

Despite recent progress in optimal hyperparameter transfer under model and dataset scaling, no unifying explanatory principle has been established. Using the Scion optimizer, we discover that *joint* optimal scaling across model and dataset sizes is governed by a single invariant: the operator norm of the output layer. Across models with up to 1.3B parameters trained on up to 138B tokens, the optimal learning rate/batch size pair (η^*, B^*) consistently has the same operator norm value – a phenomenon we term *norm transfer*. This constant norm condition is necessary but not sufficient: while for each dataset size, multiple (η, B) reach the optimal norm, only a unique (η^*, B^*) achieves the best loss. As a sufficient condition, we provide the first measurement of (η^*, B^*) scaling with dataset size for Scion, and find that the scaling rules are consistent with those of the Adam optimizer. Tuning per-layer-group learning rates also improves model performance, with the output layer being the most sensitive and hidden layers benefiting from lower learning rates. We provide practical insights on norm-guided optimal scaling and release our Distributed Scion (`Disco`) implementation with logs from over two thousand runs to support research on LLM training dynamics at scale.

1 INTRODUCTION

Recent advancements in the domain of Large Language Models (LLMs) have been largely driven by the principle of scale. Increasing model size and training dataset volume consistently yields more capable systems (Hoffmann et al., 2022; Kaplan et al., 2020), yet at an increasing computational cost. Consequently, achieving *optimal scaling* — a training regime with hyperparameters being optimally configured at various scales — becomes a challenging but necessary step to push the model frontier further.

To address the challenge of hyperparameter tuning, several powerful yet disparate methods have emerged. Theoretically grounded frameworks like Maximum Update Parametrization (μP) (Yang et al., 2022) help transfer optimal hyperparameters with model scaling. Meanwhile, empirical scaling laws (Li et al., 2025) provide rules of thumb for setting hyperparameters optimally when theory is absent, such as with dataset size scaling. Yet, these approaches often feel like pieces of a puzzle, with a unifying principle for scaling across *both* model and dataset dimensions remaining elusive.

Recently, an emerging paradigm of norm-based optimization (Bernstein & Newhouse, 2024a; Pethick et al., 2025a) has offered a new lens through which to view training dynamics: it reframes optimization as a process that controls the operator norms of the model's weight matrices and gradient updates. This perspective enables monitoring of intrinsic model properties during training, potentially revealing principles deeper than the loss curve alone. This raises a natural question: **can the norm-based view uncover the missing unifying principle of optimal scaling?**

In this work, we argue that the answer is yes. By tracking and analyzing layer norms across thousands of experiments, we have made several discoveries, summarized below:

- **Unifying invariant for optimal scaling.** The operator norm of the output layer $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\infty}$ (see Definition 2) for the optimal learning rate (η) and batch size (B) configuration remains constant across both model scaling (in width and depth) and dataset scaling (Fig. 1). We refer to this phenomenon as *norm transfer*, and it provides a *necessary*

condition for optimality. However, it is not sufficient, as multiple non-optimal (η, B) pairs can eventually reach the same optimal norm value (Fig. 2a).

- **Scaling rules for the Scion optimizer.** The relationship between optimal learning rate η^*, batch size B, and dataset size D is empirically fitted as $\log_2 \eta^*(B, D) \propto 0.62 \log_2 B - 0.56 \log_2 D$, consistent with the known square-root scaling rules for the Adam optimizer. We further find that the optimal batch size scales as $B^*(D) \propto D^{0.45 \pm 0.07}$, leading to $\eta^*(D) \propto D^{-0.28 \pm 0.07}$. Together, these rules constitute a *sufficient condition for optimality*. For fixed D, one can trade off $\eta^* \leftrightarrow B^*$ via the $\eta \propto \sqrt{B}$ rule within a low-sensitivity region around the optimal norm, without performance loss (Fig. 2b).
- **Optimal per-layer-group learning rate.** Performance can be improved by up to 6% in relative loss through additional per-layer-group tuning. In our setup, the optimal configuration is $\eta_{\text{input}} : \eta_{\text{hidden}} : \eta_{\text{output}} = 1 : 1/8 : 1$, which generalizes across dataset sizes and batch sizes (Fig. 3). We also find the uniform $1 : 1 : 1$ layout to be close to the optimal one. Among layer groups, the output layer is the most sensitive to tuning, with sensitivity decreasing gradually for the hidden layers and then the input layer.
- **Distributed Scion/Muon and experimental logs.** To facilitate further research on large-scale training dynamics, we release `Disco`[1], a distributed implementation of the Scion/Muon optimizer compatible with modern parallelization strategies, along with norm logs[2] from over two thousand training runs conducted for this study.

2 Methodology

2.1 Background & Terminology

Recently, a fundamental shift in the field of optimal scaling occurred with the work of Yang et al. (2024). It changed the focus from model parametrizations towards the norm perspective by showing that Maximum Update Parametrization (μP) (Yang et al., 2022) can be derived from a more fundamental principle: enforcing a *spectral condition* on the model weights and their updates during the training. We briefly explain the idea behind each of them below.

μP introduces theoretically grounded scaling rules for hyperparameters as a function of model width in order to ensure "maximal" feature learning in the infinite width limit. This way, the model is guaranteed to learn meaningful features while remaining stable as one scales up its size. As an important by-product, it was found that models with different widths, once parameterized within μP, all share the same optimal hyperparameters (e.g. learning rate) – therefore allowing for what is known as *zero-shot hyperparameter transfer*. This property has been extensively used for the past years to ensure optimal model scaling by tuning hyperparameters for a small (proxy) model, and then effortlessly transferring them to a larger one (OpenAI et al., 2024; Gunter et al., 2024; Dey et al., 2024; Meta AI, 2025; Zuo et al., 2025).

In turn, the spectral condition is formulated as:

Definition 1 (Spectral condition)**.** *Consider applying a gradient update $\Delta \boldsymbol{W}_\ell \in \mathbb{R}^{d^\ell_{\text{out}} \times d^\ell_{\text{in}}}$ to the ℓth weight matrix $\boldsymbol{W}_\ell \in \mathbb{R}^{d^\ell_{\text{out}} \times d^\ell_{\text{in}}}$ for a layer $\ell = 1, \ldots, L$. The spectral norms of these matrices should satisfy*

$$\|\boldsymbol{W}_\ell\|_* = \Theta\left(\sqrt{\frac{d^\ell_{\text{out}}}{d^\ell_{\text{in}}}}\right) \qquad \text{and} \qquad \|\Delta \boldsymbol{W}_\ell\|_* = \Theta\left(\sqrt{\frac{d^\ell_{\text{out}}}{d^\ell_{\text{in}}}}\right), \tag{1}$$

where $\|\boldsymbol{W}\|_*$ is the spectral norm, also equal to the largest singular value of $\boldsymbol{W}$, and $\|\boldsymbol{x}\|_{\text{RMS}} = \|\boldsymbol{x}\|_2 / \sqrt{d}$. $\Theta(1)$ indicates scaling behaviour (in this case, "constant"[3]) w.r.t. infinite width limit

[1] `https://github.com/rakkit/disco`

[2] `https://wandb.ai/sdlaml-llm/norm-transfer/reports/Norm-Transfer--VmlldzoxNDYwNjE2Mw`

[3] Formally, $f(x) = \Theta(g(x))$ indicates that there exist constants $A, B > 0$ so that $A \cdot g(x) \leq f(x) \leq B \cdot g(x)$.

$d \to +\infty$. If conditions in Definition 1 are met, the zero-shot hyperparameter transfer is guaranteed and the model is being trained in the μP regime.

Let us rewrite it in a more "natural" way as:

$$\|\boldsymbol{W}_\ell\|_{\text{RMS}\to\text{RMS}} = \Theta(1) \qquad \text{and} \qquad \|\Delta\boldsymbol{W}_\ell\|_{\text{RMS}\to\text{RMS}} = \Theta(1), \tag{2}$$

where we follow Large et al. (2024) and introduce the core concept of this work:

Definition 2 (Induced operator norm[4])**.** *Given a matrix $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ and two normed vector spaces $(\mathbb{R}^{d_{\text{in}}}, \|\cdot\|_\alpha)$ and $(\mathbb{R}^{d_{\text{out}}}, \|\cdot\|_\beta)$, the "$\alpha$ to β" induced operator norm is given by:*

$$\|\boldsymbol{W}\|_{\alpha\to\beta} = \max_{\boldsymbol{x}\in\mathbb{R}^{d_{\text{in}}}} \frac{\|\boldsymbol{W}\boldsymbol{x}\|_\beta}{\|\boldsymbol{x}\|_\alpha}. \tag{3}$$

The operator norms we are most interested in will be:

$$\|\boldsymbol{W}\|_{1\to\text{RMS}} \coloneqq \max_j \|\text{col}_j(\boldsymbol{W})\|_{\text{RMS}}\,, \tag{4}$$

$$\|\boldsymbol{W}\|_{\text{RMS}\to\text{RMS}} \coloneqq \sqrt{d_{\text{in}}/d_{\text{out}}}\,\|\boldsymbol{W}\|_*\,, \tag{5}$$

$$\|\boldsymbol{W}\|_{\text{RMS}\to\infty} \coloneqq \max_i d_{\text{in}}\,\|\text{row}_i(\boldsymbol{W})\|_{\text{RMS}}\,, \tag{6}$$

where $\text{row}_i(.)$ and $\text{col}_j(.)$ denote the i-th row and j-th column of a matrix. Bernstein & Newhouse (2024a) showed that by deriving *duality maps* for operator norms, one not only enforces the bound on weight updates as per Eq. 2, but moreover performs the *steepest descent* under assigned norms. For a set of norms in Eq. 4–6, the corresponding duality maps for the gradient $\boldsymbol{G}$ with singular value decomposition (SVD) $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ are:

$$\|.\|_{1\to\text{RMS}}: \quad \text{col}_j(\boldsymbol{G}) \mapsto \frac{\text{col}_j(\boldsymbol{G})}{\|\text{col}_j(\boldsymbol{G})\|_{\text{RMS}}} \tag{7}$$

$$\|.\|_{\text{RMS}\to\text{RMS}}: \quad \boldsymbol{G} \mapsto \sqrt{\tfrac{d_{\text{out}}}{d_{\text{in}}}} \times \boldsymbol{U}\boldsymbol{V}^\top \tag{8}$$

$$\|.\|_{\text{RMS}\to\infty}: \quad \text{row}_i(\boldsymbol{G}) \mapsto \frac{1}{d_{\text{in}}} \frac{\text{row}_i(\boldsymbol{G})}{\|\text{row}_i(\boldsymbol{G})\|_{\text{RMS}}} \tag{9}$$

where the $\|.\|_{\text{RMS}\to\infty}$ norm was added by Pethick et al. (2025a). Moreover, they wrapped the norm-based approach outlined above into a Scion optimizer.[5]

Within Scion, one has to assign an operator norm to each layer, e.g. out of those in Eq. 4–6, or additional ones like $\|.\|_{1\to\infty}$. The corresponding duality maps determine how raw gradients should be transformed for those layers before the optimizer updates the weights. To simplify the design, layers are typically grouped as e.g. input, hidden, and output, and norms are assigned to these groups. Importantly, although norms are assigned, model weights are not explicitly transformed; only the raw gradients are, via duality maps.

One prominent example of the norm-based view on model optimization is the Muon optimizer (Jordan et al., 2024), which proved to outperform Adam at scale (Liu et al., 2025) and showed great performance for models up to 1T parameters (Team et al., 2025). Muon can be viewed as a specific instantiation of Scion: it optimizes hidden layers under $\|.\|_{\text{RMS}\to\text{RMS}}$ assumption, and uses Adam for the remaining parameters. However, only in the case with no exponential moving average does Adam coincide with the steepest descent in "max-of-max norm" (Bernstein & Newhouse, 2024b). Since this is uncommon in practice, no "natural" norm applies, making Muon hard to analyze through the norm lens. By contrast, Scion naturally incorporates the norm perspective, updating every layer with an assigned, layer-specific norm.

In practice, using norm-based optimizers as of now looks like a free lunch: they require only one momentum buffer[6] (compared to two for Adam), result in better performance with almost no computational overhead in large-scale distributed scenarios, and by design have zero-shot hyperparameter transfer built in. Moreover, the norm-based approach, while having similar practical benefits as μP in terms of hyperparameter transfer, provides more insights into the dynamics of the model training: optimizer-assigned norms can be used naturally to monitor the training dynamics on a per-layer basis. This observation leads us to discoveries that we describe in Sec. 3.

[4] In the following we will omit "induced operator" for simplicity.

[5] However, we do prefer to look at Scion as a framework or a set of principles rather than as an optimizer.

[6] Or even none, see `ScionLight` (Pethick et al., 2025a).

