Standard fine-tuning inefficiently uses rare data

Anonymous Author(s)
Affiliation
Address
email

Abstract

What is the best way to improve performance on a task with a small number of data points? For such applications, it is common to initialize training with a language model pre-trained on general web data instead of training from scratch. We find that on tasks such as math, coding, and instruction following, simply replaying pre-training data while fine-tuning can outperform standard fine-tuning, effectively training on up to $1.87\times$ more samples. Though replay is often used to reduce catastrophic forgetting on previous domains, we surprisingly find that it is helpful even if you only care about loss on the new rare data. Next, we study how much additional rare data efficiency we can achieve by changing the data mixture across all of training. By searching over possible two-stage mixture schedules, we can improve data efficiency by up to $4.8\times$ over the new baseline. Interestingly, we find cases where the optimal data schedule only requires replay, removing the need to allocate rare data to pre-training. We use the findings from our controlled experiments to improve performance for specific rare fine-tuning setups, increasing agentic web navigation accuracy by 4.5% and Basque question-answering accuracy by 2%.

1 Introduction

The standard language modeling paradigm is pre-training on a large amount of web data and fine-tuning on a task of interest (e.g. math) [Radford et al., 2018, Devlin et al., 2019]. This strategy enables better performance on this task even when there are only a few relevant data points to train on. Typically, practitioners fine-tuning language models vary hyper-parameters such as learning rate and repetition count with the goal of lowering loss on the rare data.

In this paper, we investigate how to best leverage rare data points in relation to pre-training. To rigorously answer this question, we set up a controlled experimental setting with two data distributions: common data, representing generic web text, and rare data, representing few tokens for a task of interest (i.e. math, associated with high-quality tokens from FineMath). Since we only pre-train to improve rare loss, we evaluate performance purely by measuring validation loss on the rare task.

First, we determine whether we can improve upon standard fine-tuning. We carefully tune a fine-tuning baseline under standard practice of pre-training on web data followed by fine-tuning on rare data (with two seperate learning rate schedule cycles). We show that we can improve performance on the rare task by simply replaying pre-training data while fine-tuning, even if we only care about the rare task, improving data efficiency by up to $1.87\times$. This is surprising considering that the replay data is generally unrelated to the rare task. Though we do not fully understand the cause, we offer candidate hypotheses involving unstable optimization and overfitting to finite samples.

Next, we determine how much additional data efficiency we can achieve by modifying the full training stack. Instead of using seperate learning schedules for pre-training and fine-tuning, we use a single learning rate schedule, closer to common practice in mid-training. We establish a new

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

baseline without replay that achieves up to $9.42\times$ better data efficiency than the previous fine-tuning baseline. We suspect that traditional fine-tuning with a split learning rate schedule underperforms this baseline because it rewarms up the learning rate and resets the optimizer state, and we call for model developers to release checkpoints prior to annealing for fine-tuning.

Under this new strategy, we determine the optimal dynamic mixture across two training stages, mimicking pre-training and fine-tuning. In addition to using replay data (which sets the mixture for the second stage), we can decide what fraction of the rare data we allocate to the first stage instead of the second stage. Under this framework, we thoroughly search for the best data schedule for rare data loss, which is up to $4.8\times$ data efficient than the baseline. We find that increasing the replay fraction is generally most important when we allocate no rare data to the first stage. For coding data in particular, replay is so strong that it is best to not allocate any rare data to the first stage.

Overall, our experiments establish clear principles for leveraging rare data. For fine-tuning, replay is a simple and effective strategy for many data-limited tasks. We apply this to rare tasks such as web navigation with agents (improving score by 4.5%) and low-resource language learning for Basque (improving question-answering accuracy by 2%). For pre-training, we find that a majority of the benefit of introducing data early can be achieved by improving fine-tuning with replay. Knowing when this is the case is important for many downstream applications in using language models with private data, new facts, and domain-specific corpuses.

2 Improving rare data efficiency of fine-tuning

2.1 Setup

Training. We have two pools of training data: **common data**, representing generally useful web data, and **rare data**, representing data relevant for a task of interest (i.e. math). We will train a language model on 4 billion tokens of data from these pools. Since web data is abundant relative to rare data, we do not constrain the web data and never repeat these samples. Instead, we model a constraint on the amount of rare data, which we limit to 4 million tokens. Under this setup, we are allowed to choose the learning rate schedule and how many times we repeat the rare data. We use C4 for our common domain and FineMath (math), StarCoder (coding), and Flan (instruction following) for our rare domains. We train a 150 million parameter Llama-style language model ([Grattafiori et al., 2024]) using AdamW. We give full training details in Appendix D.

Evaluation. We are only interested in performance on the real domain. We evaluate this by measuring loss on a held-out validation set from the rare data distribution. We choose validation loss since it scales much more smoothly than accuracy metrics, especially for models at our scale.

Rare data efficiency. Given two fine-tuning strategies S_1 and S_2, we would like to compare how "efficiently" they are using the data. To measure this, we fit a scaling law to a fixed reference training strategy S_{ref} as it gets access to more rare data. For any training strategy S we want to evaluate, we can measure the loss of the final model and recover how much rare data the reference strategy would need to match this loss, denoted $D(\mathrm{S})$. However, this quantity depends on the effectiveness of the reference strategy. To remove this dependence, we always report data efficiency as a relative improvement from one strategy to another, or $\frac{D(\mathrm{S}_2)}{D(\mathrm{S}_1)}$. A data efficiency improvement of $k\times$ now corresponds to "training with this strategy is like effectively training on k times more rare data". We give more details on how we fit the scaling laws in Appendix G.

2.2 Fine-tuning baseline

Suppose we train for a total of T steps, with γ fraction of them being rare. Standard fine-tuning corresponds to training on common data with a cosine learning rate schedule for $(1-\gamma)T$ steps. Then, we train on the rare data for γT steps with a seperate cosine learning rate schedule. In between the stages, we reset the optimizer state (containing moving estimates of the gradients for AdamW) matching practice for fine-tuning on open-weight models. We provide a competitive fine-tuning baseline by tuning the two main choices: the learning rate and the number of rare data repetitions (exact procedure in Appendix F.1). In summary, it is best to use a learning rate of 1e-4 for pre-training and 3e-4 for fine-tuning on 64 repetitions of the rare data. If we try repeating the data more than

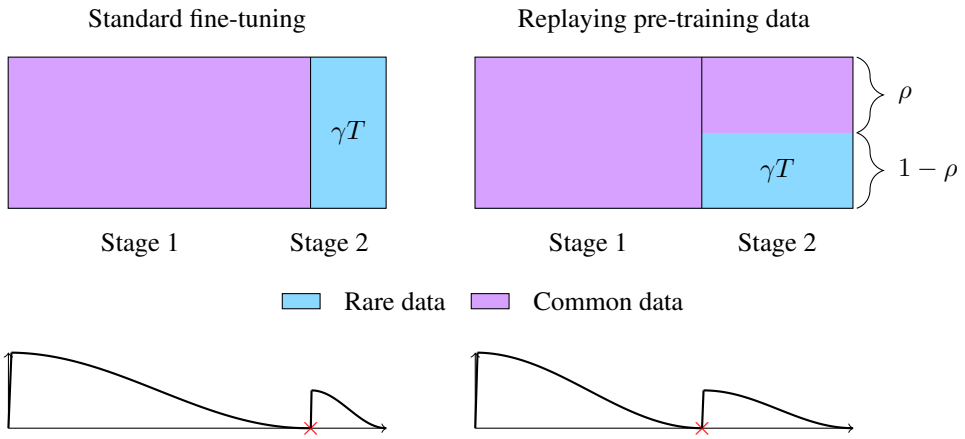


Figure 1: **Controlled fine-tuning visualization.** We systematically explore the benefit of replaying data from the pre-training domain while fine-tuning on the rare data. On the right, we show standard fine-tuning for T steps where γ fraction of the steps are on rare data. On the left, we show fine-tuning with replay fraction ρ (where we shorten pre-training to keep the total number of steps fixed). We use (independently tuned) cosine learning rate schedules for each stage, with an optimizer state reset between the stages to simulate standard practice for fine-tuning open-weight models.

this, the validation loss will start to increase akin to classical overfitting (this is not captured by prior data-constrained scaling laws, discussed in K.1). This setup serves as $1\times$ rare data efficiency.

