Critical Batch Size Revisited: A Simple Empirical Approach to Large-Batch Language Model Training

William Merrill **Shane Arora** **Dirk Groeneveld** **Hannaneh Hajishirzi**
Allen Institute for AI
willm@allenai.org

Abstract

The right batch size is important when training language models at scale: a large batch size is necessary for fast training, but a batch size that is *too large* will harm token efficiency. To navigate this tradeoff, McCandlish et al. (2018) suggest that a *critical batch size* (CBS), below which training will not substantially degrade loss, can be estimated based on the gradient noise scale during training. While their method has been adopted in practice, e.g., when training GPT-3, strong assumptions are required to justify gradient noise as a proxy for the CBS, which makes it unclear whether their approach should be trusted in practice, limiting its applicability. In this paper, we introduce a simple, empirical approach to *directly* measure the CBS and show how the CBS evolves over training. Applying our approach to the OLMo models, we find that CBS is near 0 at initialization, increases rapidly at first, and then plateaus as training progresses. Furthermore, we find that this trend holds across different model sizes (1B and 7B), suggesting CBS from small training runs can inform larger-scale training runs. Our findings about how the CBS changes over training motivate *batch size warmup* as a natural way to reliably train language models at large batch size: start the batch size small and increase it as the CBS grows. To validate this claim, we use batch size warmup to train OLMo 1B to slightly better loss than the original training run with 43% fewer gradient steps. This shows how our framework can be applied to reliably train language models at larger batch sizes, increasing data parallelism without compromising performance.

1 Introduction

Increasing the throughput of training is important for training large models. A natural way to increase throughput is by increasing data parallelism, i.e., increasing the *batch size* used during training so that more data can be processed at once and the number of sequential gradient steps can be decreased. However, naively picking a very large batch size can degrade the performance achieved by a fixed token budget, as larger batches can show diminishing returns in their ability to estimate the population gradient. Thus, in order to confidently train language models with higher token throughput, it is important to develop theoretical and empirical understanding of large-batch training.

One fundamental concept for large-batch training methodology is the following:

Critical Batch Size Hypothesis (McCandlish et al., 2018)

There is some critical batch size B^* up to which increasing the batch size (and appropriately modifying the learning rate) approximately preserves the loss trajectory as a function of tokens trained, but, above which, the loss trajectory degrades.

If such a CBS B^* exists (and we can measure it), it represents a reasonable balance between efficiency and performance (i.e., loss) and is thus a practically useful batch size at which to train. Working in a

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

 https://doi.org/10.52202/085713-3898

simplified theoretical setup, McCandlish et al. (2018) derive a correspondence between the CBS and the *gradient noise scale*, i.e., the variance of the per-example gradients from the training distribution. They suggest that an estimator for the gradient noise scale should be used in practice as a proxy for the CBS, and this can in turn be used to set the batch size for large-scale pretraining runs. The noise scale also appears to have been adopted in practice as a proxy for the CBS, having been mentioned explicitly in the GPT-3 technical report (Brown et al., 2020), and inspiring a flurry of methodological innovations for better noise scale estimation (Gray et al., 2023, 2024).

While appealing, the link between the gradient noise and the CBS requires several strong assumptions to justify: specifically, it assumes the SGD optimizer and that gradients are well-conditioned (cf. Section 2). Thus, it is unclear whether the noise scale should be a meaningful proxy for the CBS for language model pretraining in practice, where the Adam optimizer is often used and the optimization may not be well-conditioned. To this end, we aim to address the following practical questions that remain for effectively leveraging the CBS viewpoint to train language models at larger batch sizes:

1. How can we measure the CBS cheaply with minimal assumptions before launching a pretraining run?
2. How does the CBS change over the course of pretraining and as a function of model size?
3. Having measured the CBS, how should we adapt the batch size, learning rate, and other parameters over the course of a pretraining run?

This work documents our attempts to answer these questions in order to operationalize the CBS for large-batch training. We focus our investigation on the OLMo models (Groeneveld et al., 2024; OLMo et al., 2025), due to their open weights and data, making the following contributions:

1. First, we introduc an empirical method to directly measure the CBS via **branched training**. Our method avoids strong assumptions needed to justify the noise scale method from prior work. This lets us trust it more than the noise scale method, which we find unreliable.
2. We use our method to study how our CBS measurement changes over the course of training, finding it improves rapidly initially but than flattens off. Further, we find that the CBS not depend strongly on model size, in line with past findings using different methodology (Zhang et al., 2024).
3. Our knowledge of the local CBS across training checkpoints suggest a natural **batch size warmup** strategy for large batch training: begin training with a small batch size and double it whenever the CBS increases sufficiency. We use this strategy to train 1B parameter models with 43% fewer gradient steps without degrading (and, in fact, slightly improving) final loss.

Overall, our empirical framework for measuring and leveraging the CBS provides simple and principled methodology for improving the efficiency of large-scale training runs and addressing other fundamental questions in the science of language model pretraining.

2 Background: Estimating CBS via Gradient Noise Scale

Past empirical work has aimed to measure the CBS by launching many training runs to the same target loss, which is expensive (Zhang et al., 2019, 2024), or by using the gradient noise scale as a proxy (McCandlish et al., 2018; Gray et al., 2023, 2024). In contrast, we will introduce a new method that uses a small amount of additional training to estimate the CBS, which is less expensive than launching many full training runs and does not make any strong assumptions like the noise scale. Before introducing our method, we review the noise scale framework used to estimate the CBS in prior work and the underlying assumptions it relies on.

McCandlish et al. (2018) suggest that the CBS can be measured in terms of the gradient noise scale, i.e., the variance of the gradients within a batch. Concretely, their recommendations to measure the CBS and adapt the learning rate are as follows:

Existing Method: Noise Scale Proxy for CBS (McCandlish et al., 2018)

Let G be the full gradient and let Σ be the covariance matrix for the gradient across data examples. We first compute $\mathcal{B}_{\text{simple}}$ as a proxy for the CBS (using an efficient statistical estimator):

$$\mathcal{B}_{\text{simple}} = \frac{\text{tr}(\Sigma)}{\|G\|^2}.$$

We set the modified batch size to $\mathcal{B}_{\text{simple}}$ and *linearly* scale the learning rate η^*:

$$B^* = \mathcal{B}_{\text{simple}} \tag{1}$$

$$\eta^* = \frac{B^*}{B} \cdot \eta. \tag{2}$$

This viewpoint is attractive due to its simplicity and tractability, and it has inspired improved methods for estimating the noise scale (Gray et al., 2023, 2024). However, the link between noise scale and CBS requires several strong assumptions to justify, which we will argue motivates revisiting other approaches for measuring the CBS. McCandlish et al. (2018) consider training a model on a loss surface where the loss landscape is well approximated by its second-order Taylor expansion. The first crucial assumption in their analysis is that optimizer used to decrease the loss is SGD:

Assumption 1: SGD Optimizer (McCandlish et al., 2018)

The step taken to reduce the loss is a noisy estimate of the true gradient, computed via B samples.

This assumption may seem benign, but it is worth noting that it is *not* typically met in practice, as LMs are trained with the Adam optimizer (Kingma and Ba, 2017). Moreover, by analyzing training dynamics in terms of stochastic differential equations (Li et al., 2021), Malladi et al. (2022) argue theoretically that Equation (2) is an appropriate scaling rule for SGD, but not for Adam, where a *square-root* scaling rule is more principled. The square-root scaling for Adam is also supported by the theoretical analysis of Li et al. (2024, Equation 4). Thus, when training with Adam, it seems that the linear scaling rule assumed by McCandlish et al. (2018) should not apply.