2.2 TRAINING SETUP

In all experiments, we use the Llama 3 architecture (Grattafiori et al., 2024) and `torchtitan` training framework (Liang et al., 2025). Most of the experiments are performed on the small-scale proxy model with a total size of 69M trainable parameters. For additional ablations in Sec. 3.2, we scale up the model up to $\times 12$ in width (to 1.3B parameters) and up to $\times 32$ in depth (to 168M). Notably, we employ a `norm-everywhere` approach, inspired by the concept of well-normedness in Large et al. (2024) and the recent line of work (Loshchilov et al., 2025; Kim et al., 2025). Effectively, we ensure that the input $\boldsymbol{x}$ to every `Linear` layer is normalized to $||\boldsymbol{x}||_{\text{RMS}} = 1$ by a preceding `RMSNorm` layer without learnable parameters. More details on model configurations are provided in Appendix A.2 and Appendix A.3.

As optimizer, we use Scion without weight decay (i.e. its unconstrained version) (Pethick et al., 2025a) without momentum and with the norm assumptions $\|.\|_{1\to\text{RMS}} \Rightarrow \|.\|_{\text{RMS}\to\text{RMS}} \Rightarrow \|.\|_{\text{RMS}\to\infty}$ for input $\Rightarrow$ hidden $\Rightarrow$ output layers. Furthermore, we developed its distributed version, which natively integrates into `torchtitan`, supports FSDP/DDP/TP/EP/CP/PP strategies, and greatly speeds up the training at scale compared to the standard implementation. We make it openly available and provide more details in Appendix A.5.

For pretraining, we use a high-quality partition of the Nemotron-CC dataset (Su et al., 2025), Llama 3 tokenizer (Grattafiori et al., 2024) with a vocabulary size of $128{,}256$ (after padding) and a context window of 4096. All the models are pretrained with the causal language modelling task. Unless stated otherwise, a constant learning rate schedule without warmup and without decay is used. This allows us, for a given set of hyperparameters, to perform a single long run and evaluate progressively larger dataset sizes, rather than conducting several runs for each dataset individually, thereby substantially reducing computational costs (Hu et al., 2024; Hägele et al., 2024).

2.3 OPTIMAL NORM MEASUREMENT

Our initial intuition was that for a given model and data scale, there is always some optimal norm value, corresponding to some optimal hyperparameter choice. To establish this, we focus on the output layer with the Scion-assigned $\|.\|_{\text{RMS}\to\infty}$ norm (hereafter referred to as *output norm*) as being the most natural layer to study[7]. The choice of $\|.\|_{\text{RMS}\to\infty}$ norm is motivated by Bernstein & Newhouse (2024a) as mapping from a "natural" continuous RMS norm semantics for hidden model representations onto a discrete vocabulary, although we also ablate this in Appendix A.12.2. Since by default we disable momentum and any regularization, we are only left with learning rate (η) and batch size (B) as hyperparameters to tune for optimality.

To extract the optimal hyperparameter configuration and the corresponding optimal norm, we run an (η, B) grid search for a given model and a given pretraining dataset size (hereafter referred to as horizon D, measured in tokens), and evaluate model performance with training loss (cross-entropy of the next token prediction). First, we examine how the optimal norm changes as the horizon increases by tracking it over the course of a long training run. Then, we fix the horizon and scale up the model in width and depth, repeating the same optimal norm measurement.

Practically, for every batch size we are interested in "profiling" across learning rates (see Appendix A.2 for details on the grid and random seed variations), i.e. picking the best one and the corresponding optimal norm. However, an empirically lowest-loss point on the η grid turned out to be a statistically noisy estimate; therefore, for each batch size, we perform a fit to the distribution of training loss vs. output norm across learning rates. Finally, we extract the optimal norm value from the fitted curve and the corresponding optimal learning rate from the nearest data point to the fitted optimum. We provide more details on the fitting procedure in Appendix A.4.

[7]The output layer is invariant to both width and depth scaling, is the most sensitive to learning rate tuning (Sec. 3.4), and represents the model as a single entity, e.g. if viewed as a linear classifier on learned hidden representations. Interestingly, we also found that it is the only layer where out of the top 10 singular values, the largest one dominates the others by an order of magnitude, hinting towards a distinct spectrum structure.

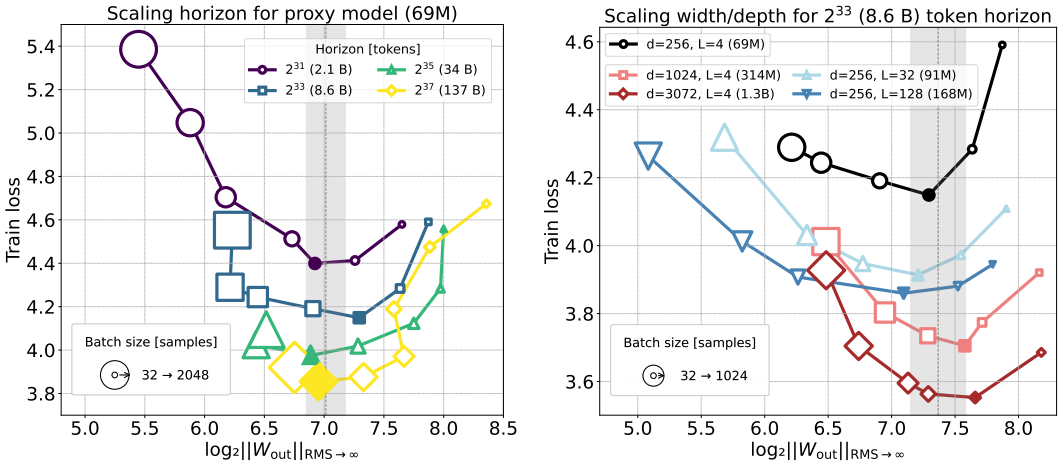
3 RESULTS

3.1 OUTPUT NORM DYNAMICS

First, we describe how the output layer norm evolves depending on the hyperparameter settings. From learning rate scans, we observe that indeed there is an optimal norm value for a given batch size and horizon (Fig. 4). Furthermore, assuming a fixed number of gradient steps/tokens, learning rate is positively correlated with the output norm: the higher the learning rate, the higher the norm (Fig. 4 and Fig. 5). Since we use an unconstrained version of Scion, the norms generally grow with the number of gradient steps (Pethick et al., 2025a). However, norm values can also be explicitly constrained with weight decay (see Appendix A.10) or with various spectral clipping techniques (Newhouse et al., 2025). Interestingly, the norm growth is not linear in log-log scale but *piecewise linear*: we observe *phase transitions* with the slope changing for all batch sizes at the norm value of $2^6 - 2^7$ and then at $2^9 - 2^{10}$, where for the latter the dynamics enters the turbulence region. This may be connected to a recent phenomenon observed in the loss curve dynamics (Mircea et al., 2025).

3.2 OPTIMAL NORM TRANSFER

After analysing (η, B) grid scans across horizons and models of varying width/depth, we visualise results in Fig. 1, with an extended set of plots in Appendix A.8 and Appendix A.14. Each data point corresponds to optimally tuned learning rate η^* for a given batch size, minimising training loss for that horizon and model. We report our observations below, separately for each direction of scaling.



(a) (b)

Figure 1: **Training loss against output layer norm across batch sizes. (a)** Fixed proxy model (69M params) while increasing token horizon from 2^{31} to 2^{37}. **(b)** Fixed token horizon 2^{33} while scaling width/depth w.r.t. the proxy model as indicated in the legend. Each batch size point (increasing from 32 in $\times 2$ steps, as reflected by marker size) has its learning rate optimally tuned, with the optimal batch size per horizon/model configuration indicated by a filled marker. All curves share optimal norm at 7.0 ± 0.2 across horizons and 7.4 ± 0.2 across models (grey band).

Data scaling: After profiling across learning rates and plotting optimal norm against batch size, we observe that for a given horizon there is *a single optimal batch size* with the corresponding optimal output norm $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\infty} = 2^{7.0\pm0.2}$. Intriguingly, this value transfers across horizons. We refer to this phenomenon as *norm transfer*: the optimal (η, B) configuration for a given horizon must result in the optimal norm of $\approx 2^7$. Also note that the optimal batch size grows with horizon scaling, which we discuss in Sec. 3.3.

Model width scaling: It is expected to preserve the optimal norm by the design of our optimizer via the spectral condition (Eq. 1). Indeed, in Fig. 1b we observe that scaling up in width by a factor of $\times 12$ while keeping the horizon fixed results in the nested "μP-style" curves, sharing the same optimal norm while resulting in lower loss as we scale up.

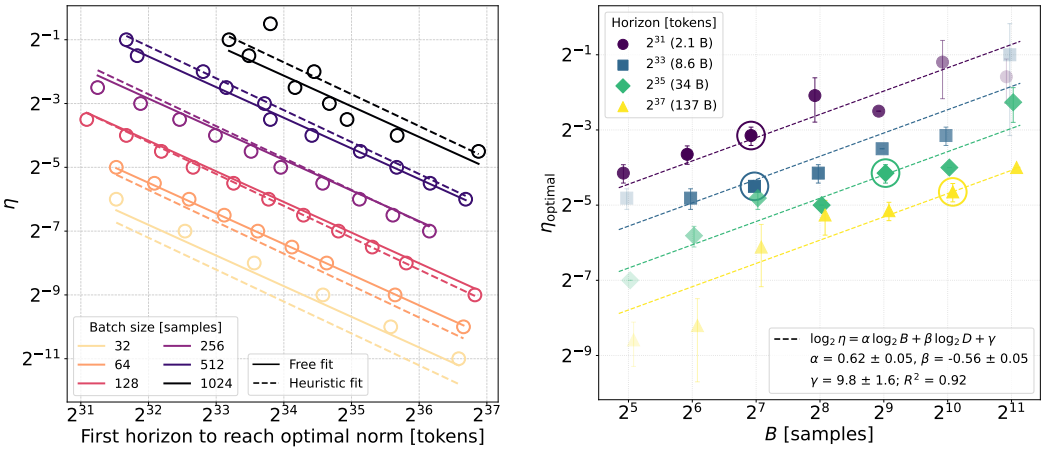
Model depth scaling: Although not obvious *a priori*, we observe experimentally that scaling up in the number of layers by a factor of $\times 32$ results in norm transfer. This is quite surprising, since we do not employ any of the established depth-transfer techniques (Bordelon et al., 2023; Yang et al., 2023; Dey et al., 2025). We ablate them in Appendix A.11 and find that in our setup they all induce learning rate transfer, but our strategy (no residual scaling factors, initialization rescaling of layers prior to residuals by $1/\sqrt{2N_{\text{layers}}}$) results in the lowest loss. We speculate that this may be related to our `norm-everywhere` approach (Sec. 2.2) and uniformity in norm treatment by the optimizer and weight initialization.

Additional ablations: In practice, one is interested in running Scion with non-zero momentum and with a decaying learning rate schedule. We study the impact of these two options in Appendix A.12 and observe that they both show norm transfer. Notably, the addition of momentum largely reduces sensitivity to batch size choice with multiple values resulting in the same optimal norm and loss (Fig. 9). Impact of learning rate decay is also important, as we find it greatly flattens the norm optimum and thus reduces sensitivity to learning rate choice (Fig. 13b). Last but not least, in Appendix A.12.2 we find that norm transfer phenomenon occurs not only in $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\infty}$ norm, but also in $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\text{RMS}}$ and $\|\boldsymbol{W}_{\text{in}}\|_{1\to\text{RMS}}$ for the input embedding.

↬ **Summary I:** Within the Scion framework, optimal norm transfers in both model (via width and depth) and data scaling directions: it is *necessary* to choose the hyperparameter configuration so that the model output norm $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\infty}$ falls into the optimal region. Tracking alternative norms ($\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\text{RMS}}$ or $\|\boldsymbol{W}_{\text{in}}\|_{1\to\text{RMS}}$) maintains the consistency of the transfer. The same behaviour holds with non-zero momentum and learning rate decay.

3.3 Optimal (η, B) scaling rule

Despite the discovered norm guidance, it is still not obvious how to select the corresponding optimal hyperparameters. Or more generally, what is the *sufficient* condition for optimality? In this Section, we explore this question.



(a) (b)

Figure 2: **(a) Multiple (η, B) combination can reach the optimal norm $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\infty} = 2^{7.0\pm0.2}$** for a given token horizon. Colours denote batch size; the y-axis is learning rate. Solid and dashed lines denote free and heuristic fits. **(b) Optimal learning rate per batch size across horizons.** Circled markers indicate optimal (η^*, B^*) with the lowest loss; opacity is linearly interpolated between the lowest- and highest-loss runs. Error bars show systematic variation from the fitting method (Appendix A.4). Dashed lines are a joint linear regression with $\log_2 \eta^* \sim \log_2 B + \log_2 D$.