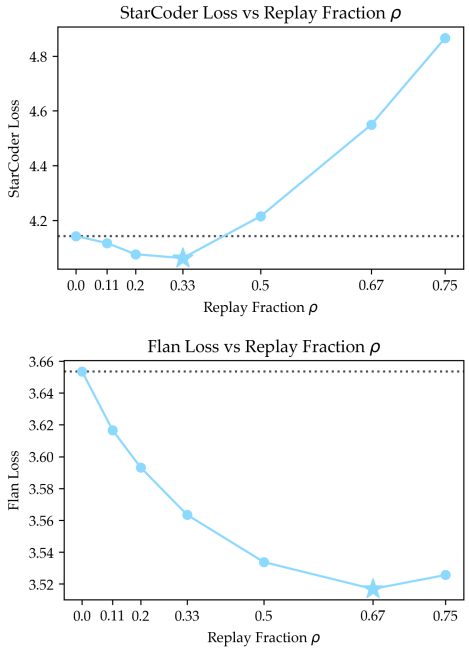
2.3 Replay improves data efficiency

We introduce a simple strategy that improves loss on the rare task: mix web data while fine-tuning on the rare data. Specifically, we introduce a replay fraction ρ which dictates what fraction of fine-tuning steps will be on common data. When we increase this replay fraction, we decrease the number of pre-training steps to conserve the total number of training steps (Figure 1, right). In Figure 2, we show how the final model's loss depends on the replay ratio. We find that for each domain, there is an optimal replay fraction that maximizes the data efficiency gains (indicated by the starred points). Replay improves data efficiency by $1.87\times$ for Flan, $1.49\times$ for FineMath, and $1.09\times$ for StarCoder. Note that even if we only care about loss on the fine-tuning task, it is helpful to add the less relevant web data while training. We observe that code, which is the most rare with respect to C4 since C4 was filtered to remove code, can tolerate less replay data than the relatively more common domains of math and instruction following, though the loss improvements are similar across domains.

Though replay is a common method in continual learning, it is almost always to prevent catastrophic forgetting of old tasks [Rolnick et al., 2019, Parisi et al., 2019]. Interestingly, we find that replay improves performance on the new in-distribution training task, which departs from the standard intuition. We provide a more detailed discussion in Section 5.

3 Improving rare data efficiency of pre-training

In the previous section, we limited ourself to using replay during fine-tuning. In this section, we aim to understand how much additional data efficiency we get by setting the data mixture for two stages of training, akin to changing both pre-training and fine-tuning. We first unify the optimization process of pre-training and fine-tuning into a single learning rate schedule cycle with no optimizer reset. We now search over possible ways to schedule the rare data by choosing a replay ratio for stage 2 as well as how much rare data is used in stage 2 (equivalently, setting the mixture for two stages of training). This extra flexibility allows us to quantify how much benefit we get from introducing rare data earlier in training (stage 1) instead of reserving it all for the end (stage 2).



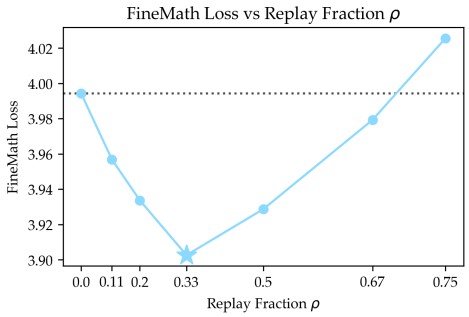


Figure 2: **Replay improves loss on rare data.** We show that across our rare domains, the correct amount of replay (starred points) beats the no replay baseline (dotted line). Though data distributions closer to pre-training (Flan) can tolerate more replay compared to further domains (StarCoder), the loss improvement is relatively constant across domains.

3.1 WSD baseline

To start we tune the baseline of seeing all common data before all the rare data. Similar to before, we tune the epoch count for this data schedule and find a slightly lower optimal epoch count of 32 (Appendix E.1.1). However, we find that the correct learning rate schedule is critical for rare data efficiency. Default practice for learning rate schedules (i.e. cosine, linear) is to slowly anneal to zero over the course of training. Recent work in mid-training has suggested using a warmup-stable-decay (WSD) learning rate schedule [Hu et al., 2024]. This consists of a short linear warmup, a stable training phase, and a sharp linear decay for a variable fraction of training referred to as the cooldown period. Interestingly, during the learning rate decay, the loss decreases at a much faster rate than the rest of training. This can be exploited to get stronger performance on target data by placing it at the end of training. We explain and visualize these benefits in more detail in Appendix H.

In Figure 3, we show that WSD offers a significant benefit over the more traditional schedules that decay the learning rate over all of training for the fine-tuning baseline. For our setting, we find it best to fix a cooldown period of 10% of training for all domains, increasing data efficiency by $28.47\times$ relative to the worst cooldown duration. In contrast to the fine-tuning experiments, we find it better to use a slightly higher learning rate of 3e-3 for all of training (details in Appendix E.1.2).

Due to the learning rate schedule, we refer to this strategy as the WSD baseline. This increases data efficiency relative to the fine-tuning baseline by $9.92\times$ for Starcoder, $6.37\times$ for FineMath, and $2.77\times$ for Flan. This is likely because the joint training doesn't reset optimizer state and rewarmup the learning rate. As such, we believe fine-tuning would benefit from initializing at a pre-annealed pre-training checkpoint instead of the final checkpoint. We call on model developers to release the model and optimizer state before cooldown since this is more useful for downstream applications.

3.2 Data schedule space

Now that we have fairly tuned our WSD baseline, we are interested in whether this is the best data schedule across pre-training and fine-tuning. To reduce the complexity of searching over all permutations, we consider data schedules of two stages. We get to pick the fraction of rare data for each stage, subject to the constraint on the total number of rare steps. We fix the sample-level ordering of data points within each domain to reduce variance when comparing different schedules.

This space now only has two degrees of freedom. There are multiple ways of parameterizing it, and we decide to use **replay fraction** ρ (how much common data is replayed) and the **rare stage**

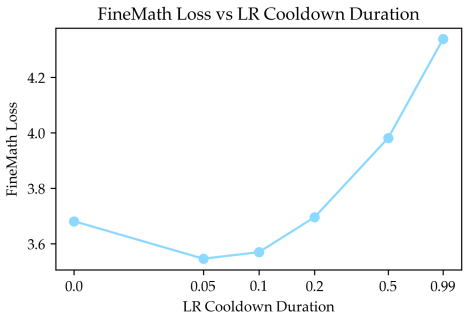


Figure 3: **Tuning learning rate cooldown.** We tune how long we should cool down the learning rate for WSD. The above plot shows the optimal cooldown period is between 0.05 and 0.1; we use 0.1 for consistency across domains and being fair to changing data schedules.