A more fundamental issue is that McCandlish et al. (2018) also require another strong assumption to derive the noise scale method for estimating the CBS. In general, their noise-scale-based estimate of the CBS has a more complex form than $\mathcal{B}_{\text{simple}}$ involving the Hessian H:

$$\mathcal{B}_{\text{noise}} = \frac{\text{tr}(\Sigma H)}{G^\top H G}.$$

In order to justify that B^* can be computed as $\mathcal{B}_{\text{simple}}$, McCandlish et al. (2018) assume:

Assumption 2: Well-Conditioned Optimization (McCandlish et al., 2018)

The Hessian H is a multiple of the identity matrix. It follows that $\mathcal{B}_{\text{noise}} = \mathcal{B}_{\text{simple}}$.

This is a strong assumption required to justify using $\mathcal{B}_{\text{simple}}$ as a proxy for the CBS because computing Hessians would be too expensive to be practice. McCandlish et al. (2018) suggest informally that, without this assumption, $\mathcal{B}_{\text{simple}}$ *may* still be correlated with $\mathcal{B}_{\text{noise}}$, but it is not obvious why this should be the case. Even if this is true, it still poses a real problem for the noise scale methodology, since the goal of the method is to produce an *absolute* measure of B^*. It is unclear for practitioners what coefficient should be used to translate $\mathcal{B}_{\text{simple}}$ to B^*—and, more fundamentally, whether it is even valid to assume that such a coefficient exists.

3 Our Method: Measuring the CBS via Local Branched Training

As discussed in Section 2, using the gradient noise scale $\mathcal{B}_{\text{simple}}$ as a proxy to estimate the CBS relies on several strong assumptions. In light of this, it unclear whether we can trust the gradient noise scale as a proxy for CBS. We thus argue that we should instead aim to measure the CBS *directly*, without the need for any strong assumptions to justify an indirect proxy. After introducing our measurement

https://doi.org/10.52202/085713-3898

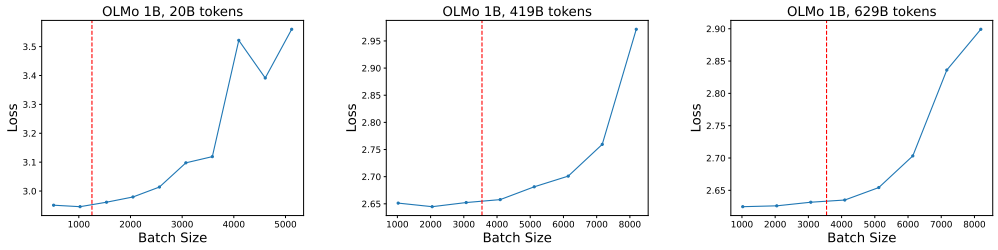


Figure 1: Smoothed final loss after branched training at particular checkpoints, with B^* shown as the dotted red line. Each point represents the loss achieved by a specific branched training run after 2B tokens. Our method detects the point at which loss starts to increase, heuristically tolerating noise within $\epsilon = 0.01$. These plots show how this plays out for three particular checkpoints; see Appendix A for loss curves for all checkpoints.

approach in this section, we will show in Section 4 how these measurements can be applied to train language models to the same (or better) target loss with fewer gradients steps.

3.1 Method

We introduce a simple **branched training** method that directly approximates the CBS by launched branched training runs from a checkpoint, which allows us to identify B^* as the largest batch size that does not degrade in loss relative to smaller batch sizes as visualized in Figure 2. To make this tractable, we train only for a fixed token budget Δ, assuming that if B^* recovers in loss by Δ, its loss will continue to match smaller batch sizes onwards as well. This allows us to estimate the CBS with only a small amount of additional training (controlled by Δ). Further, as we will find later in Figure 2, the CBS trend remains consistent across model sizes, so CBS measurements with small models could be used to inform large-scale training runs.

Our Method: Branched Training to Measure CBS

Given a training checkpoint with original batch size B and learning rate schedule η, we aim to measure the critical batch size B^*. Let $f(\eta)$ be the learning rate scaling rule: $f(k) = k$ for SGD and $\sqrt{k}$ for Adam. We create several training branches with modified batch size $k \cdot B$ and learning rate $f(k) \cdot \eta$ and train for a small number of tokens Δ to get loss L_k—following standard practice, we take L_k to be the smoothed loss. We then define k^* as the maximum k such that, for all $k < k^*$, $L_{k^*} \leq L_k + \epsilon$, where ϵ is a tolerance parameter for "similar" losses. We then define the CBS B^* and scaled learning rate η^* as

$$B^* = k^* \cdot B$$
$$\eta^* = f(k^*) \cdot \eta.$$

Our method will empirically estimate the largest batch size B^* at which the local optimization trajectory recovers roughly to its original loss after Δ steps. In contrast to the strong assumptions needed to justify the gradient noise scale, the only (weak) assumption our method relies on is:

Assumption 3: Local Recovery

If the loss achieved by batch size B^* recovers to match the loss with batch size $B < B^*$ after training for Δ tokens, the loss trajectories will remain the same beyond Δ as well.

Implementation Details. Our method requires specifying two parameters: the window size Δ and the loss tolerance ϵ. We also apply smoothing to the loss to reduce noise. In more detail:

1. **Window Size.** Because the optimizer state must update when the batch size is changed, we expect an immediate spike in the loss after adjusting the batch size. The window size Δ represents the number of steps we are willing to wait for the loss to recover from this bump. The CBS measurements could, in principle, depend on Δ, with larger values of Δ

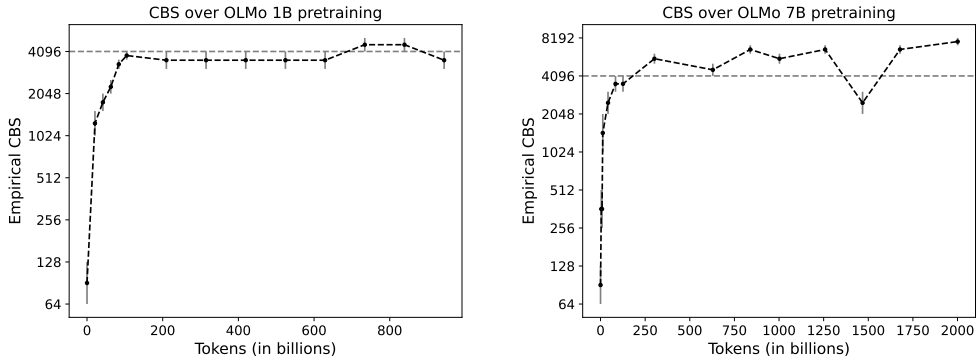


Figure 2: CBS over training for OLMo 1B and 7B, measured in documents (4096 tokens per document). The qualitative trend is similar across both model sizes. The CBS starts near 0, grows rapidly but diminishingly, and plateaus around 4096.

potentially producing larger CBS estimates. We set Δ to 2B tokens, which we take as a small, conversative window size relative to our overall pretraining budget of 600B tokens.

2. **Loss Tolerance.** Viewing loss as a function of batch size multiplier k (cf.Figure 1), we need a way to determine whether the loss at k^* has increased relative to all $k < k^*$. We operationalize this with a tolerance parameter ϵ, which we set to 0.01 arbitrarily. In principle, tolerance could be set in a more principled way using a statistical test in future work.
3. **Loss Smoothing.** Pretraining loss is noisy at the batch level, so, following standard practice, we apply exponentially moving average smoothing with parameter α set to 0.5.

It is worth comparing our method to other work that empirically measures the CBS. Our method of training for a fixed token budget Δ and measuring the change in loss can be understood as the dual view to measuring the number of steps required to achieve a target loss, which has been used in prior work (Zhang et al., 2019, 2024). Reformulating the measurement in this way has the nice property that the training budget can be fixed in advance. Under the local recovery assumption, it also allows us to train for only Δ tokens, which means we do not have to launch full training runs for each batch size. Finally, unlike Zhang et al. (2024), we apply our method at various checkpoints throughout training, whereas they apply it only from initialization. This means we can estimate the *local CBS* at a specific point in training rather than just the *global CBS*.