Fig. 2a illustrates that the optimal norm condition observed in Fig. 1 is necessary but not sufficient. For each token horizon (x-axis), we plot the learning rates (y-axis) and batch sizes (colour) that

reach[8] the optimal-norm region $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\infty} \in [2^{6.8}, 2^{7.2}]$ (x-axis), separately for each batch size (colour). One can observe that for a given horizon, every batch size will reach optimal norm with a sufficiently high learning rate. We further fit the data with linear models $\log_2 \eta = \alpha_{\text{first}} \log_2 B + \beta_{\text{first}} \log_2 D_{\text{first}} + \gamma_{\text{first}}$ (free fit) and $\log_2 \eta = 1.5 \log_2 B - \log_2 D_{\text{first}} + \gamma_{\text{first}}$ (heuristic fit). For the free fit, we find the exponents $\alpha_{\text{first}} = 1.32 \pm 0.03$ and $\beta_{\text{first}} = 0.96 \pm 0.03$, which are close to the values from the heuristic fit.

Hence, we cannot fully rely on the output norm as a guide to selecting optimal hyperparameters. Let us now explicitly derive them by first unfolding Fig. 1a and including optimal learning rate information that was profiled away. Specifically, we are interested in how the optimal learning rate η^* changes within a fixed horizon D with the batch size B change, and then with horizons D scaled up. Fig. 2b shows the corresponding data points along with a linear regression fit $\log_2 \eta^*(B, D) = \alpha \log_2 B + \beta \log_2 D + \gamma$. Note that only circled markers are per-horizon optima with the lowest loss.

- The coefficients of the fit $\alpha = 0.62 \pm 0.05$, $\beta = -0.56 \pm 0.05$ are consistent with a well-established square-root scaling with batch size (Malladi et al., 2024) and data horizon (Bjorck et al., 2025) for Adam, respectively. Similar to AI et al. (2025); Sato et al. (2025) we observe no surge phenomenon (Li et al., 2024), i.e. transition for a fixed D from $\eta^* \propto \sqrt{B}$ to $\eta^* \propto 1/\sqrt{B}$ scaling rules for batch sizes higher than the critical one (Zhang et al., 2025). Theoretically, Jianlin (2025) explains this from the mean field theory perspective.
- Different batch sizes B result in different losses, and for each horizon D there is an optimal one $B^*(D)$, as emphasized in Fig. 2b with circled markers and marker opacity for relative loss difference. The optimal batch size increases with horizon scaling: in Appendix A.9 we measure with extended set of horizons $B^*(D) \propto D^{0.45\pm0.07}$, which is consistent with Adam (Li et al., 2025; Bergsma et al., 2025) and intriguingly with $B^* \propto \sqrt{D}$.
- Using $B^*(D) \propto D^{0.45}$ and $\log_2 \eta^*(B, D) \propto 0.62 \log_2 B - 0.56 \log_2 D$ with the corresponding uncertainties, we obtain for the optimal learning rate scaling $\eta^*(D) \propto D^{-0.28\pm0.07}$. This observation is consistent with Li et al. (2025) but appears to be in tension with Shen et al. (2024); Bergsma et al. (2025), albeit our methodologies are not fully comparable.[9] Again, this is interestingly close to $\eta^*(D) \propto D^{-1/4}$.
- Since there exists a single optimal batch size for each data horizon, the number of devices usable for training is fundamentally capped: beyond a point, increasing the number of devices either hurts throughput (small per-device microbatch size to keep the optimal global batch size) or degrades loss (leaving the optimal batch size region to keep throughput). This hints towards an interesting research direction: if this limit can be bypassed.
- In fact, for a fixed horizon, it is not a single optimal (η^*, B^*) but an *optimal region* $(\eta^* \pm \Delta\eta, B^* \pm \Delta B)$ that results in optimality (see opacity in Fig. 2b). We relate this to the notion of learning rate sensitivity (Wortsman et al., 2023) that we rephrase as *norm sensitivity*. We think this region is defined by the "flatness" of the horizon curve (Fig. 1a) around the optimal norm value. Within this region, one can "exchange" learning rate for batch size via the $\eta \propto \sqrt{B}$ rule, thus allowing for some flexibility in optimal hyperparameter choice.

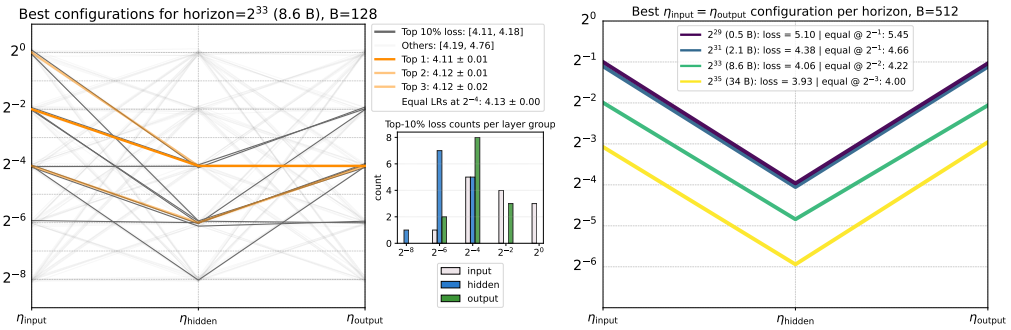
↬ **Summary II:** For Scion, we measure the following hyperparameter scaling rules inducing *sufficient* optimal scaling condition:

$$\eta^*(D) \propto D^{-0.28\pm0.07} \qquad \text{and} \qquad B^*(D) \propto D^{0.45\pm0.07}, \tag{10}$$

consistent with the Adam's scaling exponents. For a fixed horizon D, one can trade off $\eta^* \leftrightarrow B^*$ via the $\eta \propto \sqrt{B}$ rule within the region of low norm sensitivity, without loss in performance. By Scion's design, these observations hold true with model width scaling.

[8] Optimal norm will most likely be reached at some point (provided learning rate sweep resolution in Fig. 2a is too small), since in unconstrained Scion the weight norms are growing in time (see Sec. 3.1 and Fig. 5).

[9] For example, because of weight decay usage in Bergsma et al. (2025), which significantly affects norm dynamics by constraining it, as we discuss in Sec. 5.



(a) (b)

Figure 3: **(a) Parallel-coordinates view of per-layer-group learning rates** $(\eta_{\text{input}}, \eta_{\text{hidden}}, \eta_{\text{output}})$**.** Results are for the proxy model (69M parameters) and batch size $B = 128$ samples, averaged across random seeds as described in Appendix A.2. Dark gray lines are the top 10% runs (loss 4.11–4.18); light gray lines are the remainder (loss 4.19–4.76). Orange traces highlight the three best settings. The inset histogram shows the distribution of top 10% counts for each layer group. **(b) Best learning rate layouts per training horizon under the constraint** $\eta_{\text{input}} = \eta_{\text{output}}$**.** Results are for the proxy model (69M parameters) and batch size $B = 512$ samples. All horizons favor a V-shaped layout with η_{hidden} smaller than the input/output learning rates by the same $\times 1/8$ factor. In the legend we also report loss for the optimal $\eta_{\text{input}} = \eta_{\text{hidden}} = \eta_{\text{output}} \equiv \eta$ layout ("equal @η").

3.4 Optimal per-layer-group learning rate

So far, we approached scaling from a "global" learning rate point of view. However, this may not be the case, and intricate dynamics can emerge where various layers require different learning rates at different scales to be trained optimally, thus questioning our conclusions so far. In this Section, we explore if this is the case.

Fig. 3a presents results for a proxy model (69M parameters), fixed data horizon (8.6B tokens) and fixed batch size ($B = 128$ samples, optimal for this horizon) where we run grid search over learning rate values $\eta \in \{2^{-8}, 2^{-7}, \ldots, 2^{0}\}$ for input (token embedding), output (linear projection onto vocabulary) and hidden (all the others) layers, averaged across random seeds (Appendix A.2). We observe that there is little optimal learning rate imbalance across layer groups, and uniform learning rate assignment results in the same loss as the optimal configurations within uncertainties. Furthermore, from the width of the optimal nodes count histograms per layer groups, we conclude that the output layer is the most sensitive to learning rate mistuning, with the sensitivity progressively decreasing for hidden and then input layers.

From analysing Fig. 3a and additional ones for different batch sizes (Appendix A.13) we found that the configuration $\eta_{\text{input}} : \eta_{\text{output}} : \eta_{\text{hidden}} = 1 : 1/8 : 1$ is always among the top three. This symmetry simplifies the learning scan and notably contradicts the optimal configurations suggested in Pethick et al. (2025a) and Riabinin et al. (2025). To study dynamics with horizon scaling, we perform the learning rate grid scan same as in Fig. 3a but with constraining $\eta_{\text{input}} = \eta_{\text{output}}$ [10], for the proxy model with $B = 512$. Fig. 3b illustrates the results, where we see the optimal hidden ratio $(\eta_{\text{input}}/\eta_{\text{hidden}} = 1/8)$ transfer across horizons, as well as that it brings loss improvement w.r.t. a constant learning rate baseline. Lastly, we note that again, due to the optimizer design, we expect these observations to hold true under model width scaling.

↬ **Summary III:** Uniform learning rate configuration across layers is a strong baseline, which still can be improved with additional hidden layer group tuning: $\eta_{\text{input}} : \eta_{\text{output}} : \eta_{\text{hidden}} = 1 : 1/8 : 1$ yields a relative loss improvement of up to 6% and is transferable across dataset sizes.

[10] In terminology of Bernstein & Newhouse (2024a) this corresponds to *mass* tuning.

4 Related Work

Hyperparameters with model scaling Yang et al. (2022) showed how to transfer optimal hyperparameters from a small to a large model in a principled way via Maximal Update Parametrization (μP). Everett et al. (2024) later showed that such transfer is also possible in other parametrizations. Yang et al. (2023); Dey et al. (2025) extended the method towards model scaling in depth. Empirically, scaling laws on how to set optimal hyperparameters as a function of compute (DeepSeek-AI et al., 2024), loss (Hu et al., 2024) or model size (Porian et al., 2025) were measured.

Hyperparameters with data scaling Remains poorly understood theoretically: Smith & Le (2018) showed for SGD how to adjust learning rate and batch size by modelling optimization trajectory as a stochastic differential equation (SDE). Largely, the problem has been approached by measuring hyperparameter scaling rules as a function of the dataset size (Shen et al., 2024; Hu et al., 2024; Filatov et al., 2025; Bergsma et al., 2025; Li et al., 2025).

(η, B) **scaling rules** Historically, studies of interaction between learning rate and batch size emerged as an experimental effort to scale batch size without losing performance (Keskar et al., 2017; Goyal et al., 2018; Hilton et al., 2022). Later, a deeper understand has emerged from various theoretical angles: SDE (Malladi et al., 2024), loss curvature (McCandlish et al., 2018), random matrix theory (Granziol et al., 2021).

Norm-based optimization Starting from the spectral condition (Yang et al., 2024), the approach of transforming gradient updates based on norm assumptions was fully established in Large et al. (2024); Bernstein & Newhouse (2024a), and recently explored in constraining weights themselves (Newhouse et al., 2025). The steepest descent view allowed for connections with manifold learning (Cesista, 2025) and optimizer design (Riabinin et al., 2025). This line of work has led to Muon (Jordan et al., 2024) and Scion (Pethick et al., 2025a;b), along with improvements (Ahn et al., 2025; Amsel et al., 2025), and benchmarks (Wen et al., 2025; Semenov et al., 2025) thereof.

5 Conclusion and Discussion

In this work, we demonstrate that the operator norm of the output layer is a powerful measure that guides joint optimal scaling across both model and dataset dimensions. Informally, one may view our results as:

1. $(\eta,\ B,\ D)$ choice $\xRightarrow{\text{affects}}$ layer operator norm (Sec. 3.1)
2. optimal loss $\xRightarrow{\text{necessitates}}$ constant optimal norm (Sec. 3.2)
3. optimal $\eta^*(D), B^*(D)$ scaling rules $\xRightarrow{\text{result in}}$ optimal loss (Sec. 3.3)

In words, we show (1) how norms evolve with hyperparameter change and how to tune it to desired values; (2) the optimal loss configuration must have a predefined output layer norm, transferable across data and model scales; (3) optimal hyperparameter scaling rules result in optimal loss.

While the latter statement can sound trivial, we do not regard it that way. We still don't know why the rules in Summary II, resembling square-root and 1/4-power laws, are induced in this form. Moreover, how do these rules connect with our main finding, a necessary condition of scaling trajectory in (data, model) axes to have the same constant value – or one might say, to remain on a *manifold* (Bernstein, 2025). At this point more new questions arise:

- Why does optimal norm transfer? It is puzzling what makes the optimal scaling trajectory remain on the constant norm manifold, as well as what defines its structure.
- What is the reason behind sufficient optimal scaling? While we show how to set hyperparameters optimally, there is something missing in the norm perspective to explain this.
- How can the constant norm condition be leveraged? It looks like a naturally emerging inductive bias that one can take advantage of to optimize the training process.

We see that our study merely scratches the surface and yet opens up new exciting directions for future research.

ACKNOWLEDGMENTS

We thank Kaiyue Wen and Ismail Khalfaoui Hassani for helpful discussions and feedback on the manuscript. This research was supported by TrustLLM funded by Horizon Europe GA 101135671, by the Helmholtz Foundation Model Initiative as a part of the Synergy Unit. The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC). Parts of computational resources were provided by the German AI service center WestAI.