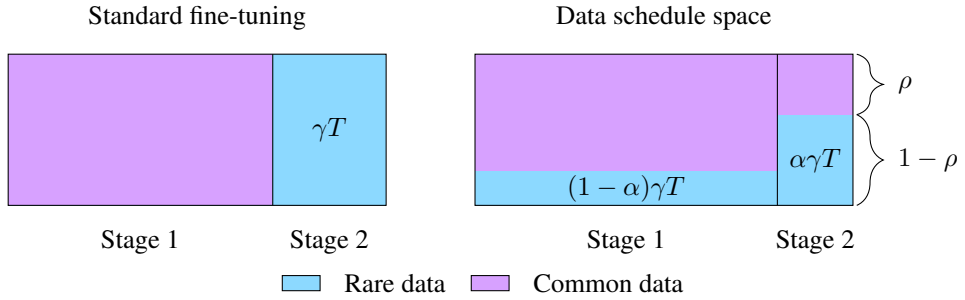


Figure 4: **Controlled mid-training visualization.** We systematically explore the space of data schedules when training on T tokens where γT are rare. A data schedule allocates an α fraction to stage 2 where stage 2 has a replay fraction ρ. Standard fine-tuning puts all rare data at the end with no replay ($\alpha = 1$, $\rho = 0$).

2 allocation α (what fraction of the total rare data is allocated to stage 2). We provide a more intuitive visualization in Figure 4. Fine-tuning now has a simple interpretation: no replay data ($\rho = 0$) and all the rare data is used in stage 2 ($\alpha = 1$). Finding the optimal data order under this parameterization is equivalent to finding the best setting of ρ and α. We provide a detailed discussion of the parameterization and equivalences in Appendix A.

3.3 Best data schedule

To determine the best data schedule, we sweep over replay fraction ρ and rare stage 2 allocation α. We are interested in three strategies: the WSD baseline of stage 1 fully common and stage 2 fully rare ($\rho = 0$, $\alpha = 1$), fine-tuning while replaying common data in stage 2 ($\alpha = 1$), and the full space of modifications (all settings). In Table 6, we share the data efficiency improvements of each strategy.

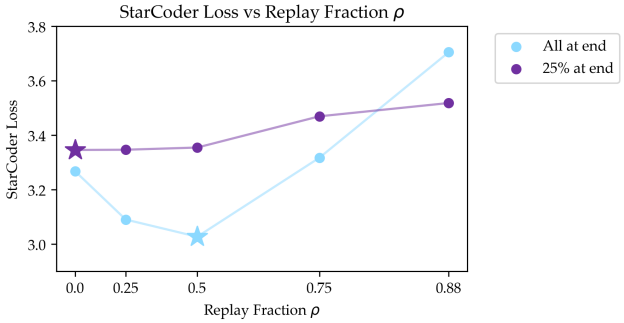


Figure 5: **Importance of replay fraction depends on rarity.** When all the rare data is seen during fine-tuning, tuning the replay fraction becomes critical to improve loss (blue line). When we change pre-training to see some of the rare data, tuning the replay fraction is not important and can sometimes hurt loss (purple line).

Domain	Replay	Full Schedule
StarCoder	1.53×	1.53×
FineMath	1.85×	2.49×
Flan	2.06×	4.80×

Figure 6: **Data efficiency across domains.** We compare the data efficiency of best replay and best schedule relative to the fine-tuning baseline.

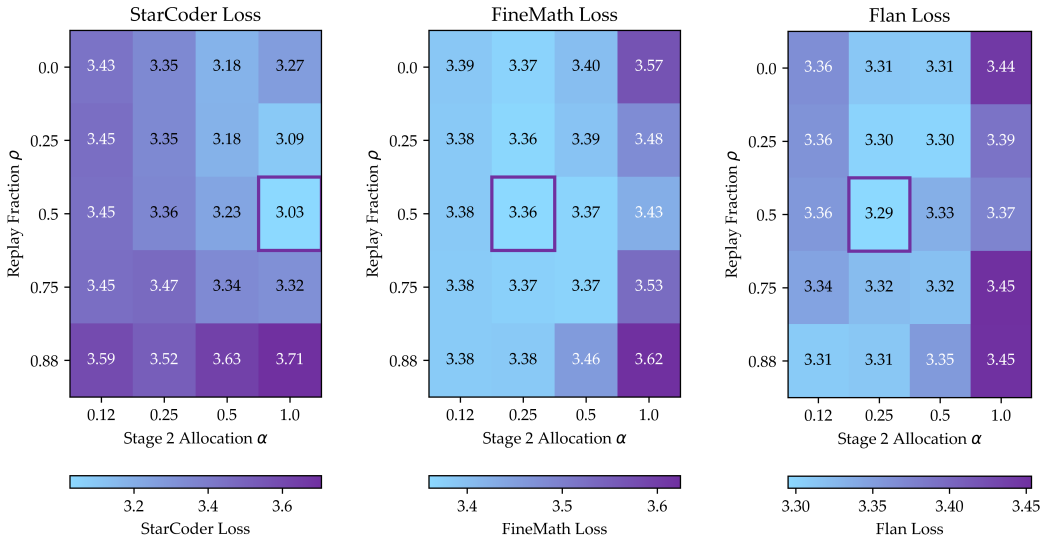


Figure 7: **Full data schedule sweep.** We sweep over data schedules, parameterized by their replay fraction and fraction of rare data allocated to stage 2. Standard fine-tuning (no replay and all rare data in stage 2, top right corner) achieves the worst loss for FineMath and Flan. This can be improved by adding replay data (right column). This can also be solved by having some rare data in stage 1, in which case replay becomes less important.

3.3.1 Replay is more helpful when data is more rare

In addition to a direct comparison of these strategies, we can determine when replay is important. We first find that when the rare data is unseen during stage 1, replay is critical for improving loss (example for Starcoder in Figure 5, blue). On the other hand when we keep 75% of the rare data for pre-training, replay is no longer helpful (Figure 5, purple). Together, this indicates that replay is more helpful as the data is less present during stage 1. One interpretation of these results is that training benefits from a smooth transition when changing data distributions, whether it comes from adding web data to stage 2 or rare data to stage 1.

We show the full results of sweeping over replay fraction and stage 2 allocation in Figure 7. In the context of this plot, pure fine-tuning is the top-right entry, fine-tuning with replay is the right column, and the full space of modifications is the entire plot.

3.4 Do we need to change pre-training?

One natural question for applications is whether one needs to change pre-training (correspondingly, stage 1) to maximally leverage task-relevant data. We can use the full sweep over data schedules to determine when it's necessary to modify pre-training. We find that for FineMath and Flan, we can not get the full benefits of the optimal data order by only changing stage 2 (achieving 67.4% of gains for FineMath and 46.0% of the gains for Flan measured logarithmically). On the other hand, for Starcoder, the optimal data schedule only requires adding replay data to stage 2. It is encouraging that we can get away with not using data early since it is prohibitive or impossible to change pre-training for many applications. We discuss these implications in more detail in Section 4.2.

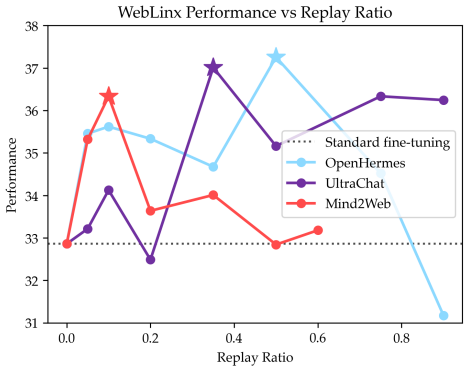


Figure 8: **Weblinx.** We fine-tune Llama 3.1-8B Instruct on Weblinx demonstrations. Without any replay data, we get 32.86% accuracy following the original hyper-parameters. We find that mixing generic instruction following data (OpenHermes, UltraChat) improves accuracy by up to 4.5%. Surprisingly, this is even better than replaying demonstrations from an alternative web agent task (Mind2Web).