3.2 Experimental Setup

We aim to measure the CBS over the course of model training and the role of model size. Because our method requires pretraining checkpoints and access to the pretraining data, we use the OLMo 1B and OLMo 7B models for our experiments (OLMo et al., 2025), whose pretraining data is openly available (Soldaini et al., 2024). For each model, we take a variety of checkpoints over the course of training, allowing us to assess how the CBS changes over the course of training; see Appendix A for more details. We also compute the noise scale across training checkpoints using the estimator proposed by McCandlish et al. (2018) to assess whether it is a valid proxy for the CBS we measure.

We define the interval for the CBS at each checkpoint by choosing B^* (as defined in Section 3) as a lower bound, and by picking the least $k > k^*$ as an upper bound. The plotted point represents the geometric mean of these interval endpoints. We measure the CBS in documents, with a pretraining sequence length of 4096 tokens per document.

3.3 Results: CBS Over Training and Across Model Sizes

Figure 2 shows the CBS B^* measured via Section 3.1 across training checkpoints for OLMo 1B and 7B. The CBS increases over training in a similar way for both model sizes: the CBS starts near 0, grows rapidly within the first 50k tokens, and then plateaus around 4096.

Impact of Model and Data Size. Prior work has suggested that the CBS is largely independent of model size, scaling primarily with data size (Zhang et al., 2024; Bergsma et al., 2025). This is largely

https://doi.org/10.52202/085713-3898

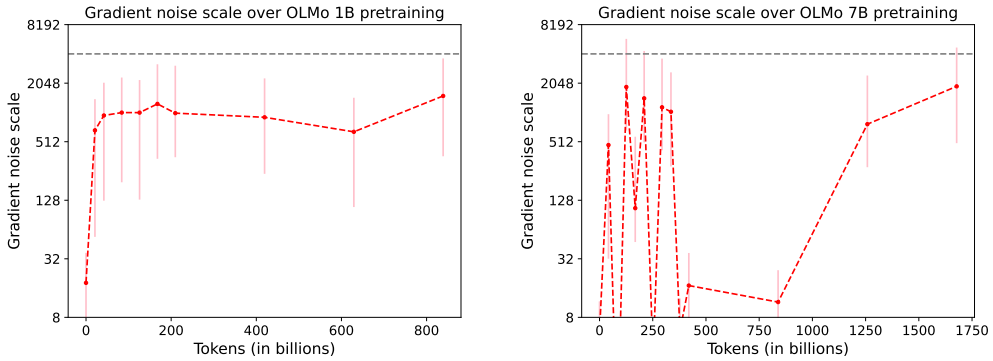


Figure 3: Gradient noise scale for OLMo 1B and 7B computed via the estimator of McCandlish et al. (2018) with 95% confidence intervals; details in Appendix B. The gradient noise scale underestimates the CBS (cf. Figure 2) and the qualitative trend does not clearly match, especially for OLMo 7B.

consistent with our findings, as the CBS curves like qualitatively similar at the 1B and 7B scales. In addition, the fact that the CBS grows over training suggests that the "aggregate" CBS should also increase as we train on more data since the average CBS over the course of training should increase. We elaborate on this in Appendix D: while CBS growth over training predicts that aggregate CBS should increase with data size, it is unclear whether the CBS growth pattern we observe would predict the aggregate CBS $\propto \sqrt{D}$ scaling law found in prior work (Zhang et al., 2024; Bergsma et al., 2025).

Comparison with Gradient Noise Scale. As discussed in Section 2, the gradient noise scale has been proposed as a proxy to measure the CBS (McCandlish et al., 2018), though this connection relies on strong assumptions to justify. We thus empirically compare our measurement of the CBS to the gradient noise scale. Figure 3 shows that, for both OLMo 1B and 7B, the gradient noise scale underestimates the CBS by several orders of magnitude. Furthermore, especially for OLMo 7B, the qualitative trend does not match the CBS. For OLMo 1B, the qualitative pattern is more similar. However, since this similarity is not found for both models, we conclude that, in general, the noise scale cannot be used reliably as a proxy for the CBS.

Motivating Batch Size Warmup. A central takeaway from these results is that the CBS starts near 0, grows rapidly, and then plateaus. This suggests that *batch size warmup*, where the batch size is dynamically increased over the beginning of a training run, is a natural way to increase the effective batch size for most of training while avoiding training with a batch size above the CBS. In the next section, we will discuss our implementation and validation of this idea.

4 Application: Batch Size Warmup for Larger Batch Training

A straightforward way to speed up training is to increase the batch size, and, as long as the new batch size is less than the CBS, we can be confident that the loss will be minimized about as effectively as before. Thus, we can leverage knowledge of the CBS to speed up training.

Furthermore, our local knowledge of the CBS can be leveraged to do this more effectively than if we only had a global sense of the CBS. If we simply increased a fixed batch size, this would mean that we are training with a batch size that is too large for a short period at the beginning of training, which could potentially destabilize training or degrade final performance. To get around this, we can use our measurement of the local CBS to "warm up" the batch size. We will train at a smaller batch size for the beginning of training when the CBS is small, and then switch to a larger batch size once the CBS grows large enough. More generally, we can aim to double the batch size whenever we determine the CBS has doubled. This should allow us to benefit from training at a larger batch size for *most* of training without training at a batch size that is too large at the beginning of training, which could degrade the final loss achieved by the training run.

4.1 Methodology for Batch Size Warmup

Given an existing training run (at a small batch size), we aim to adapt it to achieve the the same final loss with fewer overall gradient steps (i.e., a larger batch size for most of training). Assuming an original fixed batch size B and base learning rate η, our method for training with *batch size warmup* is as follows. First, we use branched training CBS method (Section 3) to measure B_t^*, the CBS after training for t tokens, using an existing checkpoint. When training a new model, we set the batch size and and learning rate as follows:

Batch Size Warmup

1. At $t = 0$, initialize the batch size to $B_0 = B$ and base learning rate to $\eta_0 = \eta$.
2. After training for t tokens, if we determine that the CBS exceeds the current batch size ($B_t^* > 2B_t$), we double the current batch size and update the base learning rate following the square-root scaling rule (cf. Section 3):

$$B_{t+1} = 2B_t$$
$$\eta_{t+1} = \sqrt{2}\eta_t.$$

This method will ensure that the batch size will increase over training *safely*, i.e., never exceeding the CBS. Since we found that the CBS increases quite rapidly at the beginning of training, this method will double the batch size twice early in training, reaching a maximum batch size of 4096 by 503B tokens. This means that we will effectively train at a larger batch size for most of training compared to the original small-batch run. But, crucially, we can be confident that the batch size will never increase our batch size above the CBS. Thus, we expect the final loss will be comparable to that of the original training across thet training trajectory.

Implementation Details. The main implementation question is how to operationalize checking $B_t^* > 2B_t$ in order to double the batch size. In practice, we do this heuristically based on the measurements in Figure 2: since we only double the batch size twice over training, this involves just picking two thresholds. In addition, this process could be automated for future training runs using online measurements of the CBS paired with some kind of curve fitting or statistical test.

Design Choices. As defined above, our method uses a square-root scaling rule, which is well-motivated for Adam (Malladi et al., 2022), but in principle a linear scaling rule could also be used. As batch size warmup only modifies the base learning rate, it is compatible to overlay with existing learning rate schedules (in practice, the models we use follow a cosine schedule). Finally, we choose to only increase the batch size at powers of two because it is not clear that having a more precise match to the CBS beyond the order-of-magnitude level would be particularly useful and because the OLMo codebase requires the number of GPUs g divides the batch size. Thus, doubling is convenient because it easily guarantees the new batch size remains a multiple of g. However, in principle one could update more frequently, e.g., by setting updating the batch size the largest multiple of $g \leq B_t^*$.