REFERENCES

Kwangjun Ahn, Byron Xu, Natalie Abreu, Ying Fan, Gagik Magakyan, Pratyusha Sharma, Zheng Zhan, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025.

Essential AI, :, Ishaan Shah, Anthony M. Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J Shah, Khoi Nguyen, Kurt Smith, Michael Callahan, Michael Pust, Mohit Parmar, Peter Rushton, Platon Mazarakis, Ritvik Kapila, Saurabh Srivastava, Somanshu Singla, Tim Romanski, Yash Vanjani, and Ashish Vaswani. Practical efficiency of muon for pretraining. *arXiv preprint arXiv:2505.02222*, 2025.

Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm, 2025. URL `https://arxiv.org/abs/2505.16932`.

Shane Bergsma, Nolan Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Power lines: Scaling laws for weight decay and batch size in llm pre-training. *arXiv preprint arXiv:2505.13738*, 2025.

Jeremy Bernstein. Modular manifolds. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250926. https://thinkingmachines.ai/blog/modular-manifolds/.

Jeremy Bernstein and Laker Newhouse. Modular duality in deep learning. *arXiv preprint arXiv:2410.21265*, 2024a.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024b.

Johan Bjorck, Alon Benhaim, Vishrav Chaudhary, Furu Wei, and Xia Song. Scaling optimal lr across token horizons. *arXiv preprint arXiv:2409.19913*, 2025.

Blake Bordelon, Lorenzo Noci, Mufan Bill Li, Boris Hanin, and Cengiz Pehlevan. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. *arXiv preprint arXiv:2309.16620*, 2023.

Franz Louis Cesista. Muon and a selective survey on Steepest Descent in Riemannian and non-Riemannian Manifolds, April 2025. URL `http://leloykun.github.io/ponder/steepest-descent-non-riemannian/`.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

DeepSeek-AI, :, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu,

Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Nolan Dey, Quentin Anthony, and Joel Hestness. The practitioner's guide to the maximal update parameterization, Sep 2024. URL https://www.cerebras.ai/blog/the-practitioners-guide-to-the-maximal-update-parameterization.

Nolan Dey, Bin Claire 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. *arXiv preprint arXiv:2505.01618*, 2025.

Katie Everett, Lechao Xiao, Mitchell Wortsman, Alexander A. Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers. *arXiv preprint arXiv:2407.05872*, 2024.

Wei Feng, Will Constable, and Yifan Mao. Getting started with fully sharded data parallel (fsdp2), 2025. URL https://docs.pytorch.org/tutorials/intermediate/FSDP_tutorial.html.

Oleg Filatov, Jan Ebert, Jiangtao Wang, and Stefan Kesselheim. Time transfer: On optimal learning rate and batch size in the infinite data limit. *arXiv preprint arXiv:2410.05838*, 2025.

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*, 2018.

Diego Granziol, Stefan Zohren, and Stephen Roberts. Learning rates as a function of batch size: A random matrix theory approach to neural network training. *arXiv preprint arXiv:2006.09092*, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan

Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov,

Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, Deepak Gopinath, Dian Ang Yap, Dong Yin, Feng Nan, Floris Weers, Guoli Yin, Haoshuo Huang, Jianyu Wang, Jiarui Lu, John Peebles, Ke Ye, Mark Lee, Nan Du, Qibin Chen, Quentin Keunebroek, Sam Wiseman, Syd Evans, Tao Lei, Vivek Rathod, Xiang Kong, Xianzhi Du, Yanghao Li, Yongqiang Wang, Yuan Gao, Zaid Ahmed, Zhaoyang Xu, Zhiyun Lu, Al Rashid, Albin Madappally Jose, Alec Doane, Alfredo Bencomo, Allison Vanderby, Andrew Hansen, Ankur Jain, Anupama Mann Anupama, Areeba Kamal, Bugu Wu, Carolina Brum, Charlie Maalouf, Chinguun Erdenebileg, Chris Dulhanty, Dominik Moritz, Doug Kang, Eduardo Jimenez, Evan Ladd, Fangping Shi, Felix Bai, Frank Chu, Fred Hohman, Hadas Kotek, Hannah Gillis Coleman, Jane Li, Jeffrey Bigham, Jeffery Cao, Jeff Lai, Jessica Cheung, Jiulong Shan, Joe Zhou, John Li, Jun Qin, Karanjeet Singh, Karla Vega, Kelvin Zou, Laura Heckman, Lauren Gardiner, Margit Bowler, Maria Cordell, Meng Cao, Nicole Hay, Nilesh Shahdadpuri, Otto Godwin, Pranay Dighe, Pushyami Rachapudi, Ramsey Tantawi, Roman Frigg, Sam Davarnia, Sanskruti Shah, Saptarshi Guha, Sasha Sirovica, Shen Ma, Shuang Ma, Simon Wang, Sulgi Kim, Suma Jayaram, Vaishaal Shankar, Varsha Paidi, Vivek Kumar, Xin Wang, Xin Zheng, Walker Cheng, Yael Shrager, Yang Ye, Yasu Tanaka, Yihao Guo, Yunsong Meng, Zhao Tang Luo, Zhi Ouyang, Alp Aygar, Alvin Wan, Andrew Walkingshaw, Andy Narayanan, Antonie Lin, Arsalan Farooq, Brent Ramerth, Colorado Reed, Chris Bartels, Chris Chaney, David Riazati, Eric Liang Yang, Erin Feldman, Gabriel Hochstrasser, Guillaume Seguin, Irina Belousova, Joris Pelemans, Karen Yang, Keivan Alizadeh Vahid, Liangliang Cao, Mahyar Najibi, Marco Zuliani, Max Horton, Minsik Cho, Nikhil Bhendawade, Patrick Dong, Piotr Maj, Pulkit Agrawal, Qi Shan, Qichen Fu, Regan Poston, Sam Xu, Shuangning Liu, Sushma Rao, Tashweena Heeramun, Thomas Merth, Uday Rayala, Victor Cui, Vivek Rangarajan Sridhar, Wencong Zhang, Wenqi Zhang, Wentao Wu, Xingyu Zhou, Xinwen Liu, Yang Zhao, Yin Xia, Zhile Ren, and Zhongzheng Ren. Apple intelligence foundation language models. *arXiv preprint arXiv:2407.21075*, 2024.

Jacob Hilton, Karl Cobbe, and John Schulman. Batch size-invariance for policy optimization. *arXiv preprint arXiv:2110.00641*, 2022.

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. *arXiv preprint arXiv:2203.15556*, 2022.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. *arXiv preprint arXiv:2405.18392*, 2024.

Su Jianlin. Rethinking learning rate and batch size (part 3): Muon, Sep 2025. URL `https://kexue.fm/archives/11285`.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

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.

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. *arXiv preprint arXiv:1609.04836*, 2017.

Jeonghoon Kim, Byeongchan Lee, Cheonbok Park, Yeontaek Oh, Beomjun Kim, Taehwan Yoo, Seongjin Shin, Dongyoon Han, Jinwoo Shin, and Kang Min Yoo. Peri-ln: Revisiting normalization layer in the transformer architecture. *arXiv preprint arXiv:2502.02732*, 2025.

Tim Large, Yang Liu, Minyoung Huh, Hyojin Bahng, Phillip Isola, and Jeremy Bernstein. Scalable optimization in the modular norm. *arXiv preprint arXiv:2405.14813*, 2024.

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Zhenyu Ding, Haoying Wang, Ning Ding, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable scale: Part i – optimal hyperparameter scaling law in large language model pretraining. *arXiv preprint arXiv:2503.04715*, 2025.

Shuaipeng Li, Penghao Zhao, Hailin Zhang, Xingwu Sun, Hao Wu, Dian Jiao, Weiyan Wang, Chengjun Liu, Zheng Fang, Jinbao Xue, Yangyu Tao, Bin Cui, and Di Wang. Surge phenomenon in optimal learning rate and batch size scaling. *arXiv preprint arXiv:2405.14578*, 2024.

Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, et al. Torchtitan: One-stop pytorch native solution for production ready llm pretraining. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. ngpt: Normalized transformer with representation learning on the hypersphere. *arXiv preprint arXiv:2410.01131*, 2025.

Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the sdes and scaling rules for adaptive gradient algorithms. *arXiv preprint arXiv:2205.10287*, 2024.

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

Meta AI. Introducing llama 4: Advancing multimodal intelligence, 2025. URL `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`.

Andrei Mircea, Supriyo Chakraborty, Nima Chitsazan, Milind Naphade, Sambit Sahu, Irina Rish, and Ekaterina Lobacheva. Training dynamics underlying language model scaling laws: Loss deceleration and zero-sum learning. *arXiv preprint arXiv:2506.05447*, 2025.

Laker Newhouse, R. Preston Hess, Franz Cesista, Andrii Zahorodnii, Jeremy Bernstein, and Phillip Isola. Training transformers with enforced lipschitz constants. *arXiv preprint arXiv:2507.13338*, 2025.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey

Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2024.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. *arXiv preprint arXiv:2502.07529*, 2025a.

Thomas Pethick, Wanyun Xie, Mete Erdogan, Kimon Antonakopoulos, Tony Silveti-Falls, and Volkan Cevher. Generalized gradient norm clipping & non-euclidean (l_0, l_1)-smoothness. *arXiv preprint arXiv:2506.01913*, 2025b.

Tomer Porian, Mitchell Wortsman, Jenia Jitsev, Ludwig Schmidt, and Yair Carmon. Resolving discrepancies in compute-optimal scaling of language models. *arXiv preprint arXiv:2406.19146*, 2025.

Artem Riabinin, Egor Shulgin, Kaja Gruntkowska, and Peter Richtárik. Gluon: Making muon & scion great again! (bridging theory and practice of lmo-based optimizers for llms). *arXiv preprint arXiv:2505.13416*, 2025.

Naoki Sato, Hiroki Naganuma, and Hideaki Iiduka. Convergence bound and critical batch size of muon optimizer. *arXiv preprint arXiv:2507.01598*, 2025.

Andrei Semenov, Matteo Pagliardini, and Martin Jaggi. Benchmarking optimizers for large language model pretraining. *arXiv preprint arXiv:2509.01440*, 2025.

Yikang Shen, Matthew Stallone, Mayank Mishra, Gaoyuan Zhang, Shawn Tan, Aditya Prasad, Adriana Meza Soria, David D. Cox, and Rameswar Panda. Power scheduler: A batch size and token number agnostic learning rate scheduler. *arXiv preprint arXiv:2408.13359*, 2024.

Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2018.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. *arXiv preprint arXiv:2412.02595*, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Hao Hu, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Chao Hong, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Lijun Lu, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Xinjie Sun, Flood Sung, Heyi Tang, Jiawen Tao, Qifeng Teng, Chensi Wang, Dinglu Wang, Feng Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Xiaofei Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yangkun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Huabin Zheng, Shaojie Zheng, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. *arXiv preprint arXiv:2509.02046*, 2025.

Mitchell Wortsman, Peter J. Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D. Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-dickstein, Kelvin Xu, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. Small-scale proxies for large-scale transformer training instabilities. *arXiv preprint arXiv:2309.14322*, 2023.

Greg Yang, Edward J. 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. *arXiv preprint arXiv:2203.03466*, 2022.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks. *arXiv preprint arXiv:2310.02244*, 2023.

Greg Yang, James B. Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2024.

Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean Foster, and Sham Kakade. How does critical batch size scale in pre-training? *arXiv preprint arXiv:2410.21676*, 2025.

Jingwei Zuo, Maksim Velikanov, Ilyas Chahed, Younes Belkada, Dhia Eddine Rhayem, Guillaume Kunsch, Hakim Hacid, Hamza Yous, Brahim Farhat, Ibrahim Khadraoui, Mugariya Farooq, Giulia Campesan, Ruxandra Cojocaru, Yasser Djilali, Shi Hu, Iheb Chaabane, Puneesh Khanna, Mohamed El Amine Seddik, Ngoc Dung Huynh, Phuc Le Khac, Leen AlQadi, Billel Mokeddem, Mohamed Chami, Abdalgader Abubaker, Mikhail Lubinets, Kacper Piskorski, and Slim Frikha. Falcon-h1: A family of hybrid-head language models redefining efficiency and performance. *arXiv preprint arXiv:2507.22448*, 2025.

A APPENDIX

A.1 LLM USAGE

LLMs were used solely to aid in polishing the writing and improving the clarity of exposition. In addition, code-assist tools were occasionally used for minor programming support, such as code completion and syntax suggestions; they were not employed to design algorithms, generate experiments, or implement the proposed methods from scratch.