4 Implications

4.1 Rare post-training benefits from replay

Our findings offer an immediate prescription for practitioners hoping to train on rare data: replay pre-existing data. Though this sounds like a simple modification, this is rarely done while fine-tuning in practice as it is commonly believed that fine-tuning is the correct way to leverage rare data. For most fine-tuning setups, the pre-trained model is downloaded off the internet with the learning rate fully annealed and optimizer state reset, close to the setup in 2. Moreover, since we usually do not have access to the common data distribution, we have to pick an approximation for our replay data. We note that using a replay fraction of ρ requires $\frac{1}{\rho}$ times as many training steps. This is generally a favorable tradeoff for fine-tuning since we are rarely compute-constrained. We demonstrate that for the data-constrained domain of web agents and the low-resource domain of Basque, replay is able to provide increased accuracy.

4.1.1 Web Agents

Recently, language models have been trained to perform agentic tasks, requiring advanced capabilities such as web navigation. However, collecting demonstrations requires expensive human labor and training data is scarce. Therefore, it is important to understand how to best leverage this data.

We follow the data and evaluation protocol from Weblinx [Lù et al., 2024]. Since this task requires instruction following, we start training with Llama 3.1-8B Instruct on a fixed number of rare demonstrations. For the replay data, we use OpenHermes [Teknium, 2023] or UltraChat [Ding et al., 2023] instruction-following data instead of web data since we are fine-tuning an instruct model. For training, we use the hyperparameters from the paper's training pipeline. We measure accuracy on the validation and in-distribution test set under the paper's established static evaluation metric. We defer to the original paper for more details on the data and evaluation.

We find that when training on web agents data, there is a consistent advantage to replaying instruction following data. In Figure 8, we show that replaying instruction following data improves accuracy by up to 4.5%. We provide additional details and experiments in Appendix I.

4.1.2 Basque

Basque is a low-resource language spoken by approximately 800k people in the Basque Country. As such, this language is scarce on the web and pre-training datasets, constituting only 0.035% of Common Crawl [Etxaniz et al., 2024]. However, thanks to a thriving NLP research community, there is a large amount of Basque data available through the Latxa corpus [Etxaniz et al., 2024].

We are interested in how to fine-tune a model to have high Basque understanding. We start continual pre-training from Llama 3.1-8B with access to 200M Basque tokens. For replay data, we use the SlimPajama replication [Soboleva et al., 2023, Weber et al., 2024] as a proxy for the unreleased Llama pre-training data. For evaluation, we measure validation loss as before, as well as accuracy on a professional Basque translation [Baucells et al., 2025] of the commonsense reasoning benchmark COPA [Gordon et al., 2012, Ponti et al., 2020] supported on `lm-eval-harness` [Gao et al., 2024].

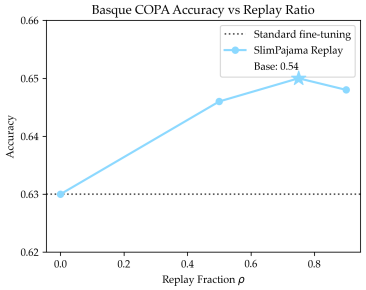


Figure 9: **Basque.** We fine-tune Llama 3.1-8B on 200M Basque tokens from the Latxa training corpus and measure accuracy on Basque COPA. We find that replaying generic pre-training data from SlimPajama improves accuracy by up to 2%.

We find that when training on Basque data, there is a consistent advantage to replaying pre-training data. In Figure 9, we show that the model achieves higher accuracy on the Basque evaluation task. We provide more details and experiments in Appendix J.

4.2 Can fine-tuning subsume pre-training?

Since its expensive to modify pre-training, practitioners only control fine-tuning in most applications. For example, companies want models to utilize internal data for their operations, model developers want to update models with new knowledge about the world, and domain experts want models specialized for their field. In all these cases, it is prohibitively expensive to retrain a model from scratch. In these scenarios, how much do we lose by not changing pre-training?

We find evidence in the StarCoder setting that the optimal data schedule only requires tuning replay ratio. This shows that for some domains, it's possible to get the benefits of scheduling data without changing pre-training. However, this only works if we introduce replay data and use a single learning rate schedule. It will be important to develop better fine-tuning methods to match altering pre-training, as well as delineate when it's possible in the first place.

5 Related work

Mid-training. Many recent language models augment pre-training with a mid-training phase where they reduce the learning rate while training on high-quality data OLMo et al. [2025], Grattafiori et al. [2024], Li et al. [2025], Nvidia et al. [2024]. There has been some initial work on characterizing the benefit of putting rare data at the end of training [Aryabumi et al., 2024, Blakeney et al., 2024] or annealing the learning rate [Hu et al., 2024]. In addition to prior knowledge, we conduct experiments on changing the pre-training in conjunction with mid-training. Moreover, we show new experiments at maximal repetition count, as well as for various ablation factors such as model size.

Robust fine-tuning. There is a rich literature on how to robustly fine-tune language models to maximize in-distribution and out-of-distribution accuracy [Phang et al., 2019, Zhang et al., 2021, Kumar et al., 2022]. Weight averaging has been one such technique to improve post-training performance [Wortsman et al., 2022, Ilharco et al., 2023, Dang et al., 2025]. Replay can be seen as qualitatively similar to weight averaging where the averaging takes places in data distribution space instead of parameter space. In contrast to prior work, we characterize the interaction between pre-training and fine-tuning, showing that the optimal way to fine-tune depends on how much exposure the pre-trained model has to the target task. Moreover, since the focus was primarily out-of-distribution performance, they under-focused on the opportunity to improve in-distribution performance.

Optimizing data mixtures. Prior work has proposed algorithms to optimize the data mixture [Chen et al., 2023, Xie et al., 2023, Jiang et al., 2024a, Fan et al., 2024]. Most such algorithms fall under an online optimization framework [Chen et al., 2025b], where the algorithm estimates which components it should upweight to maximize performance. However, such online algorithms miss the most important aspect of data mixtures: better data should be at the end. Instead, such algorithms greedily upweight the best data at the start since they do not factor in constraints on the number of available data points. Moreover, they do not account for the optimization challenges that arise when changing data distributions. Though our work uses relatively unsophisticated methods, we are able to answer more general questions and achieve stronger performance improvements.

Continual learning. There has been a lot of work on continual learning for new tasks [Rolnick et al., 2019, Parisi et al., 2019]. Such works have traditionally focused on reducing catastrophic forgetting [Kirkpatrick et al., 2017] instead of improving target task performance [Gupta et al., 2023, Ibrahim et al., 2024, Kotha et al., 2024, Çağatay Yıldız et al., 2025, Chen et al., 2025a, Springer et al., 2025]. There has also been work on methods and evaluation for teaching models new facts [Meng et al., 2023, Yang et al., 2024b, Ghosal et al., 2024, Gekhman et al., 2024, Chang et al., 2024]. Our two-stage framework helps build intuition for when pretraining is necessary, as well as shows better ways to teach models new facts. However, answering further questions about capability and knowledge will require using more refined metrics and data distributions.

Curriculum learning. Curriculum learning is generally concerned with proposing a sequence of training distributions from easy to hard [Bengio et al., 2009]. Theoretically, it has been shown that this can accelerate the speed of convergence over training by introducing tractable intermediate tasks [Abbe et al., 2023, Panigrahi et al., 2024]. Recent works have tried to design curricula using reference models [Mindermann et al., 2022, Fan and Jaggi, 2023, Lin et al., 2025] or structure over the data distribution [Chen et al., 2023]. However, there is limited evidence that changing data order improves the final iid performance of models [Wu et al., 2021]. In contrast, our work focuses on the *relevance* of the data with respect to the target task, where it is will known that changing data order improves performance (exemplified by fine-tuning). This allows there to be much larger gains in performance.

Necessity of pretraining. Many prior works argue that it is necessary to incorporate target skills during pretraining. For example, [Allen-Zhu and Li, 2024, Jiang et al., 2024b] argue that instruction-tuning data needs to be seen during pretraining. Moreover, as discussed in 4.2, many practitioners pretrain language models from scratch with the belief that it is necessary to see this data during pretraining. Our work shows that this might not be the case: for some tasks, data might not need to be seen during pretraining, as long as one follows optimal training procedures for adaptation.