Connection to Existing Methods. Our batch size warmup method is similar to preliminary experiments by McCandlish et al. (2018, Appendix D) using a dynamic batch size on the SVHN dataset. However, they use their noise scale method (Section 2) to set the batch size, which is potentially unreliable, and only consider small-scale classification. In the context of image classification, prior work explored replacing learning rate decay with increasing the batch size (Smith et al., 2018). This is conceptually related to our batch size warmup approach in that it leverages scaling rules between batch size and learning rate to achieve larger batch training. However, our method can apply on top of existing learning rate schedules, can generalize to Adam (vs. SGD), and, most crucially, ensures that the batch size number exceeds the critical batch size.

4.2 Experimental Setup

We evaluate the viability of training with batch size warmup via the following training runs. For all models, we use the default OLMo 1B pretraining hyperparameters unless stated otherwise.

1. **Batch Size Warmup:** We train OLMo 1B with our batch size warmup method as detailed in Section 4.1. We initialize the batch size to 1024. Based on a manual reading of the CBS

https://doi.org/10.52202/085713-3898

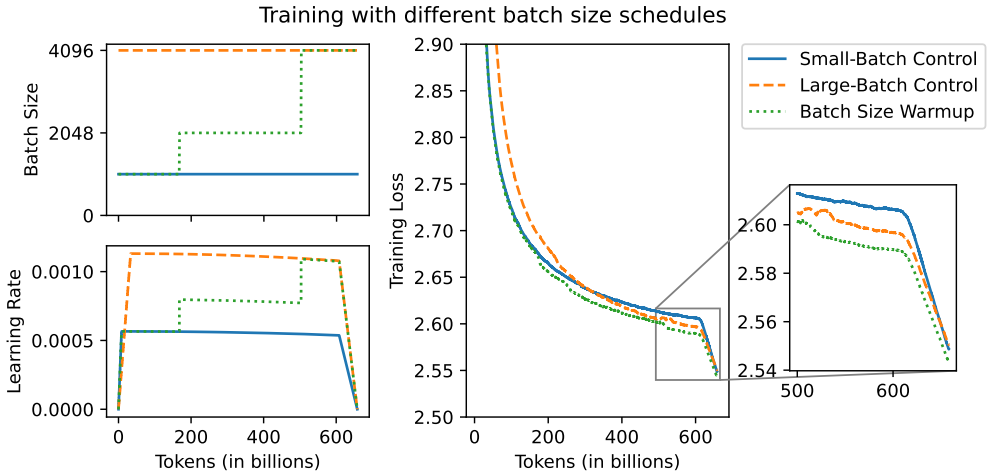


Figure 4: Batch size schedule (left, top), learning rate schedule (left, bottom) and training loss (right) for the pretraining of an OLMo model with different batch size schedules. Training loss is smoothed by taking the moving average over the past 10B tokens.

Method	↓ PT Loss	↓ MT Loss	↑ Grad. Steps Saved
Batch Size Warmup (Ours)	**2.5891**	**2.5433**	43%
Small-Batch Control	2.6057	2.5486	0%
Large-Batch Control	2.5962	2.5506	**75%**

Table 1: Loss after pretraining (PT) and mid-training (MT) for each run, averaged over the past 10B tokens and the percentage of gradient steps saved vs. the small-batch control (including annealing steps). Both before and after annealing, batch size warmup slightly outperforms both controls in loss.

measurements in Section 3, we determine that the CBS reaches 2048 by 168B tokens and 4096 by 503B tokens.[1] Our method thus doubles the batch size at each of these points.

2. **Small-Batch Control:** We train OLMo 1B with batch size $B = 1024$ and base learning rate $\eta = \sqrt{2} \cdot 0.0004$.[2] We expect that our batch size warmup method should be able to achieve similar final loss to the small-batch control with fewer gradient steps.
3. **Large-Batch Control:** We train OLMo 1B with a fixed large batch size of $B = 4096$ and base learning rate $\eta = 2\sqrt{2} \cdot 0.0004$. Since its batch size will exceed the critical batch size for the initial part of training, we expect that the large-batch control will show degraded loss compared to our batch size warmup method.

Evaluations. Language model training consists of two steps, pre-training and mid-training (OLMo et al., 2025). We evaluate training loss as well as several measures of out-of-distribution loss at the end of the pre-training stage as well as after the mid-training stage.

- **Loss after Pretraining.** We evaluate the loss at the end of the pre-training stage for all models. The original training run for OLMo 1B ran for 4T tokens, so we do not have the

[1]These thresholds were determined from preliminary CBS measurements at an earlier stage of the project. Our measurements changed slightly after this point, but we deemed that it was not worth it to restart the expensive training runs because we do not think the results should be sensitive to a precise choice of threshold. Going forward, it could be useful to establish systematic methodology for choosing batch size warmup thresholds given the CBS measurements.

[2]The default hyperparameters for OLMo 1B use a batch size of 512 and learning rate of 0.0004. Since we chose an initial base size of 1024 for our experiments, we adapted the learning rate appropriately via the square-root scaling rule (Malladi et al., 2022).

Method	↓ Task BPB		↓ C4		↓ Pile	
	PT	MT	PT	MT	PT	MT
Batch Size Warmup (Ours)	1.0316	1.0076	**2.8049**	**2.7597**	**2.1916**	2.1521
Small-Batch Control	**1.0112**	**0.9999**	2.8196	2.7622	2.2073	**2.1471**
Large-Batch Control	1.0571	1.01927	2.8107	2.7658	2.1996	2.1586

Table 2: According to the method from Bhagia et al. (2024), we evaluate downstream performance via BPB on downstream tasks, as well as cross-entropy loss on two held-out sets, C4 and the Pile, both after pretraining and after mid-training. Batch-size warmup generally performs comparably or better compared to the small-batch control, suggesting it does not degrade downstream performance.

resources to replicate this full training run. Instead, we pre-train for 608B tokens using the original learning rate schedule for the longer run.

- **Loss after Mid-Training.** We report loss at the end of mid-training stage, which more closely reflects how these checkpoints are used in language model training. Specifically, starting from final pretraining checkpoint, we linearly anneal the remaining learning rate down to 0 for 50B tokens, keeping the batch size fixed at its final value (Figure 1). Because mid-training is a part of the standard language modeling pipeline, we take the loss after mid-training to represent the loss that would be achieved by these training runs in a practical context. Furthermore, OLMo et al. (2025) suggest that this kind of learning rate annealing can induce significant gains in loss for partial pretraining runs. We thus also take the loss after mid-training loss as a proxy for how these runs would compare if we were to fully train them for the full learning rate schedule.
- **Out-of-Distribution Losses.** To measure performance beyond the training loss, we also track three kinds of out-of-distribution losses. The first two are straightforward cross-entropy losses on common pretraining datasets, C4 (Dodge et al., 2021) and The Pile (Gao et al., 2020). The third out-of-distribution loss follows the method from Bhagia et al. (2024), which argues for computing the loss (in bits-per-byte; BPB) on the correct answers of multiple popular question-answering datasets such as ARC-Easy, ARC-Challenge, MMLU, etc. For the full list of datasets used, see Appendix E.

4.3 Batch Size Warmup Results

We compare batch size warmup to the small and large-batch controls at two points: first, after the pretraining stage and, second, after the mid-training stage that anneals the remaining learning rate down to 0. After pretraining, batch size warmup reaches lower loss than both controls, as shown in Figure 2 and Table 1, outperforming the small-batch control in loss by a margin of 0.0166. After mid-training, batch size warmup still slightly outperforms the small-batch control in loss, this time by a smaller margin of 0.0053. The large-batch control now achieves the worst overall loss. Overall, batch size warmup slightly exceeds the small-batch control in loss while using 43% fewer gradient steps. In contrast, the large-batch control uses 75% fewer gradient steps, but its final loss degrades compared to the small-batch control. Thus, we find batch size warmup is a reliable method to train with fewer gradient steps without degrading final loss.