A.2 MODEL TRAINING CONFIGURATION

- Proxy model, 69M parameters: 4 hidden layers with $d_{\text{model}} = 256$, Multi-Head Attention with $n_{\text{heads}} = 4$ and $n_{\text{kv-heads}} = 4$, SwiGLU activation function with MLP expansion factor $f_{\text{MLP}} = 2.75$, RoPE with $\theta = 10000$ (Su et al., 2024), Llama 3 tokenizer with vocabulary size of $128\,256$ (after padding) (Grattafiori et al., 2024), input and output embedding layers are not tied.
- $\times 4(12)$ wider model, 314M (1.3B) parameters: same as proxy, except $d_{\text{model}} = 1024\ (3072)$. In width scaling, we keep fixed $d_{\text{head}} = 64$ and scale the number of heads accordingly.
- $\times 8(32)$ deeper model, 91M (168M) parameters: same as proxy, except 32 (128) hidden layers.
- Semi-orthogonal initialization for hidden linear layers and row-wise normalized Gaussian initialization for input/output embedding layers (Pethick et al., 2025a). Initialisation of the last layer of both MLP and attention blocks (those with the output being added with the residual stream) is multiplied by $1/\sqrt{2N_{\text{layers}}}$.
- Dropout disabled, no biases in all `Linear` layers, no weight sharing between input and output embedding layers.
- `norm-everywhere`: normalise input to every `Linear` layer via `RMSNorm` without learnable parameters with $\epsilon = 1e^{-20}$. Effectively, this corresponds to Pre-LN setup with QK-norm plus three additional normalisation layers: V-norm, O-norm (before output projection matrix in Attention block), and MLP-norm (after SwiGLU and before the last MLP layer). Residual connections, including the ones injecting the input embedding layer information, remain intact.
- Random seeds:
  - For all proxy model runs in Sec. 3.2 and Sec. 3.3: `30`
  - For all width/depth-scaled-up model runs: interleaved `30` + `3034` (every 2^2 step is `30`, every other 2^2 step is `3034`)
  - For layout scans in Fig. 3a and Fig. 11: averaging over `30` + `3034` + `303409` for the three "core" learning rate values ($\{2^{-4}, 2^{-6}, 2^{-8}\}$ for $B = 32$, $\{2^{-2}, 2^{-4}, 2^{-6}\}$ for $B = 128$, $\{2^{-1}, 2^{-3}, 2^{-5}\}$ for $B = 512$), `3034` + `303409` for the rest
  - For layout scans in Fig. 3b: `30`
- `torchtitan` codebase, (Liang et al., 2025), FSDP2 (Feng et al., 2025), FlashAttention-2 (Dao, 2023)

A.3 OPTIMIZER CONFIGURATION

Except dedicated ablations, we use the following set of hyperparameters:

- Unconstrained version,
- Learning rate η: grid with $2^{0.5}$ step for the proxy model, and 2^1 step for the width/depth-scaled-up models,
- weight decay $\lambda = 0$,
- momentum $\mu = 0$, without Nesterov momentum,
- no warmup, constant learning rate schedule,

- $\epsilon = 1e^{-20}$ (used in gradient normalisation),
- orthogonalization of gradients for hidden layers ($\|.\|_{\mathrm{RMS}\to\mathrm{RMS}}$ norm assumption) with Newton-Schulz algorithm for $n_{\mathrm{iter}} = 5$ with original Muon coefficients $a, b, c = (3.4445, -4.7750, 2.0315)$ (Jordan et al., 2024).

A.4 Optimal norm fitting & loss smoothing

After naïvely taking the empirical optimum across the learning rate grid (e.g. as the one emphasized with dashed black lines in Fig. 4), we found that the corresponding norm scans, although still indicating norm transfer, are quite noisy (e.g. compare Fig. 1a vs. Fig. 6(a)). From Fig. 4 we noted that data points in loss vs. norm plot resemble parabola if plotted in log-log scale. Furthermore, we know by design that at initialization (step 0) the output norm equals to 1, and train loss equals to 11.765. With this, we chose to perform a constrained fit with a second-order polynomial function in log-log scale $\log(\mathrm{loss}) = a\log(\mathrm{norm})^2 + b\log(\mathrm{norm}) + c$, where the free term c is fixed at precisely the loss value at initialization. We do this using weighted least squares fitting with `np.linalg.lstsq`, where the weighting is done with inverse uncertainties coming from *loss smoothing*, described below. The optimal loss and norm values are then extracted as the parabola optimum coordinates. Optimal learning rate is taken from the data point closest to the fitted optimum. Results of such fits for Fig. 1 can be found in Fig. 12.

Since running several random seeds is computationally intensive, we perform *loss smoothing* to estimate the loss variance and make loss estimates more robust. Essentially, for a given horizon point, instead of taking its loss value, we average it with the previous and next evaluated points (approximately 67M tokens away, or e.g. 128 steps from each other with $B = 128$). Empirically estimated standard deviation is then used in the fits as described above.

In order to get variance estimate in Fig. 2b without running several random seed runs, we vary the fitting procedure outlined above (with/without fitting, with/without loss smoothing, with/without constraint to loss at initialization), thus resulting in 6 total variations. For each of those we track how optimal norm/loss/learning rate changes, and propagate this to plotting and downstream analysis.

A.5 DISTRIBUTED SCION

We implemented a distributed version of Scion/Muon. In this section, we briefly describe the implementation. We assume that the vectorized momentum buffer update is performed before applying the actual weight update.

A.5.1 DDP-DISCO

As a warm-up, we first consider the DDP case (note that a DDP-based version of Muon has already been implemented in `modded-nanogpt`[11]). Our implementation differs slightly from theirs, as we do not explicitly apply communication–computation overlap for DDP.

Algorithm 1: Disco `step_ddp`

Input: Parameters $\{p_i\}_{i=0}^{P-1}$ with $P = |\{p\}|$, world size M, local rank r
bucket_size $\leftarrow M$;
total_buckets $\leftarrow \lceil P/M \rceil$;
global_updates $\leftarrow$ array of length P ;

`/* Step 1: Compute local updates */`
for $i = 0$ **to** $P - 1$ **do**
  if $i \bmod M = r$ **then**
    $g_i \leftarrow$ GETMOMENTUM(p_i) ;
    $u_i \leftarrow \mathrm{LMO}(g_i)$;
    global_updates$[i] \leftarrow u_i$;

`/* Step 2: Communicate updates in buckets */`
for $b = 0$ **to** *total_buckets* -1 **do**
  start_idx $\leftarrow b \cdot M$;
  end_idx $\leftarrow \min(\textit{start\_idx} + M,\ P)$;
  my_idx $\leftarrow$ *start_idx* $+r$;
  if *my_idx* $<$ *end_idx* **then**
    $u_{\text{send}} \leftarrow$ *global_updates*[*my_idx*] ;
  else
    $u_{\text{send}} \leftarrow 0$
  $\{u_j\}_{j=0}^{M-1} \leftarrow$ ALLGATHER(u_{send}) ;
  for $j = 0$ **to** *end_idx* $-$ *start_idx* -1 **do**
    global_updates[*start_idx* $+j$] $\leftarrow u_j$;

`/* Step 3: Apply updates vectorized */`
APPLYUPDATES$(\{p_i\}_{i=0}^{P-1}$, *global_updates*$)$;

Helper functions:

- GETMOMENTUM(p): returns the momentum of p from the momentum buffer.
- LMO(g): runs the LMO based on the chosen norm of p.
- ALLGATHER(u): gathers one tensor u from each rank in the data-parallel group.
- APPLYUPDATES($\{p\}, \{u\}$): applies the global updates $\{u\}$ to the parameters $\{p\}$ in a single vectorized operation.

Notice this version works out-of-the-box for PP+DDP, as we could let each PP(Pipeline Parallelism) stage only manage the parts of the model that the current PP stages needed for forward and backward.

To make it work with TP, one needs to do an extra all-gather in the local update loop.

[11] `https://github.com/KellerJordan/modded-nanogpt`

A.5.2 FSDP-DISCO

Here, "FSDP" refers to a combination of `FSDP2` with arbitrary parallelisms, including Data Parallelism (DP), Context Parallelism (CP), Expert Parallelism (EP), Tensor Parallelism (TP), and Pipeline Parallelism (PP). In this section, we restrict our discussion to FSDP and EP (via DP2EP). In principle, there is no need to treat DP and PP separately: one only needs to all-gather the full gradient before communication in the FSDP case to ensure compatibility with TP.

We assume the design of this work, which applies an $\|.\|_{1\to\mathrm{RMS}}$ norm for the LLM's embedding layer and an $\|.\|_{\mathrm{RMS}\to\infty}$ norm for the output linear layer. (`SignNorm` is also acceptable and remains compatible if one strictly follows Scion's design.)

The `FSDP2` implementation in PyTorch shards weights and gradients along the tensor's first dimension. We discuss Disco under this assumption and further assume that each tensor or matrix corresponds to a single layer. Consequently, fused tensors such as `fused_QKV` in attention layers or `fused_W13` in SwiGLU are not supported.

Under these hypotheses, we can classify parameters into three groups: `embedding`, `experts`, and (pure-)`fsdp`. For updates, no extra communication is required for `embedding` and `experts` parameters, thanks to the `Shard(0)` strategy in `FSDP2`.

Algorithm 2: Disco `step_embedding`

Input: Embedding parameters $\{p_i\}_{i=0}^{P-1}$

`/* Initialise updates storage */`
updates $\leftarrow$ array of length P ;

`/* get momentum and compute LMO update on local shards */`
for $i = 0$ **to** $P-1$ **do**
  $g_i \leftarrow \textsc{GetMomentum}(p_i)$;
  $u_i \leftarrow \mathrm{LMO}(g_i)$;
  updates$[i] \leftarrow u_i$;

`/* Apply updates vectorized */`
$\textsc{ApplyUpdates}(\{p_i\}_{i=0}^{P-1},$ *updates*$)$;

Algorithm 3: Disco `step_experts`

Input: Expert parameters $\{p_i\}_{i=0}^{P-1}$, transpose flag *transpose*

`/* Initialise updates storage */`
updates $\leftarrow$ array of length P ;

`/* get momentum and compute LMO update on local shards */`
for $i = 0$ **to** $P-1$ **do**
  $g_i \leftarrow \textsc{GetMomentum}(p_i)$;
  $u_i \leftarrow \textsc{BatchedLMO}(g_i;$ *transpose_experts* $=$ *transpose*$)$;
  updates$[i] \leftarrow u_i$;

`/* Apply updates vectorized */`
$\textsc{ApplyUpdates}(\{p_i\}_{i=0}^{P-1},$ *updates*$)$;

Noting that MoE expert weights are typically laid out as either $(\text{total\_experts}, d_{\text{out}}, d_{\text{in}})$ or $(\text{total\_experts}, d_{\text{in}}, d_{\text{out}})$, we apply a transpose in the latter case to ensure that the output dimension comes first. In an FSDP + DP2EP setting, each gradient passed to `LMO` is therefore a 3D tensor with layout $(\text{local\_experts}, d_{\text{out}}, d_{\text{in}})$. Accordingly, SVD or Newton-Schulz-based algorithms must correctly handle batched inputs.

And below is the algorithm for purely fsdp-shard parameters.

Algorithm 4: Disco `step_fsdp`

Input: FSDP-sharded parameters $\{p_i\}_{i=0}^{P-1}$, world size M over `fsdp`, local rank r
bucket_size $\leftarrow M$;
total_buckets $\leftarrow \lceil P/M \rceil$;
global_updates $\leftarrow$ array of length P;

for $b = 0$ **to** *total_buckets* -1 **do**
  start $\leftarrow b \cdot M$; *end* $\leftarrow \min(\textit{start} + M,\ P)$;
  my_idx $\leftarrow$ *start* $+ r$;
  for $j = 0$ **to** $M - 1$ **do**
    $i \leftarrow$ *start* $+ j$;
    if $i <$ *end* **then**
      $g_i \leftarrow$ GETMOMENTUM(p_i) `// row-sharded by FSDP;`
      send_list[j] $\leftarrow g_i$;
    else
      send_list[j] $\leftarrow 0$ `// zero padding`
  recv_list $\leftarrow$ ALLTOALL(*send_list*)
  $g^\star \leftarrow$ CONCATROWS(*recv_list*) `// reconstruct full gradient for` $p_{i^\star}$
  $u^\star \leftarrow \mathrm{LMO}(g^\star)$
  updates_send_list $\leftarrow$ SPLITROWS$(u^\star,\ M)$ `// split` $u^\star$ `by rows;`
  updates_recv_list $\leftarrow$ ALLTOALL(*updates_send_list*);
  for $j = 0$ **to** *end* $-$ *start* -1 **do**
    global_updates[*start* + j] $\leftarrow$ *updates_recv_list*[j];

`/* Single vectorized apply */`
APPLYUPDATES($\{p_i\}_{i=0}^{P-1}$, *global_updates*);

Helper functions:

- ALLTOALL(*list*): list-based ALLTOALL over `dp_shard_cp`.
- CONCATROWS(*list*): concatenates row-shards into a full tensor.
- SPLITROWS(u, M): splits u into M contiguous row blocks.