6 Discussion

6.1 Hypotheses for inefficiency of fine-tuning

In this paper, we thoroughly characterize the failure of fine-tuning. We share two hypotheses for why fine-tuning underperforms replay data. We first identify a training instability that occurs for a few steps of fine-tuning which replay slightly mitigates. We provide experiments and discussion in Appendix B.1. However, even with perfect optimization, we identify a statistical barrier to leveraging rare samples due to a natural tendency to overfit to small samples. In Appendix B.2, we detail a toy model where failure arises due to a small number of noisy data points, leveraging classical intuition from the double-descent literature. Another reasonable hypothesis is that this is an artifact of our current training scale. To control for this, we ablate model size in Appendix B.3. Interestingly, the benefit of replay seems to persist across model scale, decreasing support for this hypothesis.

6.2 Limitations

Our controlled setting is different from real-world settings in possibly important ways. For example, we assume two distributions, where as in practice, pre-training is a multi-task learning problem with much higher diversity. Our space of data schedules also precludes us from studying more complicated methods such as continuous annealing of the distribution, sample-level orderings, and more advanced fine-tuning methods. There might be other important hyperparameters and joint interactions that we did not explore, possibly affecting the robustness of our results. Many properties that we care about for language models might not perfectly correlate with validation loss, which requires using different metrics under our framework. In practice, replay comes at the cost of increased compute, which might be a limiting factor outside of standard fine-tuning settings.

6.3 Broader Impacts

We hope our work can be used to improve the data efficiency of language models, especially for low resource domains that otherwise receive little attention. We acknowledge our work may increase the compute used in language model training. We believe most other harms associated with our work are generally applicable to most language modeling research.

References

E. Abbe, E. Cornacchia, and A. Lotfi. Provable advantage of curriculum learning on parity targets with mixed inputs, 2023. URL https://arxiv.org/abs/2306.16921.

L. B. Allal, A. Lozhkov, E. Bakouch, G. M. Blázquez, G. Penedo, L. Tunstall, A. Marafioti, H. Kydlíček, A. P. Lajarín, V. Srivastav, J. Lochner, C. Fahlgren, X.-S. Nguyen, C. Fourrier, B. Burtenshaw, H. Larcher, H. Zhao, C. Zakka, M. Morlon, C. Raffel, L. von Werra, and T. Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Z. Allen-Zhu and Y. Li. Physics of language models: Part 3.1, knowledge storage and extraction, 2024. URL https://arxiv.org/abs/2309.14316.

V. Aryabumi, Y. Su, R. Ma, A. Morisot, I. Zhang, A. Locatelli, M. Fadaee, A. Üstün, and S. Hooker. To code, or not to code? exploring impact of code in pre-training, 2024. URL https://arxiv.org/abs/2408.10914.

I. Baucells, J. Aula-Blasco, I. de Dios-Flores, S. Paniagua Suárez, N. Perez, A. Salles, S. Sotelo Docio, J. Falcão, J. J. Saiz, R. Sepulveda Torres, J. Barnes, P. Gamallo, A. Gonzalez-Agirre, G. Rigau, and M. Villegas. IberoBench: A benchmark for LLM evaluation in Iberian languages. In O. Rambow, L. Wanner, M. Apidianaki, H. Al-Khalifa, B. D. Eugenio, and S. Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10491–10519, Abu Dhabi, UAE, Jan. 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.699/.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116 (32):15849–15854, July 2019. ISSN 1091-6490. doi: 10.1073/pnas.1903070116. URL http://dx.doi.org/10.1073/pnas.1903070116.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

C. Blakeney, M. Paul, B. W. Larsen, S. Owen, and J. Frankle. Does your data spark joy? performance gains from domain upsampling at the end of training, 2024. URL https://arxiv.org/abs/2406.03476.

H. Chang, J. Park, S. Ye, S. Yang, Y. Seo, D.-S. Chang, and M. Seo. How do large language models acquire factual knowledge during pretraining?, 2024. URL https://arxiv.org/abs/2406.11813.

H. Chen, J. Geng, A. Bhaskar, D. Friedman, and D. Chen. Continual memorization of factoids in language models, 2025a. URL https://arxiv.org/abs/2411.07175.

M. F. Chen, N. Roberts, K. Bhatia, J. Wang, C. Zhang, F. Sala, and C. Ré. Skill-it! a data-driven skills framework for understanding and training language models, 2023. URL https://arxiv.org/abs/2307.14430.

M. F. Chen, M. Y. Hu, N. Lourie, K. Cho, and C. Ré. Aioli: A unified optimization framework for language model data mixing, 2025b. URL https://arxiv.org/abs/2411.05735.

J. M. Cohen, S. Kaur, Y. Li, J. Z. Kolter, and A. Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability, 2022. URL https://arxiv.org/abs/2103.00065.

J. M. Cohen, A. Damian, A. Talwalkar, Z. Kolter, and J. D. Lee. Understanding optimization in deep learning with central flows, 2024. URL https://arxiv.org/abs/2410.24206.

X. Dang, C. Baek, K. Wen, Z. Kolter, and A. Raghunathan. Weight ensembling improves reasoning in language models, 2025. URL https://arxiv.org/abs/2504.10478.

A. Defazio, A. Cutkosky, H. Mehta, and K. Mishchenko. Optimal linear decay learning rate schedules and further refinements, 2024. URL https://arxiv.org/abs/2310.07831.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

N. Ding, Y. Chen, B. Xu, Y. Qin, Z. Zheng, S. Hu, Z. Liu, M. Sun, and B. Zhou. Enhancing chat language models by scaling high-quality instructional conversations, 2023.

J. Etxaniz, O. Sainz, N. Perez, I. Aldabe, G. Rigau, E. Agirre, A. Ormazabal, M. Artetxe, and A. Soroa. Latxa: An open language model and evaluation suite for basque, 2024. URL https://arxiv.org/abs/2403.20266.

K. Everett, L. Xiao, M. Wortsman, A. A. Alemi, R. Novak, P. J. Liu, I. Gur, J. Sohl-Dickstein, L. P. Kaelbling, J. Lee, and J. Pennington. Scaling exponents across parameterizations and optimizers, 2024. URL https://arxiv.org/abs/2407.05872.

S. Fan and M. Jaggi. Irreducible curriculum for language model pretraining, 2023. URL https://arxiv.org/abs/2310.15389.

S. Fan, M. Pagliardini, and M. Jaggi. Doge: Domain reweighting with generalization estimation, 2024. URL https://arxiv.org/abs/2310.15393.

L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, A. Le Noac'h, H. Li, K. McDonell, N. Muennighoff, C. Ociepa, J. Phang, L. Reynolds, H. Schoelkopf, A. Skowron, L. Sutawika, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Z. Gekhman, G. Yona, R. Aharoni, M. Eyal, A. Feder, R. Reichart, and J. Herzig. Does fine-tuning llms on new knowledge encourage hallucinations?, 2024. URL https://arxiv.org/abs/2405.05904.

G. Ghosal, T. Hashimoto, and A. Raghunathan. Understanding finetuning for factual knowledge extraction, 2024. URL https://arxiv.org/abs/2406.14785.

A. Gordon, Z. Kozareva, and M. Roemmele. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In E. Agirre, J. Bos, M. Diab, S. Manandhar, Y. Marton, and D. Yuret, editors, **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398, Montréal, Canada, 7-8 June 2012. Association for Computational Linguistics. URL https://aclanthology.org/S12-1052/.

S. Goyal, P. Maini, Z. C. Lipton, A. Raghunathan, and J. Z. Kolter. Scaling laws for data filtering – data curation cannot be compute agnostic, 2024. URL https://arxiv.org/abs/2404.07177.