Beyond the final loss, we also evaluate whether batch size warmup leads to similar downstream performance as small-batch training using our downstream-task BPB evaluation as well as validation loss on C4 and the Pile. As shown in Table 2, we find that BPB on validation sets is broadly competitive with the small-batch control, performing best on three out of four conditions considered. With the caveat that precisely measuring the downstream impact of pretraining decisions is difficult, this suggests that, just as batch-size warmup does not degrade final loss, it should not degrade downstream measures of performance either.

5 Conclusion

In this work, we introduced a simple empirical method for estimating the CBS throughout language model pretraining runs, which can be used to increase batch size (and thus effective token throughput) for large scale training runs without sacrificing performance. As we discussed, the existing noise

https://doi.org/10.52202/085713-3898

scale method (McCandlish et al., 2018) for estimating the CBS requires strong assumptions to justify, which our approach avoids. We used our method to study the evolution of the CBS during training for the OLMo models, finding that CBS increases monotonically but diminishing over the course of training, and that CBS does not seem to depend on model size, in line with prior work (Zhang et al., 2024). Guided by these findings, we showed that our measurements could be used to pick a **batch size warmup** schedule that enables larger batch training without harming final training loss. We take these results to demonstrate the validity and utility of our CBS measurement approach, and believe our framework could be useful for increasing the efficiency of future large-scale pretraining efforts.

There are several details and extensions of our method that would be interesting to explore going forward. First, it would be interesting to carry out a more systematic analysis of the impact of the hyperparameter Δ on CBS measurements: how sensitive is the method to Δ and do different values for Δ potentially bias the measurement? It would also be interesting to further investigate the conditions under which noise scale might be a meaningful CBS proxy and used for batch size warmup. When applying our CBS measurements to picking a batch size warmup schedule, we manually picked doubling threshold based on the CBS. Going forward, it would be useful to establish a systematic way to set threshold for increasing the batch size given CBS measurements. There are also other potential methodological improvements, such as removing the arbitrary power of 2 constraint and estimating the CBS in an online fashion. Overall, these improvements would allow batch size warmup to be applied more robustly and easily across different pretraining setups.

Acknowledgments

We thank Joel Hestness, Sadhika Malladi, and Ananya Harsh Jha for discussions.

References

J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, and C. Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

S. Bergsma, N. Dey, G. Gosal, G. Gray, D. Soboleva, and J. Hestness. Power lines: Scaling laws for weight decay and batch size in llm pre-training, 2025. URL https://arxiv.org/abs/2505.13738.

A. Bhagia, J. Liu, A. Wettig, D. Heineman, O. Tafjord, A. Jha, L. Soldaini, N. A. Smith, D. Groeneveld, P. W. Koh, J. Dodge, and H. Hajishirzi. Establishing task scaling laws via compute-efficient model ladders. *ArXiv*, abs/2412.04403, 2024. URL https://api.semanticscholar.org/CorpusID:274514987.

Y. Bisk, R. Zellers, R. Le bras, J. Gao, and Y. Choi. PIQA: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, Apr. 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. Ponde de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *CoRR*, arXiv:1803.05457, 2018.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

J. Dodge, A. Marasovic, G. Ilharco, D. Groeneveld, M. Mitchell, M. Gardner, and W. Agnew. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Conference on Empirical Methods in Natural Language Processing*, 2021. URL https://api.semanticscholar.org/CorpusID:237568724.

L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy. The pile: An 800gb dataset of diverse text for language modeling. *ArXiv*, abs/2101.00027, 2020. URL https://api.semanticscholar.org/CorpusID:230435736.

G. Gray, A. Samar, and J. Hestness. Efficient and approximate per-example gradient norms for gradient noise scale. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@NeurIPS 2023)*, 2023. URL https://openreview.net/forum?id=xINTMAvPQA.

G. Gray, A. Tiwari, S. Bergsma, and J. Hestness. Normalization layer per-example gradients are sufficient to predict gradient noise scale in transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=S7THlpvH8i.

D. Groeneveld, I. Beltagy, E. Walsh, A. Bhagia, R. Kinney, O. Tafjord, A. Jha, H. Ivison, I. Magnusson, Y. Wang, S. Arora, D. Atkinson, R. Authur, K. Chandu, A. Cohan, J. Dumas, Y. Elazar, Y. Gu, J. Hessel, T. Khot, W. Merrill, J. Morrison, N. Muennighoff, A. Naik, C. Nam, M. Peters, V. Pyatkin, A. Ravichander, D. Schwenk, S. Shah, W. Smith, E. Strubell, N. Subramani, M. Wortsman, P. Dasigi, N. Lambert, K. Richardson, L. Zettlemoyer, J. Dodge, K. Lo, L. Soldaini, N. Smith, and H. Hajishirzi. OLMo: Accelerating the science of language models. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.841. URL https://aclanthology.org/2024.acl-long.841/.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

S. Li, P. Zhao, H. Zhang, S. Sun, H. Wu, D. Jiao, W. Wang, C. Liu, Z. Fang, J. Xue, Y. Tao, B. CUI, and D. Wang. Surge phenomenon in optimal learning rate and batch size scaling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=hD9TUV4xdz.

Z. Li, S. Malladi, and S. Arora. On the validity of modeling sgd with stochastic differential equations (sdes), 2021. URL https://arxiv.org/abs/2102.12470.

I. Magnusson, N. Tai, B. Bogin, D. Heineman, J. D. Hwang, L. Soldaini, A. Bhagia, J. Liu, D. Groeneveld, O. Tafjord, N. A. Smith, P. W. Koh, and J. Dodge. Datadecide: How to predict best pretraining data with small experiments. 2025. URL https://api.semanticscholar.org/CorpusID:277787795.

S. Malladi, K. Lyu, A. Panigrahi, and S. Arora. On the SDEs and scaling rules for adaptive gradient algorithms. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=F2mhzjHkQP.

S. McCandlish, J. Kaplan, D. Amodei, and O. D. Team. An empirical model of large-batch training, 2018. URL https://arxiv.org/abs/1812.06162.

 https://doi.org/10.52202/085713-3898

T. OLMo, P. Walsh, L. Soldaini, D. Groeneveld, K. Lo, S. Arora, A. Bhagia, Y. Gu, S. Huang, M. Jordan, N. Lambert, D. Schwenk, O. Tafjord, T. Anderson, D. Atkinson, F. Brahman, C. Clark, P. Dasigi, N. Dziri, M. Guerquin, H. Ivison, P. W. Koh, J. Liu, S. Malik, W. Merrill, L. J. V. Miranda, J. Morrison, T. Murray, C. Nam, V. Pyatkin, A. Rangapur, M. Schmitz, S. Skjonsberg, D. Wadden, C. Wilhelm, M. Wilson, L. Zettlemoyer, A. Farhadi, N. A. Smith, and H. Hajishirzi. 2 olmo 2 furious, 2025. URL https://arxiv.org/abs/2501.00656.

K. Sakaguchi, R. Le Bras, C. Bhagavatula, and Y. Choi. WinoGrande: An adversarial winograd schema challenge at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 8732–8740, Apr. 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

M. Sap, H. Rashkin, D. Chen, R. Le Bras, and Y. Choi. Social IQa: Commonsense reasoning about social interactions. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454.