A.6 Output norm variation with learning rate

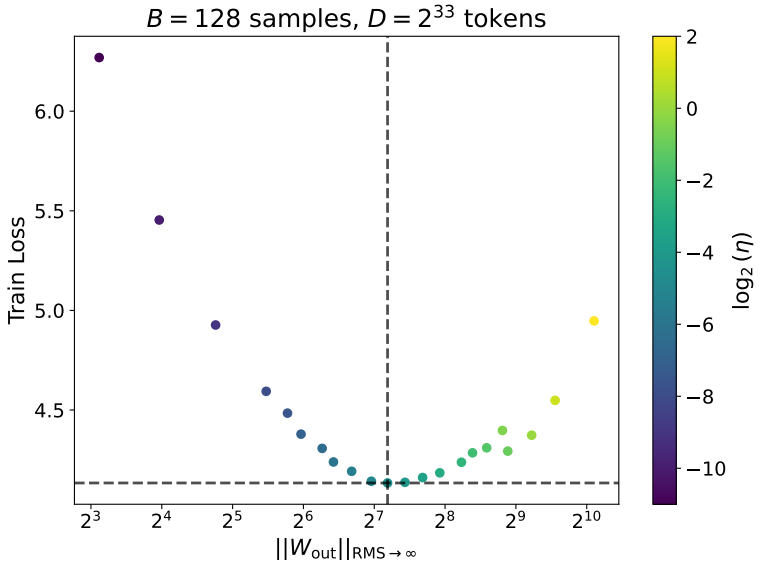
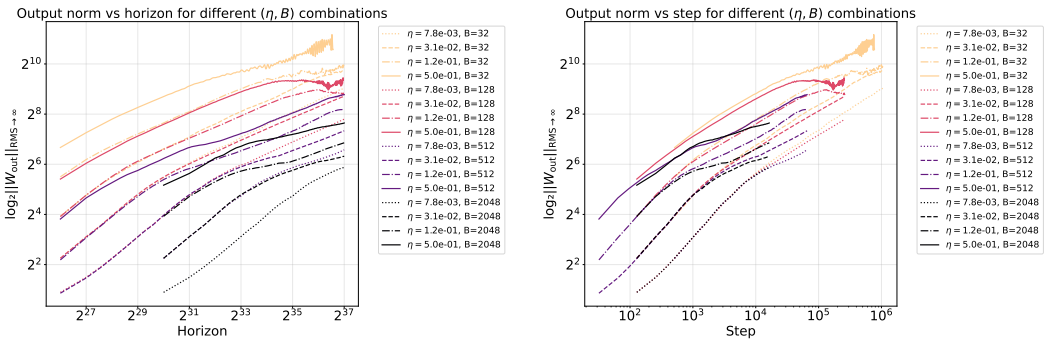


Figure 4: **Learning rate sweep in loss vs. output norm space for a fixed batch size $B = 128$ samples and fixed horizon $D = 2^{33}$ tokens.** Points are colored by $\log_2(\eta)$ where η is the learning rate. Black dashed lines mark the optimal configuration with minimum training loss.

A.7 Output norm evolution with different (η, B)



(a) (b)

Figure 5: **Growth of the output layer norm $\log_2 \|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\infty}$ vs. horizon, in tokens (a) and number of steps (b).** Results are for the proxy model (69M parameters). Each curve is a (learning rate η, batch size B) pair, with B measured in samples: colour encodes batch size and line style encodes learning rate.

A.8 SUPPLEMENTARY PLOTS TO FIG. 1

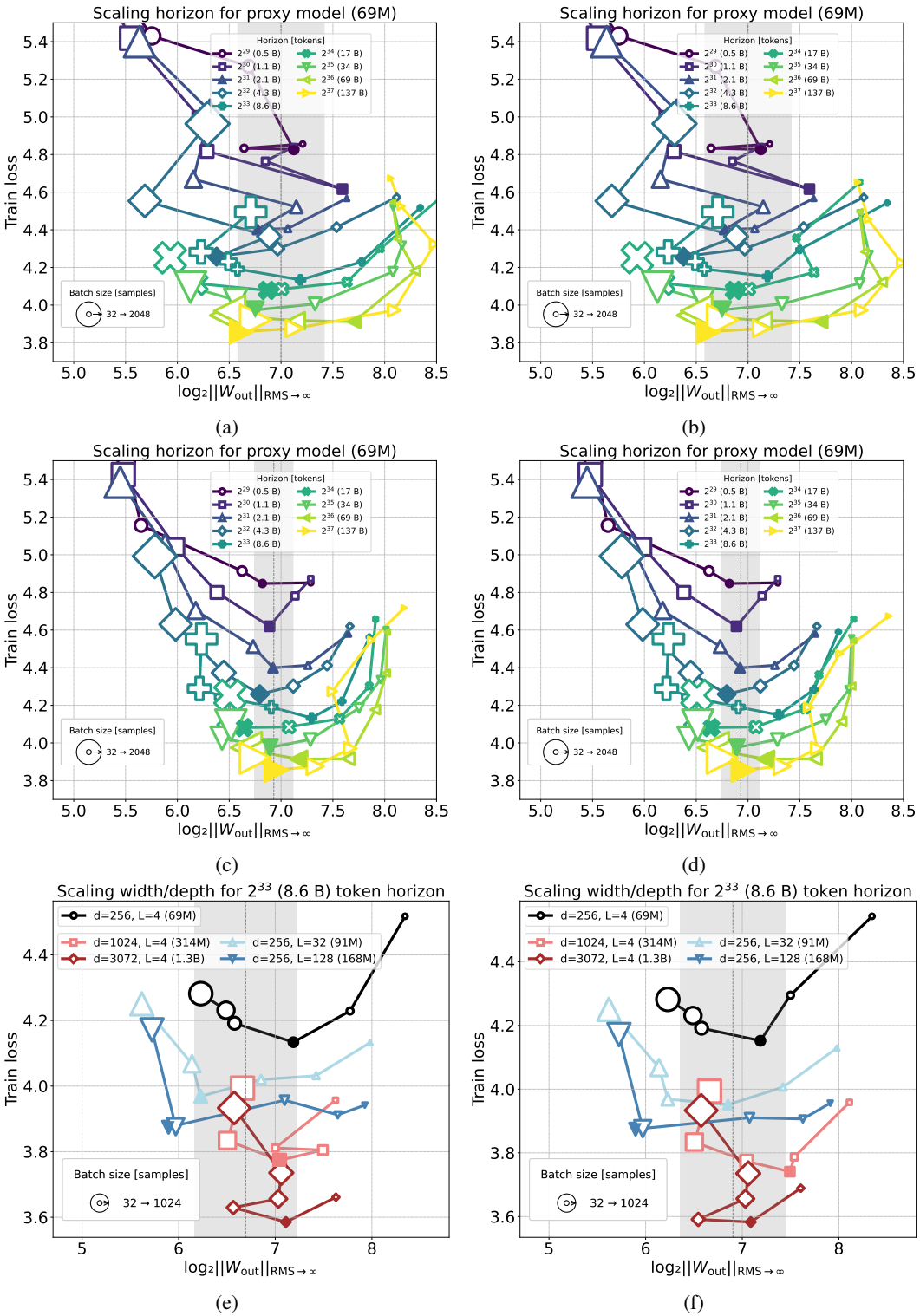
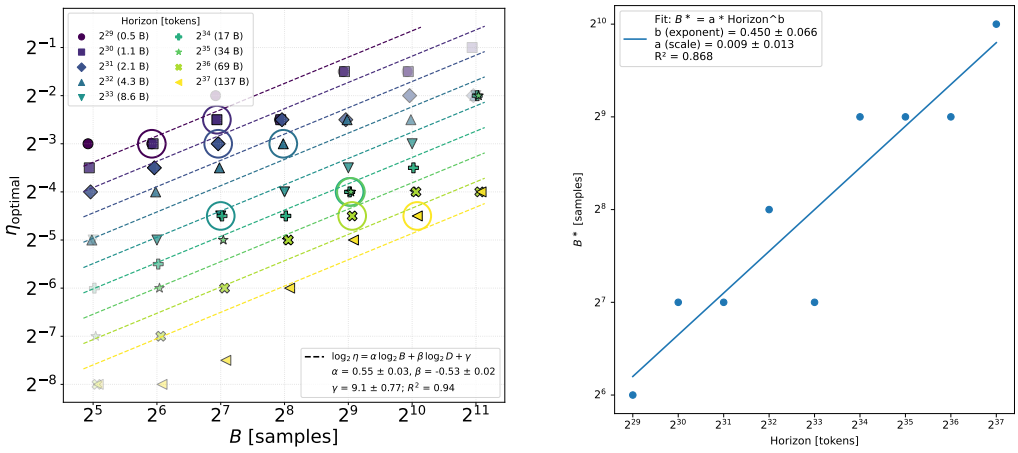


Figure 6: **(a)** Fig. 1a with an extended set of horizons, raw data (i.e. no loss smoothing, no fitting, see Appendix A.4 for details on fitting and loss smoothing). **(b)** Same as (a) + loss smoothing. **(c)** Same as (a) + fitting. **(d)** Same as (a) + fitting + loss smoothing. **(e)** Fig. 1b, raw data (no loss smoothing, no fitting). **(f)** Same as (e) + loss smoothing.

A.9 Optimal $B^*(D)$ measurement

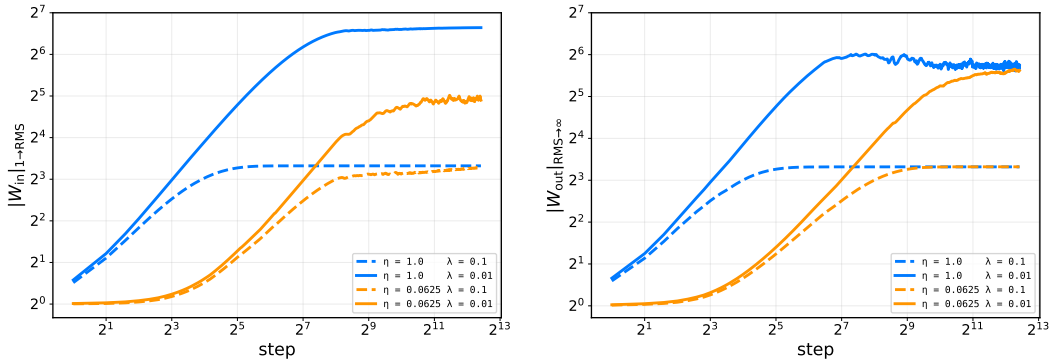


(a) (b)

Figure 7: **(a)** Same as Fig. 2b, but with extended set of horizons. **(b)** Optimal batch size B^* vs. horizon for the optimal (η^*, B^*) configuration at each horizon. The line is a power-law fit (described in legend).

Fig. 2b, for the sake of clarity and simplicity, illustrates only four horizons. This is not really sufficient to extract the scaling of optimal (η^*, B^*) (circled markers) with D, as it would mean fitting of four data points. We therefore perform the ordinary least squares (OLS) fit on the extended set of 9 horizons from Fig. 7a, effectively fitting the circled markers with a line. We model optimal batch size dependency on horizon D as a power law $B^*(D) = aD^b$ and present results on Fig. 7b. We extract $B^* \propto D^{0.45 \pm 0.07}$, consistent with the square-root scaling.

A.10 Norm constraint with weight decay



(a) (b)

Figure 8: **Operator norm against number of gradient update steps.** Fixed batch size $B = 32$, momentum $\mu = 0.1$, two values of learning rate $\eta = \{0.0625, 1.\}$ and two values of weight decay $\lambda = \{0.01, 0.1\}$ (applied as in Pethick et al. (2025a)), for a proxy model (69M parameters). **(a)** $\|\boldsymbol{W}_{\text{in}}\|_{1\to\text{RMS}}$ norm **(b)** $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\infty}$. We see for $\lambda = 0.1$ both norms converging to $1/\lambda$, while for $\lambda = 0.01$ asymptotic values are not conclusive.

A.11 ABLATION OF DEPTH TRANSFER TECHNIQUES

Model: same as our proxy model (Appendix A.2), with the only difference in the head configuration: $n_{\text{query\_heads}} = 2$, $n_{\text{kv\_heads}} = 1$. We run a combination of two ablations: (i) weight initialisation depth-wise scaling (via gains/variance), and (ii) residual branch summation ratios.

For weight *initialisation*, the depth-wise scaling factors are applied to **only** the output linear projection of attention and SwiGLU. We compare three flavours of depth init scaling: `identity` (baseline), `total-depth`, and `relative-depth`, defined by multiplying the gain σ by

$$\sigma* = \begin{cases} 1/\sqrt{2N_{\text{layers}}} & \text{scale by } \texttt{total-depth}, \\ 1/\sqrt{2\,l_i} & \text{scale by } \texttt{relative-depth}, \\ 1 & \text{scale by } \texttt{identity}. \end{cases} \tag{11}$$

where N_{layers} is the total number of Transformer blocks, and $l_i \in \{1, \ldots, 2N_{\text{layers}}\}$ is the relative depth of the current block; σ is the scaled orthogonal gain, $\sigma = \sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}}$, for hidden weights $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$.