A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. C. Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz, D. Livshits, D. Wyatt, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Dinan, E. M. Smith, F. Radenovic, F. Guzmán, F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Thattai, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. A. Ibarra, I. Kloumann, I. Misra, I. Evtimov, J. Zhang, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. van der Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. V. Alwala, K. Prasad, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, K. El-Arini, K. Iyer, K. Malik, K. Chiu, K. Bhalla, K. Lakhotia, L. Rantala-Yeary, L. van der Maaten, L. Chen, L. Tan, L. Jenkins, L. Martin, L. Madaan, L. Malo, L. Blecher, L. Landzaat, L. de Oliveira, M. Muzzi, M. Pasupuleti, M. Singh, M. Paluri, M. Kardas, M. Tsimpoukelli, M. Oldham, M. Rita, M. Pavlova, M. Kambadur, M. Lewis, M. Si, M. K. Singh, M. Hassan, N. Goyal, N. Torabi, N. Bashlykov, N. Bogoychev, N. Chatterji, N. Zhang,

O. Duchenne, O. Çelebi, P. Alrassy, P. Zhang, P. Li, P. Vasic, P. Weng, P. Bhargava, P. Dubal, P. Krishnan, P. S. Koura, P. Xu, Q. He, Q. Dong, R. Srinivasan, R. Ganapathy, R. Calderer, R. S. Cabral, R. Stojnic, R. Raileanu, R. Maheswari, R. Girdhar, R. Patel, R. Sauvestre, R. Polidoro, R. Sumbaly, R. Taylor, R. Silva, R. Hou, R. Wang, S. Hosseini, S. Chennabasappa, S. Singh, S. Bell, S. S. Kim, S. Edunov, S. Nie, S. Narang, S. Raparthy, S. Shen, S. Wan, S. Bhosale, S. Zhang, S. Vandenhende, S. Batra, S. Whitman, S. Sootla, S. Collot, S. Gururangan, S. Borodinsky, T. Herman, T. Fowler, T. Sheasha, T. Georgiou, T. Scialom, T. Speckbacher, T. Mihaylov, T. Xiao, U. Karn, V. Goswami, V. Gupta, V. Ramanathan, V. Kerkez, V. Gonguet, V. Do, V. Vogeti, V. Albiero, V. Petrovic, W. Chu, W. Xiong, W. Fu, W. Meers, X. Martinet, X. Wang, X. Wang, X. E. Tan, X. Xia, X. Xie, X. Jia, X. Wang, Y. Goldschlag, Y. Gaur, Y. Babaei, Y. Wen, Y. Song, Y. Zhang, Y. Li, Y. Mao, Z. D. Coudert, Z. Yan, Z. Chen, Z. Papakipos, A. Singh, A. Srivastava, A. Jain, A. Kelsey, A. Shajnfeld, A. Gangidi, A. Victoria, A. Goldstand, A. Menon, A. Sharma, A. Boesenberg, A. Baevski, A. Feinstein, A. Kallet, A. Sangani, A. Teo, A. Yunus, A. Lupu, A. Alvarado, A. Caples, A. Gu, A. Ho, A. Poulton, A. Ryan, A. Ramchandani, A. Dong, A. Franco, A. Goyal, A. Saraf, A. Chowdhury, A. Gabriel, A. Bharambe, A. Eisenman, A. Yazdan, B. James, B. Maurer, B. Leonhardi, B. Huang, B. Loyd, B. D. Paola, B. Paranjape, B. Liu, B. Wu, B. Ni, B. Hancock, B. Wasti, B. Spence, B. Stojkovic, B. Gamido, B. Montalvo, C. Parker, C. Burton, C. Mejia, C. Liu, C. Wang, C. Kim, C. Zhou, C. Hu, C.-H. Chu, C. Cai, C. Tindal, C. Feichtenhofer, C. Gao, D. Civin, D. Beaty, D. Kreymer, D. Li, D. Adkins, D. Xu, D. Testuggine, D. David, D. Parikh, D. Liskovich, D. Foss, D. Wang, D. Le, D. Holland, E. Dowling, E. Jamil, E. Montgomery, E. Presani, E. Hahn, E. Wood, E.-T. Le, E. Brinkman, E. Arcaute, E. Dunbar, E. Smothers, F. Sun, F. Kreuk, F. Tian, F. Kokkinos, F. Ozgenel, F. Caggioni, F. Kanayet, F. Seide, G. M. Florez, G. Schwarz, G. Badeer, G. Swee, G. Halpern, G. Herman, G. Sizov, Guangyi, Zhang, G. Lakshminarayanan, H. Inan, H. Shojanazeri, H. Zou, H. Wang, H. Zha, H. Habeeb, H. Rudolph, H. Suk, H. Aspegren, H. Goldman, H. Zhan, I. Damlaj, I. Molybog, I. Tufanov, I. Leontiadis, I.-E. Veliche, I. Gat, J. Weissman, J. Geboski, J. Kohli, J. Lam, J. Asher, J.-B. Gaya, J. Marcus, J. Tang, J. Chan, J. Zhen, J. Reizenstein, J. Teboul, J. Zhong, J. Jin, J. Yang, J. Cummings, J. Carvill, J. Shepard, J. McPhie, J. Torres, J. Ginsburg, J. Wang, K. Wu, K. H. U, K. Saxena, K. Khandelwal, K. Zand, K. Matosich, K. Veeraraghavan, K. Michelena, K. Li, K. Jagadeesh, K. Huang, K. Chawla, K. Huang, L. Chen, L. Garg, L. A, L. Silva, L. Bell, L. Zhang, L. Guo, L. Yu, L. Moshkovich, L. Wehrstedt, M. Khabsa, M. Avalani, M. Bhatt, M. Mankus, M. Hasson, M. Lennie, M. Reso, M. Groshev, M. Naumov, M. Lathi, M. Keneally, M. Liu, M. L. Seltzer, M. Valko, M. Restrepo, M. Patel, M. Vyatskov, M. Samvelyan, M. Clark, M. Macey, M. Wang, M. J. Hermoso, M. Metanat, M. Rastegari, M. Bansal, N. Santhanam, N. Parks, N. White, N. Bawa, N. Singhal, N. Egebo, N. Usunier, N. Mehta, N. P. Laptev, N. Dong, N. Cheng, O. Chernoguz, O. Hart, O. Salpekar, O. Kalinli, P. Kent, P. Parekh, P. Saab, P. Balaji, P. Rittner, P. Bontrager, P. Roux, P. Dollar, P. Zvyagina, P. Ratanchandani, P. Yuvraj, Q. Liang, R. Alao, R. Rodriguez, R. Ayub, R. Murthy, R. Nayani, R. Mitra, R. Parthasarathy, R. Li, R. Hogan, R. Battey, R. Wang, R. Howes, R. Rinott, S. Mehta, S. Siby, S. J. Bondu, S. Datta, S. Chugh, S. Hunt, S. Dhillon, S. Sidorov, S. Pan, S. Mahajan, S. Verma, S. Yamamoto, S. Ramaswamy, S. Lindsay, S. Lindsay, S. Feng, S. Lin, S. C. Zha, S. Patil, S. Shankar, S. Zhang, S. Zhang, S. Wang, S. Agarwal, S. Sajuyigbe, S. Chintala, S. Max, S. Chen, S. Kehoe, S. Satterfield, S. Govindaprasad, S. Gupta, S. Deng, S. Cho, S. Virk, S. Subramanian, S. Choudhury, S. Goldman, T. Remez, T. Glaser, T. Best, T. Koehler, T. Robinson, T. Li, T. Zhang, T. Matthews, T. Chou, T. Shaked, V. Vontimitta, V. Ajayi, V. Montanez, V. Mohan, V. S. Kumar, V. Mangla, V. Ionescu, V. Poenaru, V. T. Mihailescu, V. Ivanov, W. Li, W. Wang, W. Jiang, W. Bouaziz, W. Constable, X. Tang, X. Wu, X. Wang, X. Wu, X. Gao, Y. Kleinman, Y. Chen, Y. Hu, Y. Jia, Y. Qi, Y. Li, Y. Zhang, Y. Zhang, Y. Adi, Y. Nam, Yu, Wang, Y. Zhao, Y. Hao, Y. Qian, Y. Li, Y. He, Z. Rait, Z. DeVito, Z. Rosnbrick, Z. Wen, Z. Yang, Z. Zhao, and Z. Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

K. Gupta, B. Thérien, A. Ibrahim, M. L. Richter, Q. Anthony, E. Belilovsky, I. Rish, and T. Lesort. Continual pre-training of large language models: How to (re)warm your model?, 2023. URL https://arxiv.org/abs/2308.04014.