S. L. Smith, P.-J. Kindermans, and Q. V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1Yy1BxCZ.

L. Soldaini, R. Kinney, A. Bhagia, D. Schwenk, D. Atkinson, R. Authur, B. Bogin, K. Chandu, J. Dumas, Y. Elazar, V. Hofmann, A. H. Jha, S. Kumar, L. Lucy, X. Lyu, N. Lambert, I. Magnusson, J. Morrison, N. Muennighoff, A. Naik, C. Nam, M. E. Peters, A. Ravichander, K. Richardson, Z. Shen, E. Strubell, N. Subramani, O. Tafjord, P. Walsh, L. Zettlemoyer, N. A. Smith, H. Hajishirzi, I. Beltagy, D. Groeneveld, J. Dodge, and K. Lo. Dolma: an open corpus of three trillion tokens for language model pretraining research, 2024. URL https://arxiv.org/abs/2402.00159.

A. Talmor, J. Herzig, N. Lourie, and J. Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421.

S. Wiegreffe, O. Tafjord, Y. Belinkov, H. Hajishirzi, and A. Sabharwal. Answer, assemble, ace: Understanding how lms answer multiple choice questions. In *International Conference on Learning Representations*, 2024. URL https://api.semanticscholar.org/CorpusID:276903925.

R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. HellaSwag: Can a machine really finish your sentence? In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

G. Zhang, L. Li, Z. Nado, J. Martens, S. Sachdeva, G. Dahl, C. Shallue, and R. B. Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/e0eacd983971634327ae1819ea8b6214-Paper.pdf.

H. Zhang, D. Morwani, N. Vyas, J. Wu, D. Zou, U. Ghai, D. Foster, and S. Kakade. How does critical batch size scale in pre-training?, 2024. URL https://arxiv.org/abs/2410.21676.

A CBS Measurement Details

Our empirical method for measuring the CBS Section 3 is, in principle, sensitive to the choice of checkpoints and batch size multipliers. We therefore document the checkpoints and multipliers we used here.

OLMo 1B. When measuring the OLMo 1B CBS with branching, we set the base batch size to 1024 and the base learning rate to $0.0004 \cdot \sqrt{2}$, reflecting the default batch size of 512 and learning rate of 0.0004 in the OLMo codebase under a square-root scaling rule (Malladi et al., 2022). We then chose the following checkpoints and multipliers k:

1. Step 0: k ranging over 0.0625, 0.125, 0.25.
2. Steps 10K, 20K, . . ., 50K: k ranging over 0.5, 1, . . ., 5.
3. Steps 100K, 150K, . . . , 450K: k ranging over 1, 2, . . ., 8.

Figure 5 shows loss vs. batch size plots for all checkpoints of OLMo 1B.

OLMo 7B. We set the base batch size to 1024 and the base learning rate to 0.0003, as specified in the OLMo codebase. We then chose the following checkpoints and multipliers k:

1. Step 0: k ranging over 0.0625, 0.125, 0.25.
2. Steps 1K, 2K, 3K: k ranging over 0.25, 0.5, 1, 2, 3, 4.
3. Steps 10K, 20K, 30K: k ranging over 1, 2, 3, 4, 5.
4. Steps 72K, 150K, 200K, 239K, 300K, 350K, 400K, 477K: k ranging over 1, 2, 3, 4, 5, 6, 7, 8.

Appendix A shows loss vs. batch size plots for all checkpoints of OLMo 7B. These checkpoints were chosen manually as we developed this project. Over the course of our experimentation, we launched many additional runs beyond the ones discussed above. Since the choice of k can influence the conclusions of our method, we filtered down the included runs to make the choice of k systematic.

1B runs were launched on a single node of H100 GPUs, and 7B runs were launched on 8 nodes.

B Noise Scale Measurement Details

We use the gradient noise scale estimator proposed by McCandlish et al. (2018, Appendix A) to estimate the gradient noise scale. The method estimates the gradient noise scale using gradient norms at two different batch sizes B_{big} and B_{small} according to:

$$\mathcal{B}_{\text{simple}} \approx \frac{\mathcal{S}}{\|\mathcal{G}\|^2}, \text{where}$$
$$\mathcal{S} = \frac{\|G_{\text{small}}\|^2 - \|G_{\text{big}}\|^2}{1/B_{\text{small}} - 1/B_{\text{big}}}$$
$$\|\mathcal{G}\|^2 = \frac{B_{\text{big}}\|G_{\text{big}}\|^2 - B_{\text{small}}\|G_{\text{small}}\|^2}{B_{\text{big}} - B_{\text{small}}}.$$

We use large batch size $B_{\text{big}} = 64$ and small batch size $B_{\text{small}} = 1$.

It holds that $\mathbb{E}[\mathcal{S}] = \text{tr}(\Sigma)$ and $\mathbb{E}\left[\|\mathcal{G}\|^2\right] = \|G\|^2$. We thus average $\mathcal{S}$ and $\|\mathcal{G}\|^2$ over 4096 batches reduce variance and then return their ratio as our estimate of the noise scale $\mathcal{B}_{\text{simple}}$, using offline (i.e., unseen) data in each batch.

We estimate a confidence interval for $\mathcal{B}_{\text{simple}}$ in two steps. First, we estimate 95% confidence intervals for $\mathcal{S}$ and $\|\mathcal{G}\|^2$, assuming the data are exponentially[3] and normally distributed, respectfully, based

[3]For the exponential distribution, we use approximate confidence interval under "Confidence Intervals" here: https://en.wikipedia.org/wiki/Exponential_distribution.