Each transformer block is assigned depth 2, since attention and FFN sub-blocks each count as depth 1. When using `relative-depth`, the depth of all FFN blocks can be offset by 1.

For depth-wise residual scaling, we write the residual connection in transformer as:

$$Y \;=\; \alpha \cdot X \;+\; \beta \cdot \text{Block}\big(\text{Norm}(X)\big), \tag{12}$$

where X is the block input and $\text{Block}(\cdot)$ denotes either self-attention or a FFN, and Norm is RMSNorm in our setup.

We consider three depth-wise residual scaling schemes:

$$(\alpha, \beta) = \begin{cases} \left(\frac{2N_{\text{layers}}-1}{2N_{\text{layers}}}, \frac{1}{2N_{\text{layers}}}\right) & \text{scale by } \texttt{depth-normalized}, \\ \left(1, \frac{1}{2N_{\text{layers}}}\right) & \text{scale by } \texttt{completeP}, \\ (1, 1) & \text{scale by } \texttt{identity}. \end{cases} \tag{13}$$

`depth-normalized` Large et al. (2024) scales both the residual and block contributions proportionally to depth. `completeP` Dey et al. (2025) preserves the residual branch while scaling down the block contribution by depth. `identity` corresponds to the conventional unscaled residual formulation.

We fixed batch size (B) to 32 samples, the sequence length to 4096, and the number of training steps to 2048. Experiments were conducted using proxy models with depths $N_{\text{layers}} \in \{2, 16, 64\}$. For all models, we performed a sweep over the learning rate $\{2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 2^{0}\}$.

We report the final-step losses in Table 1, Table 2, and Table 3 for the three depths, respectively, with the two lowest losses highlighted. From the perspective of learning rate transfer, we find that with our optimizer, the optimal learning rate consistently remains around 2^{-2}, regardless of weight initialisation or residual scaling. We also observe that combining `total-depth` weight initialisation with `identity` residual scaling yields a negligible improvement compared to using `identity` weight initialisation.

Table 1: 2 layers ($B = 32$, steps=2048)

Residual init	**Residual multiplier**	**Learning rate η**				
		2^{-4}	2^{-3}	2^{-2}	2^{-1}	2^{0}
total-depth	identity	4.20	**4.11**	**4.09**	4.13	4.22
total-depth	depth-normalized	4.20	**4.12**	**4.11**	4.17	4.21
total-depth	completeP	4.22	**4.15**	**4.16**	4.17	4.28
identity	identity	4.19	**4.10**	**4.10**	4.13	4.23
identity	depth-normalized	4.22	**4.12**	**4.12**	4.13	4.21
identity	completeP	4.21	**4.15**	**4.13**	4.16	4.24
relative-depth	identity	4.20	**4.11**	**4.09**	4.13	4.23
relative-depth	depth-normalized	4.20	**4.13**	**4.11**	4.16	4.25
relative-depth	completeP	4.21	**4.16**	**4.14**	4.18	4.24

Table 2: 16 layers ($B = 32$, steps=2048)

Residual init	**Residual multiplier**	**Learning rate η**				
		2^{-4}	2^{-3}	2^{-2}	2^{-1}	2^{0}
total-depth	identity	3.81	**3.75**	**3.73**	3.77	3.88
total-depth	depth-normalized	3.85	**3.79**	**3.80**	3.84	3.92
total-depth	completeP	3.87	**3.82**	**3.82**	3.85	3.94
identity	identity	3.81	**3.74**	**3.75**	3.79	3.89
identity	depth-normalized	3.83	**3.78**	**3.78**	3.83	3.92
identity	completeP	3.86	**3.81**	**3.81**	3.85	3.94
relative-depth	identity	3.82	**3.79**	**3.74**	3.80	3.90
relative-depth	depth-normalized	3.84	**3.79**	**3.80**	3.83	3.95
relative-depth	completeP	3.88	**3.82**	**3.82**	3.85	3.95

Table 3: 64 layers (B=32, steps=2048)

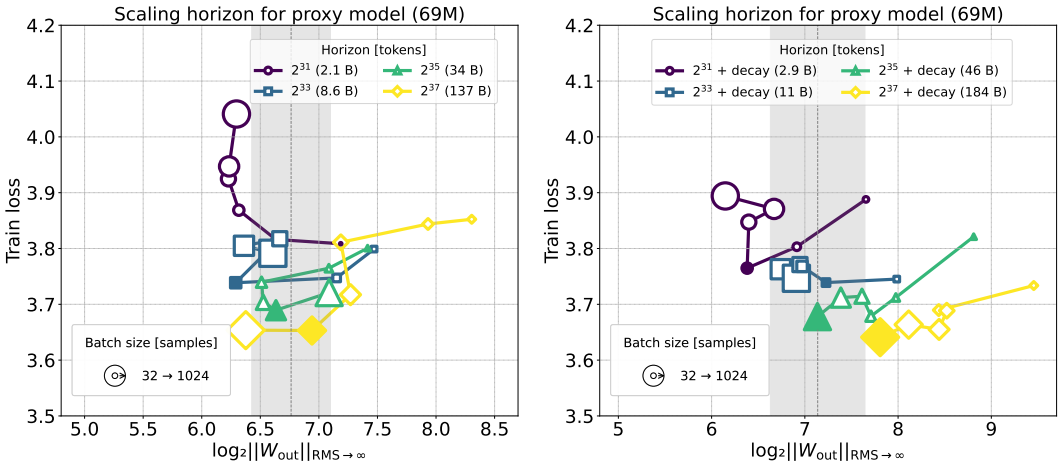
Residual init	**Residual multiplier**	**Learning rate η**				
		2^{-4}	2^{-3}	2^{-2}	2^{-1}	2^{0}
total-depth	identity	3.67	**3.60**	**3.60**	3.65	3.79
total-depth	depth-normalized	3.71	**3.65**	**3.65**	3.69	3.80
total-depth	completeP	3.72	**3.67**	**3.67**	3.72	3.82
identity	identity	3.70	**3.63**	**3.62**	3.66	3.78
identity	depth-normalized	3.70	**3.64**	**3.64**	3.69	3.80
identity	completeP	3.70	**3.70**	**3.67**	3.72	3.82
relative-depth	identity	3.70	**3.61**	**3.61**	3.67	3.82
relative-depth	depth-normalized	3.71	**3.65**	**3.65**	3.69	3.80
relative-depth	completeP	3.72	**3.68**	**3.67**	3.73	3.83

A.12 Ablations on Fig. 1a

A.12.1 Momentum & learning rate decay

In this set of experiments we set momentum to 0.1 (which is by default disabled in the main text) and firstly run the same horizon scaling experiment for the proxy model (69M parameters) with the constant learning rate schedule and evaluate at the same horizons $D = \{2^{31}, 2^{33}, 2^{35}, 2^{37}\}$ as Fig. 1a. The results are presented in Fig. 9a. Here we perform loss smoothing in the same way as for the no-momentum scenario, but do not perform the fitting, i.e. for each batch size we take the optimal norm from the empirically best performing learning rate run. We find that the curves look more like "blobs", where multiple batch sizes give almost the same performance and are centered around the optimal norm (which also transfers across horizons). Also the difference between horizons is not well-pronounced as in the no-momentum scenario.

Then, we add learning rate decay, where we start from checkpoints of the horizons specified above, assume that that constitutes 75% of the total horizon, and linearly decay learning rate to 0 for the rest 25%. Likewise, we smooth loss values and take optimum value per batch size across empirical ones on the learning rate grid. In Fig. 9b we see that there is potentially a slight drift of the optimal norm with horizon scaling. However, after examining individual scans (Fig. 13) we surprisingly found that for long horizons the learning rate decay smooths out the norm optimum: a broad range (factor $\times 4 - 8$ in norm) of learning rates results in the same loss. Hence, there is no longer a single optimal norm, but rather a range, indicating that learning rate decay significantly reduces norm sensitivity. Therefore, we conclude that the seaming drift in Fig. 9b is not significant.

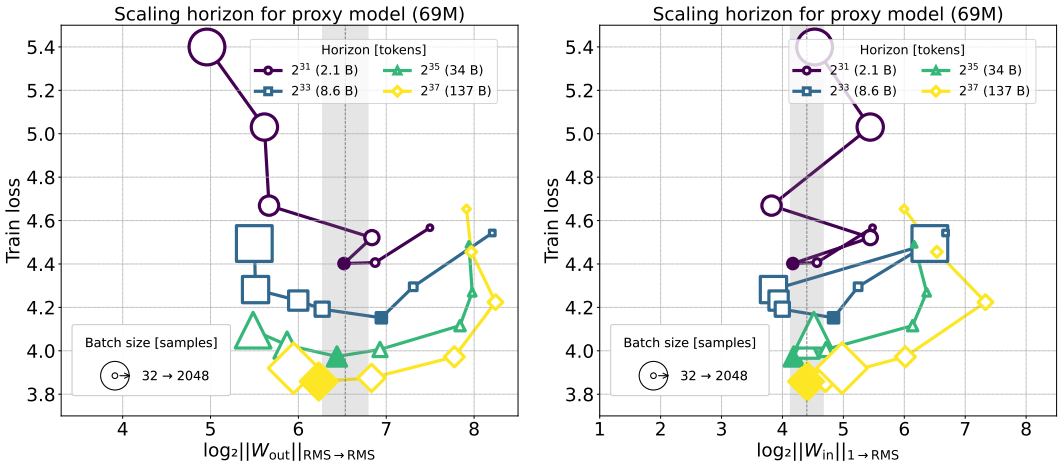


(a) (b)

Figure 9: **Same as Fig. 1a but with momentum = 0.1. (a)** Without learning rate decay. **(b)** With linear learning rate decay to 0 for extra 25% of the total horizon.

A.12.2 NORM CHOICE

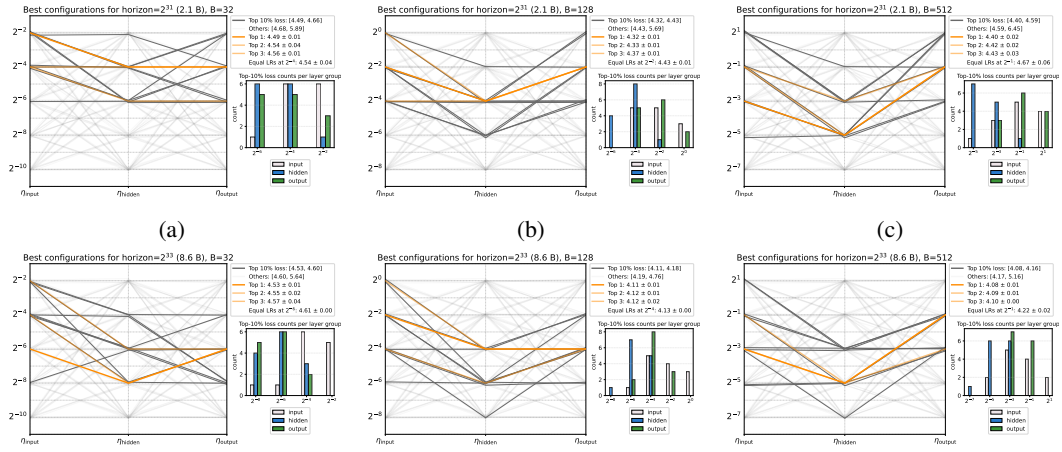
In this Section, we ablate if it is only the output layer norm that induces norm transfer. We replot Fig. 1a, with loss smoothing and without fitting, but now where we use $\|.\|_{\text{RMS}\to\text{RMS}}$ norm of the output or $\|.\|_{1\to\text{RMS}}$ of the input layers instead default $\|.\|_{\text{RMS}\to\infty}$ of the output layer. We observe in Fig. 10 (see also individual norm scans in Fig. 14) that interestingly both norms induce norm transfer.



(a) (b)

Figure 10: **Same as Fig. 1a but with $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\infty}$ norm for the X-axis changed to: (a) $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\text{RMS}}$ (output layer). (b) $\|\boldsymbol{W}_{\text{in}}\|_{1\to\text{RMS}}$ (input layer).**

A.13 LEARNING RATE LAYOUT FOR ADDITIONAL BATCH SIZES AND HORIZONS



(a) (b) (c)

(d) (e) (f)

Figure 11: **Extended version of Fig. 3a with additional batch sizes and horizons.** Top (bottom) row: $D = 2^{31}(2^{33})$ token horizons. Batch sizes, in samples: $B = 32$ (left), $B = 128$ (middle), $B = 512$ (right). Performance is averaged across random seeds as described in Appendix A.2. Note that the optimal B^* is 128 for both $D = 2^{31}$ and $D = 2^{33}$ according to Fig. 2b.