T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation, 2020. URL https://arxiv.org/abs/1903.08560.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre.

Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

S. Hu, Y. Tu, X. Han, C. He, G. Cui, X. Long, Z. Zheng, Y. Fang, Y. Huang, W. Zhao, X. Zhang, Z. L. Thai, K. Zhang, C. Wang, Y. Yao, C. Zhao, J. Zhou, J. Cai, Z. Zhai, N. Ding, C. Jia, G. Zeng, D. Li, Z. Liu, and M. Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies, 2024. URL https://arxiv.org/abs/2404.06395.

A. Ibrahim, B. Thérien, K. Gupta, M. L. Richter, Q. Anthony, T. Lesort, E. Belilovsky, and I. Rish. Simple and scalable strategies to continually pre-train large language models, 2024. URL https://arxiv.org/abs/2403.08763.

G. Ilharco, M. T. Ribeiro, M. Wortsman, S. Gururangan, L. Schmidt, H. Hajishirzi, and A. Farhadi. Editing models with task arithmetic, 2023. URL https://arxiv.org/abs/2212.04089.

Y. Jiang, A. Zhou, Z. Feng, S. Malladi, and J. Z. Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws, 2024a. URL https://arxiv.org/abs/2410.11820.

Z. Jiang, Z. Sun, W. Shi, P. Rodriguez, C. Zhou, G. Neubig, X. V. Lin, W. tau Yih, and S. Iyer. Instruction-tuned language models are better knowledge learners, 2024b. URL https://arxiv.org/abs/2402.12847.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, Mar. 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

S. Kotha, J. M. Springer, and A. Raghunathan. Understanding catastrophic forgetting in language models via implicit inference, 2024. URL https://arxiv.org/abs/2309.10105.

A. Kumar, A. Raghunathan, R. Jones, T. Ma, and P. Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution, 2022. URL https://arxiv.org/abs/2202.10054.

J. Li, A. Fang, G. Smyrnis, M. Ivgi, M. Jordan, S. Gadre, H. Bansal, E. Guha, S. Keh, K. Arora, S. Garg, R. Xin, N. Muennighoff, R. Heckel, J. Mercat, M. Chen, S. Gururangan, M. Wortsman, A. Albalak, Y. Bitton, M. Nezhurina, A. Abbas, C.-Y. Hsieh, D. Ghosh, J. Gardner, M. Kilian, H. Zhang, R. Shao, S. Pratt, S. Sanyal, G. Ilharco, G. Daras, K. Marathe, A. Gokaslan, J. Zhang, K. Chandu, T. Nguyen, I. Vasiljevic, S. Kakade, S. Song, S. Sanghavi, F. Faghri, S. Oh, L. Zettlemoyer, K. Lo, A. El-Nouby, H. Pouransari, A. Toshev, S. Wang, D. Groeneveld, L. Soldaini, P. W. Koh, J. Jitsev, T. Kollar, A. G. Dimakis, Y. Carmon, A. Dave, L. Schmidt, and V. Shankar. Datacomp-lm: In search of the next generation of training sets for language models, 2025. URL https://arxiv.org/abs/2406.11794.

R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, Q. Liu, E. Zheltonozhskii, T. Y. Zhuo, T. Wang, O. Dehaene, M. Davaadorj, J. Lamy-Poirier, J. Monteiro, O. Shliazhko, N. Gontier, N. Meade, A. Zebaze, M.-H. Yee, L. K. Umapathi, J. Zhu, B. Lipkin, M. Oblokulov, Z. Wang, R. Murthy, J. Stillerman, S. S. Patel, D. Abulkhanov, M. Zocca, M. Dey, Z. Zhang, N. Fahmy, U. Bhattacharyya, W. Yu, S. Singh, S. Luccioni, P. Villegas, M. Kunakov, F. Zhdanov, M. Romero, T. Lee, N. Timor, J. Ding, C. Schlesinger, H. Schoelkopf, J. Ebert, T. Dao, M. Mishra, A. Gu, J. Robinson, C. J. Anderson, B. Dolan-Gavitt, D. Contractor, S. Reddy, D. Fried, D. Bahdanau, Y. Jernite, C. M. Ferrandis, S. Hughes, T. Wolf, A. Guha, L. von Werra, and H. de Vries. Starcoder: may the source be with you!, 2023. URL https://arxiv.org/abs/2305.06161.

Z. Lin, Z. Gou, Y. Gong, X. Liu, Y. Shen, R. Xu, C. Lin, Y. Yang, J. Jiao, N. Duan, and W. Chen. Rho-1: Not all tokens are what you need, 2025. URL https://arxiv.org/abs/2404.07965.

S. Longpre, L. Hou, T. Vu, A. Webson, H. W. Chung, Y. Tay, D. Zhou, Q. V. Le, B. Zoph, J. Wei, and A. Roberts. The flan collection: Designing data and methods for effective instruction tuning, 2023. URL https://arxiv.org/abs/2301.13688.

X. H. Lù, Z. Kasner, and S. Reddy. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*, 2024.

K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual associations in gpt, 2023. URL https://arxiv.org/abs/2202.05262.

S. Mindermann, J. Brauner, M. Razzak, M. Sharma, A. Kirsch, W. Xu, B. Höltgen, A. N. Gomez, A. Morisot, S. Farquhar, and Y. Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt, 2022. URL https://arxiv.org/abs/2206.07137.

N. Muennighoff, A. M. Rush, B. Barak, T. L. Scao, A. Piktus, N. Tazi, S. Pyysalo, T. Wolf, and C. Raffel. Scaling data-constrained language models, 2023. URL https://arxiv.org/abs/2305.16264.

Nvidia, :, B. Adler, N. Agarwal, A. Aithal, D. H. Anh, P. Bhattacharya, A. Brundyn, J. Casper, B. Catanzaro, S. Clay, J. Cohen, S. Das, A. Dattagupta, O. Delalleau, L. Derczynski, Y. Dong, D. Egert, E. Evans, A. Ficek, D. Fridman, S. Ghosh, B. Ginsburg, I. Gitman, T. Grzegorzek, R. Hero, J. Huang, V. Jawa, J. Jennings, A. Jhunjhunwala, J. Kamalu, S. Khan, O. Kuchaiev, P. LeGresley, H. Li, J. Liu, Z. Liu, E. Long, A. S. Mahabaleshwarkar, S. Majumdar, J. Maki, M. Martinez, M. R. de Melo, I. Moshkov, D. Narayanan, S. Narenthiran, J. Navarro, P. Nguyen, O. Nitski, V. Noroozi, G. Nutheti, C. Parisien, J. Parmar, M. Patwary, K. Pawelec, W. Ping, S. Prabhumoye, R. Roy, T. Saar, V. R. N. Sabavat, S. Satheesh, J. P. Scowcroft, J. Sewall, P. Shamis, G. Shen, M. Shoeybi, D. Sizer, M. Smelyanskiy, F. Soares, M. N. Sreedhar, D. Su, S. Subramanian, S. Sun, S. Toshniwal, H. Wang, Z. Wang, J. You, J. Zeng, J. Zhang, J. Zhang, V. Zhang, Y. Zhang, and C. Zhu. Nemotron-4 340b technical report, 2024. URL https://arxiv.org/abs/2406.11704.