https://doi.org/10.52202/085713-3898

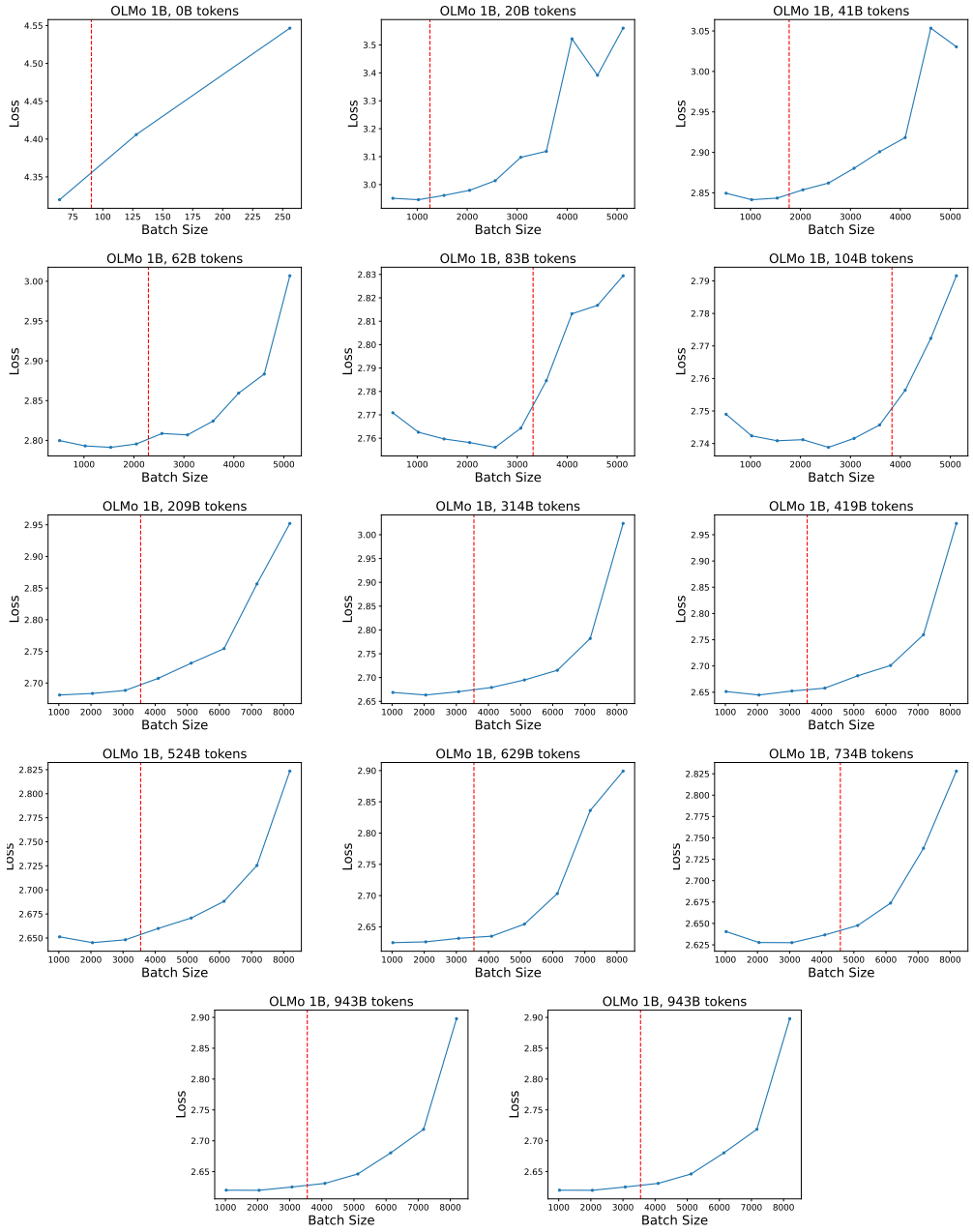
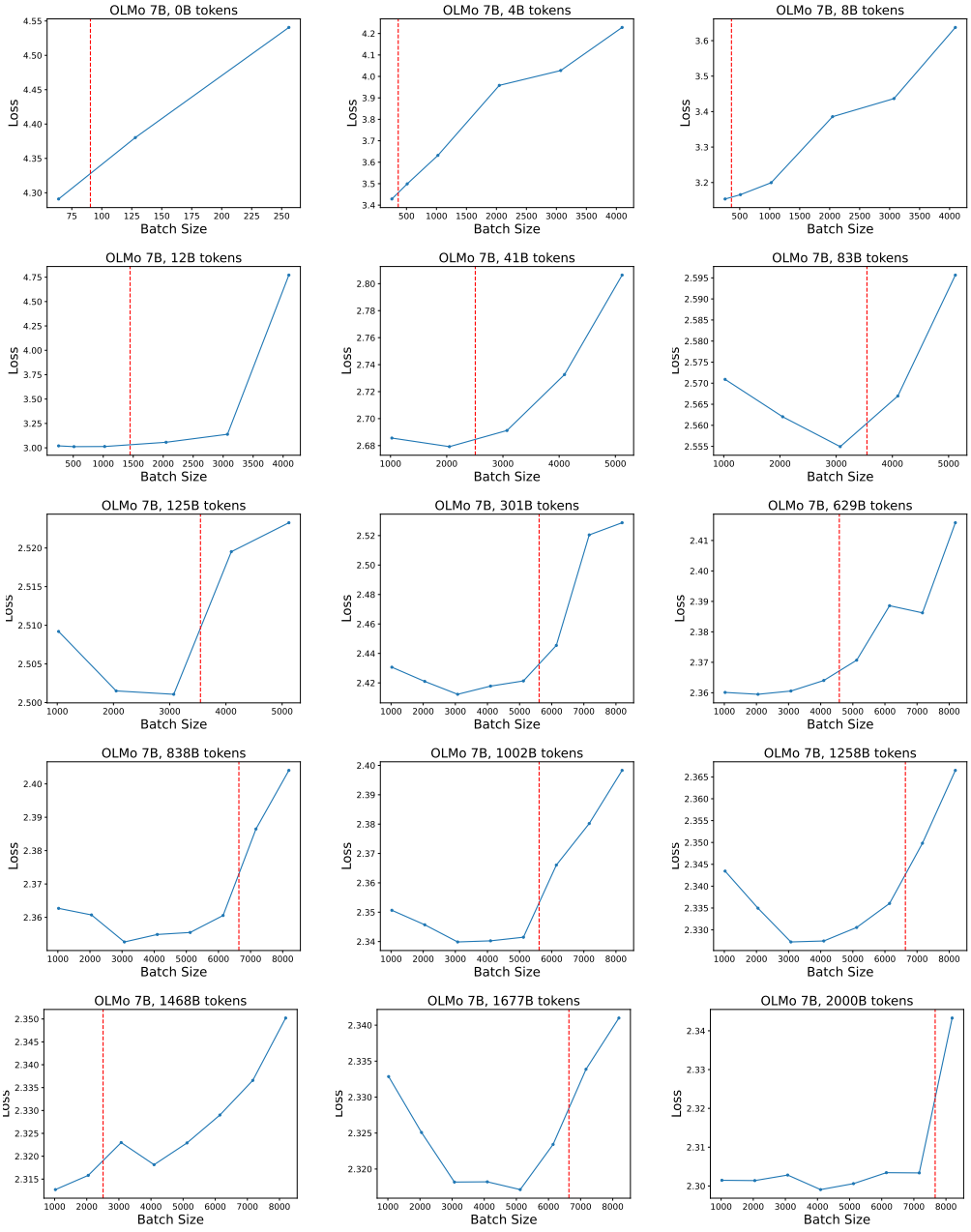


Figure 5: All loss vs. batch size plots for OLMo 1B. Overall, the red line moves to the right over time, showing that the CBS increases.



https://doi.org/10.52202/085713-3898

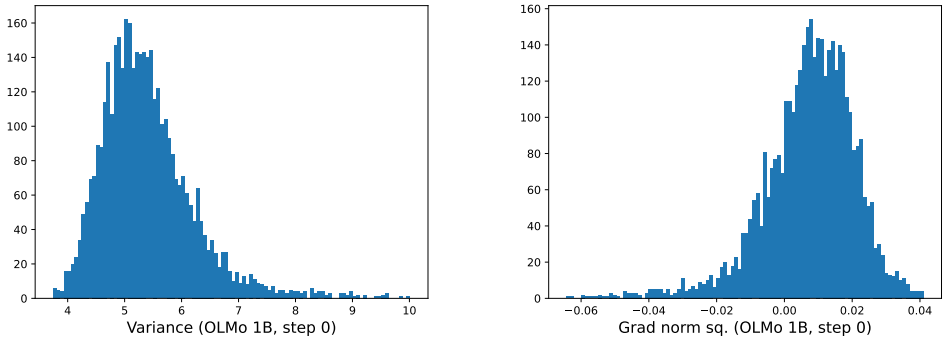


Figure 6: Representative histograms for $\mathcal{S}$ and $\|\mathcal{G}\|^2$, showing data from the 1st to 99th percentiles. The distribution for $\mathcal{S}$ is positive, leading us to use an exponential distribution, while the fact that some samples of $\|\mathcal{G}\|^2$ are negative motivates a normal distribution.

on manual inspection of their distributions (cf. Figure 6). We denote these intervals $[a_{\mathcal{S}}, b_{\mathcal{S}}]$ and $[a_{\|\mathcal{G}\|^2}, b_{\|\mathcal{G}\|^2}]$, respectively. We then define the confidence interval for $\mathcal{B}_{\text{simple}}$ as follows:

$$\left[\frac{a_{\mathcal{S}}}{b_{\|\mathcal{G}\|^2}}, \frac{b_{\mathcal{S}}}{a_{\|\mathcal{G}\|^2}}\right].$$

If our estimates for $\mathcal{S}$ or $\|\mathcal{G}\|^2$ (or their lower or upper bounds) come out negative, we consider them to be 0.