A.14 INDIVIDUAL NORM SCANS AND FITS

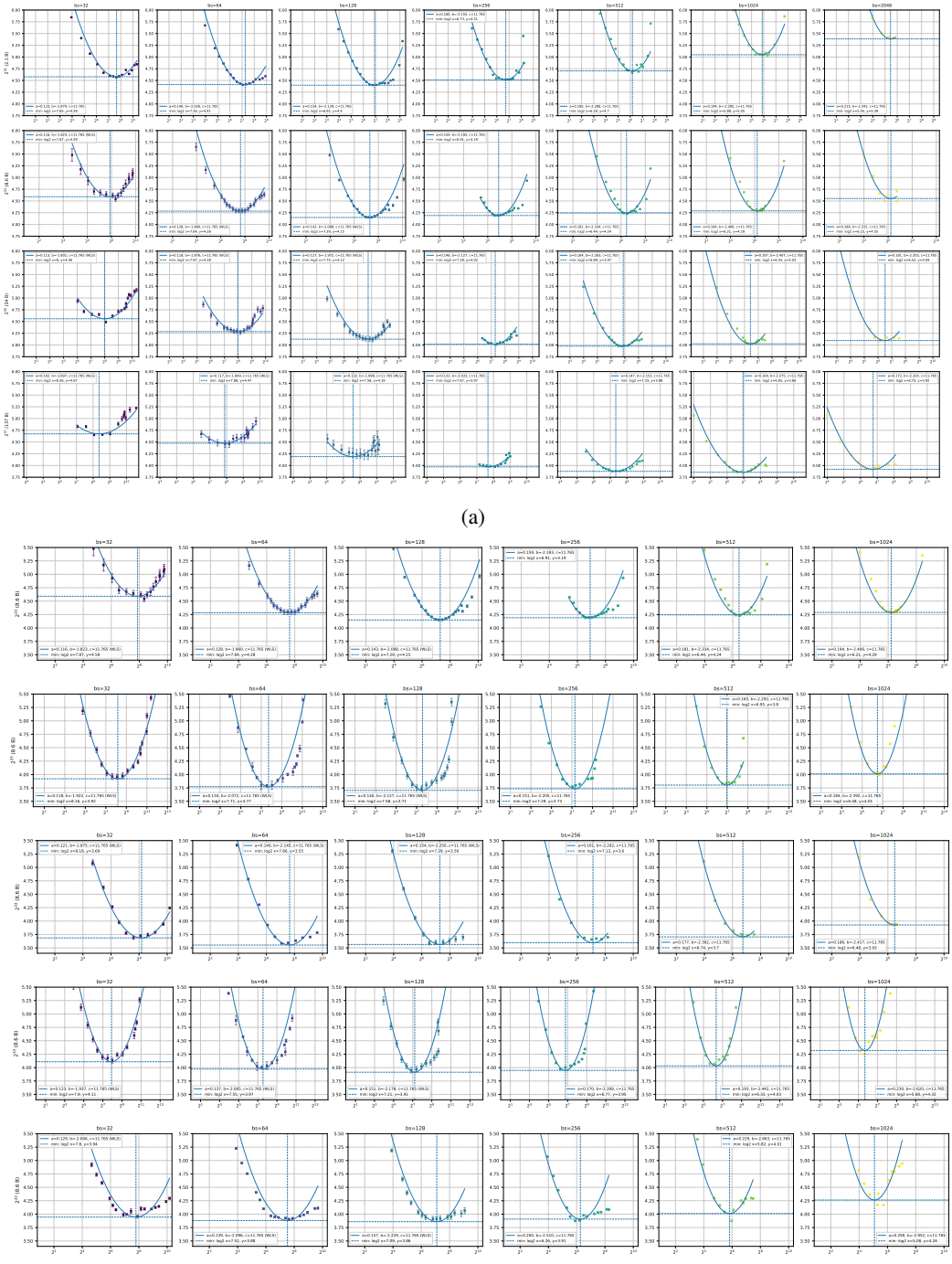
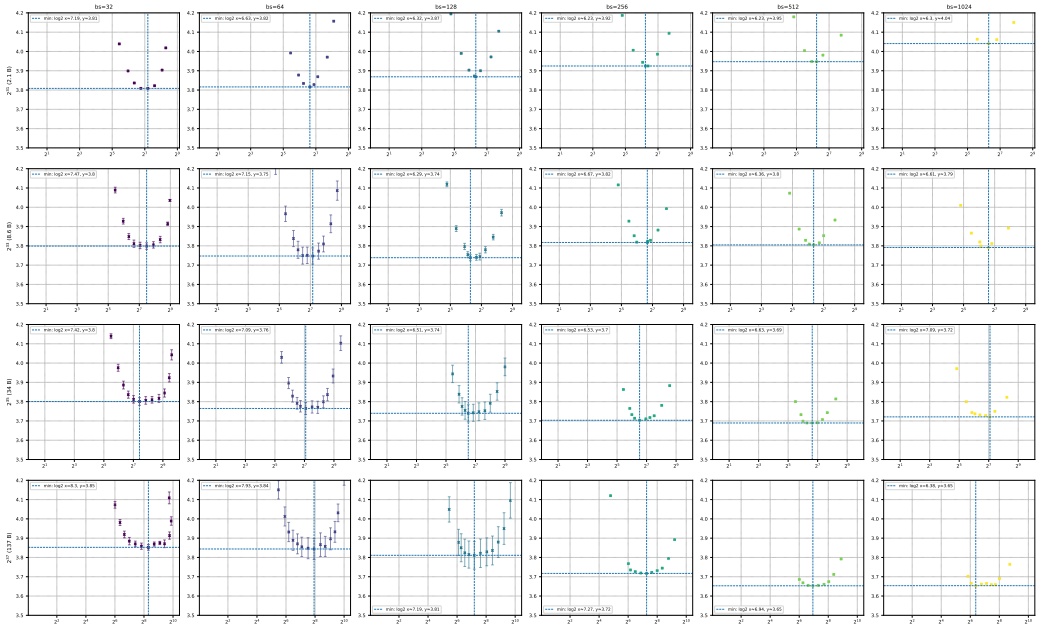
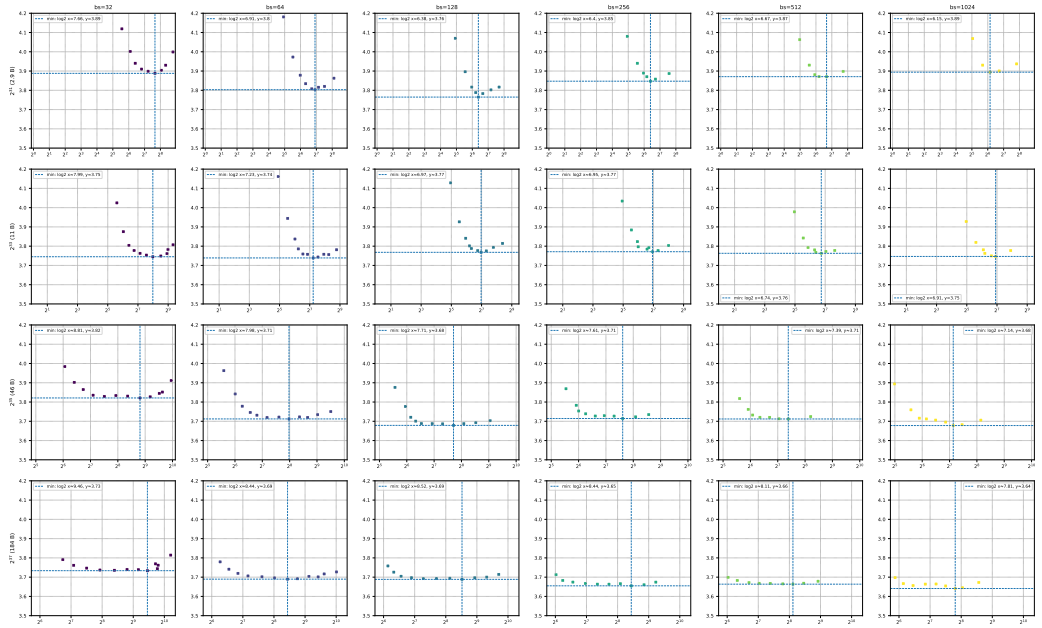


Figure 12: **Individual norm scans for various batch sizes B (columns), across various horizons D in (a), across various models in (b) (rows).** We plot train loss (Y-axis) against the output layer operator norm $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\infty}$, where each point corresponds to a different learning rate run and error bars correspond to loss smoothing variance (see Appendix A.4). The best-loss point for each (B, D) is pinpointed with the blue dashed line, fitted curves are shown with blue solid lines. These fit results are used for: **(a)** Fig. 1a and Fig. 2b, **(b)** Fig. 1b, from top to bottom rows: proxy, $\times 4$-width, $\times 12$-width, $\times 8$-depth, $\times 32$-depth.

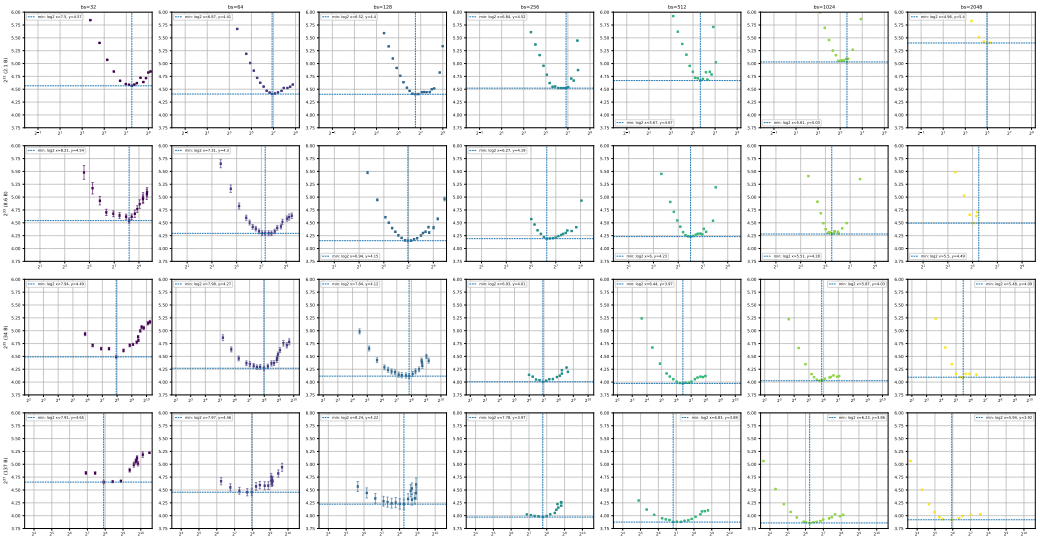


(a)

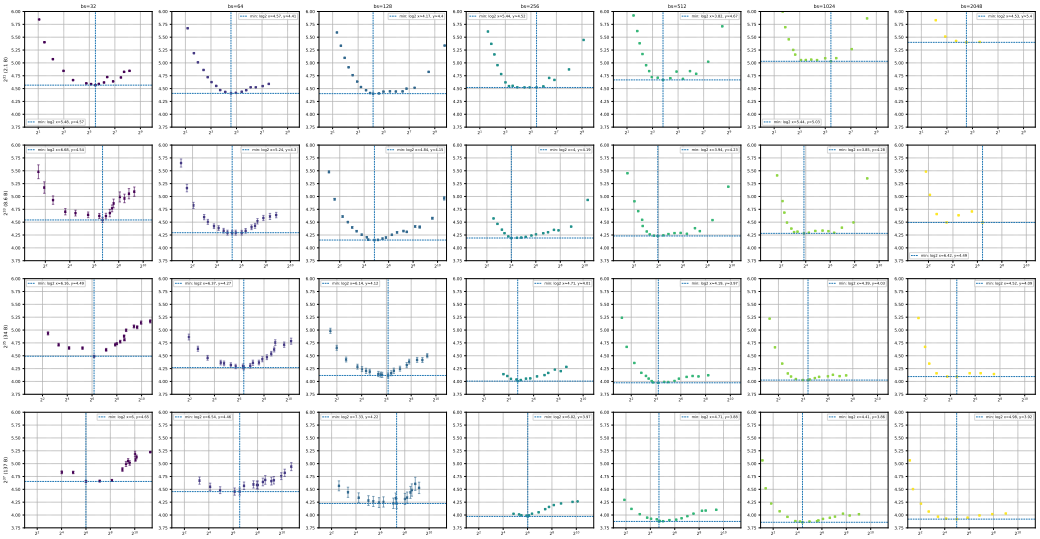


(b)

Figure 13: **Individual output norm $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\infty}$ scans for various batch sizes B (columns) across various horizons D (rows). (a)** with momentum = 0.1, no learning rate decay. **(b)** with momentum = 0.1, with learning rate decay linearly to 0 for 25% of total horizon.



(a)



(b)

Figure 14: **Individual norm scans for various batch sizes B (columns) across various horizons D (rows). (a)** For $\|\boldsymbol{W}_{\text{out}}\|_{\text{RMS}\to\text{RMS}}$ (output layer). **(b)** For $\|\boldsymbol{W}_{\text{in}}\|_{1\to\text{RMS}}$ (input layer).