The checkpoints considered for OLMo 1B are steps 0, 10K, 20K, 40K, . . ., 100K, 200K, . . . 400K. For OLMo 7B, we use checkpoints at steps 0, 10K, . . ., 40K, 60K, 70K, . . ., 100K, 200K, . . . 400K. The noise scale experiment for each checkpoint (for both the 1B and 7B models) was launched on a single GPU.

C License Information

The OLMo models (Groeneveld et al., 2024; OLMo et al., 2025) and pretraining code, which we use, are released under Apache-2.0 license. C4 (Dodge et al., 2021) is released under ODC-BY license. The Pile (Gao et al., 2020) is released under MIT license.

D Deriving CBS Scaling Laws: An Attempt

In this section, we explore whether our empirical fits for the critical batch size over training can be used to derive scaling laws for aggregate critical batch size that have been derived in prior work. These scaling laws assume we want to use a fixed batch size B over training, and then train many different models to the same target loss. They then measure the critical B^* up to which increases in batch size do not diminish token efficiency. The standard finding from such work is that CBS grows $\propto \sqrt{T}$, where T is the total training budget in tokens. This is consistent with our finding that CBS increases over the course of training—moreover, we now seek to analyze whether this scaling law can be derived from our empirical measurements of local CBS. If so, this would provide converging evidence and a simpler method for fitting CBS scaling laws that only requires training a single model.

To begin, we assume that the goal of picking a fixed batch size B is to minimize the L2 distance to the local CBS over the course of training. It is not obvious that minimizing L2 distance is the right way to pick the fixed CBS: for instance, we might want to weight training at a batch size *above* the local CBS more negatively than training below it. Regardless, we will proceed for now under the assumption that this is the right perspective. We also make the weaker assumption that $f(t) = 0$, in line with our empirical findings (Section 3). It follows that the best batch size to train at (i.e., fixed CBS) is simply the average local CBS over training:

Proposition 1. *Let $f(t)$ be integrable with $f(0) = 0$ and define*

$$R_2 = \sqrt{\int_0^T (B - f(t))^2 \, \mathrm{d}t}.$$

Then R_2 is minimized by $B^* = \frac{1}{T}\int_0^T f(t)\mathrm{d}t$.

Proof. We can first simplify the expression for $(R_2)^2$:

$$\begin{aligned}(R_2)^2 &= \int_0^T (B - f(t))^2 \, \mathrm{d}t \\ &= \int_0^T \left(B^2 - 2Bf(t) + f(t)^2\right) \mathrm{d}t \\ &= B^2T - \int_0^T \left(2Bf(t) - f(t)^2\right) \mathrm{d}t.\end{aligned}$$

Now, taking the derivative with respect to B, we get

$$\frac{\mathrm{d}}{\mathrm{d}B}(R_2)^2 = 2BT - 2\int_0^T f(t)\mathrm{d}t.$$

Note that the second derivative $2T$ is positive. Thus, setting the derivative to 0 and solving for B, we conclude that the following value of B minimizes R_2:

$$B = \frac{1}{T}\int_0^T f(t)\mathrm{d}t. \qquad \square$$

Thus, under the assumptions we have made, if we are trying to pick a fixed batch size that best approximates the local CBS throughout training, we can simply pick the average CBS over training. We can use Proposition 1 to derive a scaling law for the fixed B^* as a function of the final CBS or, equivalently, the total steps T. We now consider various reasonable functional forms $f(t)$ for the CBS.

D.1 Power Law CBS Scaling

We first consider the prediction for the fixed CBS scaling law if the local CBS evolves as a power law.

Proposition 2 (B^* for power-law CBS)**.** *Let $f(t) = t^c$ for $c > 0$. Then the fixed CBS is*

$$B^* = \frac{1}{c+1}T^c.$$

Proof. Plug in and solve the integral:

$$\begin{aligned}B &= \frac{1}{T}\int_0^T t^c \mathrm{d}t \\ &= \frac{1}{T} \cdot \left[\frac{t^{c+1}}{c+1}\right]_0^T \\ &= \frac{T^c}{c+1}.\end{aligned} \qquad \square$$

In the case where $c = 1/2$ (square root), $B^* = \frac{2}{3}B^*_T = \frac{2}{3}\sqrt{T}$, which derives the $\sqrt{T}$ scaling law proposed by prior work (Zhang et al., 2024).

https://doi.org/10.52202/085713-3898

task	split	# shots	reference
ARC-Challenge	Test	5	(Clark et al., 2018)
ARC-Easy	Test	5	(Clark et al., 2018)
CommonsenseQA	Val	5	(Talmor et al., 2019)
HellaSwag	Val	5	(Zellers et al., 2019)
MMLU	Val and Test	5	(Hendrycks et al., 2021)
PIQA	Val	5	(Bisk et al., 2020)
Social IQa	Val	5	(Sap et al., 2019)
WinoGrande	Val	5	(Sakaguchi et al., 2020)
GSM8K	Gold	5	(Cobbe et al., 2021)
Minerva	Gold	0	(Lewkowycz et al., 2022)
Humaneval	Gold	0	(Chen et al., 2021)
MBPP	Gold	0	(Austin et al., 2021)
Copycolors 10-way		0	(Wiegreffe et al., 2024)

D.2 Logarithmic CBS Scaling

Proposition 3 (B^* for log CBS)**.** *Let $f(t) = \log(t+1)$. Then the fixed CBS is*

$$B^* = \frac{T}{T+1}\log(T+1) - 1.$$

Proof. Plug in and solve the integral:

$$\begin{aligned} B &= \frac{1}{T}\int_0^T \log(t+1)\mathrm{d}t \\ &= \frac{1}{T}\cdot\Big[((t+1)\log(t+1) - t)\Big]_0^T \\ &= \frac{T}{T+1}\log(T+1) - 1. \end{aligned}$$

□

Thus, for large T, the fixed CBS will scale as $B^* \approx \log T$.

D.3 Discussion

These results show that, if we are choosing the fixed batch size to minimize average distance to the CBS as it evolves over training, we should pick it, more or less, as a simple function that slightly discounts the final CBS. Specifically, if we believe that the local CBS grows as $\sqrt{T}$ during training, then this derives the $\sqrt{T}$ scaling law for B^* proposed in prior work.

One limitation of this view is that the L2 residuals may not be the right way to measure closeness to the CBS. In particular, it may be worse to overestimate the CBS compared to underestimate, as training above the CBS (with a scaled up learning rate) can be unstable. We thus do not read to much into this analysis, but view it as a potentially useful starting point for future empirical and theoretical that derives CBS scaling laws from the development of the local CBS over training.

E BPB Evaluation on Downstream Tasks

This section lists the datasets we used to compute BPB measures for downstream tasks. For multiple-choice tasks, we use the Cloze/Completion Formulation (CF), and compute the BPB metric on the gold answer. For completion tasks, we simply compute BPB over the correct answer. This approach was inspired by Bhagia et al. (2024). The selection of tasks follows the guidelines from Magnusson et al. (2025).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: CBS measurements in Section 3 and batch size warmup tested in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Assumptions and limitations of our CBS method are discussed in Section 3.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

https://doi.org/10.52202/085713-3898

Justification: Our (minor) theoretical results are fully justified in Appendix D.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Hyperparameter information and other experimental choices are documented in Appendices A and B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: If accepted, we will release code with the camera-ready version.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Provided in Appendices A and B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Figure 2 shows an interval representing lower and upper bounds on the CBS, as documented in Section 3. Figure 3 reports a confidence interval for the noise scale whose details are documented in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

https://doi.org/10.52202/085713-3898

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Provided in Appendices A and B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: No explanation of special circumstances necessary.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This work is foundational research on the science of pretraining language models with no immediate impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No pretrained models or datasets released.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The licenses for OLMo, C4, and the Pile are acknowledged in Appendix C.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

https://doi.org/10.52202/085713-3898

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets released.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No human subjects involved.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No human subjects involved.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.