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.

A. Panigrahi, B. Liu, S. Malladi, A. Risteski, and S. Goel. Progressive distillation induces an implicit curriculum, 2024. URL https://arxiv.org/abs/2410.05464.

G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

J. Phang, T. Févry, and S. R. Bowman. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks, 2019. URL https://arxiv.org/abs/1811.01088.

E. M. Ponti, G. Glavaš, O. Majewska, Q. Liu, I. Vulić, and A. Korhonen. Xcopa: A multilingual dataset for causal commonsense reasoning, 2020. URL https://arxiv.org/abs/2005.00333.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. URL https://arxiv.org/abs/1910.10683.

D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne. Experience replay for continual learning, 2019. URL https://arxiv.org/abs/1811.11682.

F. Schaipp, A. Hägele, A. Taylor, U. Simsekli, and F. Bach. The surprising agreement between convex optimization theory and learning-rate scheduling for large model training, 2025. URL https://arxiv.org/abs/2501.18965.

D. Soboleva, F. Al-Khateeb, R. Myers, J. R. Steeves, J. Hestness, and N. Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 06 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

J. M. Springer, S. Goyal, K. Wen, T. Kumar, X. Yue, S. Malladi, G. Neubig, and A. Raghunathan. Overtrained language models are harder to fine-tune, 2025. URL https://arxiv.org/abs/2503.19206.

Teknium. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023. URL https://huggingface.co/datasets/teknium/OpenHermes-2.5.

M. Weber, D. Fu, Q. Anthony, Y. Oren, S. Adams, A. Alexandrov, X. Lyu, H. Nguyen, X. Yao, V. Adams, B. Athiwaratkun, R. Chalamala, K. Chen, M. Ryabinin, T. Dao, P. Liang, C. Ré, I. Rish, and C. Zhang. Redpajama: an open dataset for training large language models, 2024. URL https://arxiv.org/abs/2411.12372.

K. Wen, Z. Li, J. Wang, D. Hall, P. Liang, and T. Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective, 2024. URL https://arxiv.org/abs/2410.05192.

M. Wortsman, G. Ilharco, J. W. Kim, M. Li, S. Kornblith, R. Roelofs, R. Gontijo-Lopes, H. Hajishirzi, A. Farhadi, H. Namkoong, and L. Schmidt. Robust fine-tuning of zero-shot models, 2022. URL https://arxiv.org/abs/2109.01903.

X. Wu, E. Dyer, and B. Neyshabur. When do curricula work?, 2021. URL https://arxiv.org/abs/2012.03107.

S. M. Xie, H. Pham, X. Dong, N. Du, H. Liu, Y. Lu, P. Liang, Q. V. Le, T. Ma, and A. W. Yu. Doremi: Optimizing data mixtures speeds up language model pretraining, 2023. URL https://arxiv.org/abs/2305.10429.

G. Yang, E. J. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, and J. Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer, 2022. URL https://arxiv.org/abs/2203.03466.

G. Yang, J. B. Simon, and J. Bernstein. A spectral condition for feature learning, 2024a. URL https://arxiv.org/abs/2310.17813.

Z. Yang, N. Band, S. Li, E. Candès, and T. Hashimoto. Synthetic continued pretraining, 2024b. URL https://arxiv.org/abs/2409.07431.

T. Zhang, F. Wu, A. Katiyar, K. Q. Weinberger, and Y. Artzi. Revisiting few-sample bert fine-tuning, 2021. URL https://arxiv.org/abs/2006.05987.

Çağatay Yıldız, N. K. Ravichandran, N. Sharma, M. Bethge, and B. Ermis. Investigating continual pretraining in large language models: Insights and implications, 2025. URL https://arxiv.org/abs/2402.17400.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: Our abstract and introduction makes claims concretely supported by our experiments (refer to abstract and introduction).

   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: We discuss the limitations of our work in the limitations section (refer to limitations).

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: [NA]

Justification: We do not have any theorems.

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: We provide all the information needed to reproduce the main experimental results of the paper in the main paper (refer to appendix).

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: [Yes]

   Justification: We will open source our code for launching runs, as well as logs for the loss values of each run.

   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: We provide all the details necessary to reproduce the main experimental results of the paper in the appendix (refer to appendix).

   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: Though we can not report error bars due to the high computational cost associated with training language models, we show clean scaling curves to assure that our experiments are statistically significant (refer to plots).

   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).
   - 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: We detail our compute resources in the appendix (refer to appendix).

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: We follow the NeurIPS Code of Ethics.

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: [Yes]

Justification: We discuss the potential positive and negative societal impacts of our work in the broader impacts section (refer to broader 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: We do not release any data or models that have a high risk for misuse.

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: We properly credit the creators of the assets we use (refer to appendix).

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.
- 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: We do not release any new assets.

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: We do not involve crowdsourcing or research with human subjects.

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: We do not involve crowdsourcing or research with human subjects.

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.

A Data schedule equivalences

As discussed in Section 3.2, data schedules have two degrees of freedom. However, they can be described with many intuitive variables, each a few equations away from each other. We can use the following variables to describe the data schedule:

- **Total training steps** T: the total number of training steps.
- **Rare step fraction** γ: the fraction of training steps that are rare, after deciding the repetition count.
- **Replay fraction** ρ: the fraction of pre-training data that is replayed.
- **Rare stage 2 allocation** α: the fraction of the total rare data that is allocated to stage 2.
- **Stage 2 duration** δ: the fraction of training that is in stage 2.
- **Stage 1 rare weight** w_1: the weight of the rare data in stage 1.
- **Stage 2 rare weight** w_2: the weight of the common data in stage 2.

If there are 7 variables, why are there only 2 degrees of freedom? The first two variables are set by problem setting and we do not control them (besides the repetition count, which we treat as fixed for this section). For the rest of this section, without loss of generality, we set $T = 1$. We claim that given the next 2 variables, you can derive the other 3.

If the replay fraction is ρ, then the stage 2 rare weight is automatically fixed as $w_2 = 1 - \rho$. If the stage 2 allocation is α, then we know that the number of rare steps in stage 2 is $\alpha\gamma$. This means that the number of total steps in stage 2 is $\frac{\alpha\gamma}{1-\rho}$, giving the stage 2 duration δ. Now that we know δ, it suffices to determine the stage 1 rare weight. We know that there are $\gamma(1-\alpha)$ rare steps in stage 1, as well as $1 - \delta$ total steps. This means that the stage 1 rare weight is $w_1 = \frac{\gamma(1-\alpha)}{1-\delta}$. Therefore, we've recovered all 7 variables from the 2 degrees of freedom. You can confirm that with these choices of w_1, w_2 that $w_1(1-\delta) + w_2\delta = \gamma$.

B Potential failure modes of fine-tuning

We discuss potential conceptual failure modes of fine-tuning here, using a mix of experiments and toy models for intuition.

B.1 Instability of fine-tuning

We notice that at the start of fine-tuning, there is a pretty large loss spike (Figure 10). This is especially true at higher learning rates. However, after some steps of training, the loss goes below where it started. In fact, it is still correct to use higher learning rates even though they make the spike larger because you end up with a lower loss.

One relatively vague hypothesis is that the loss spike is the reason fine-tuning underperforms replay. This can happen in at least two concrete ways

1. When there is replay data, there is less distribution shift between stage 1 and stage 2. Therefore, the loss spike is less pronounced, and there is less steps spent recovering from it.
2. There seems to be a minimum number of steps before one recovers from the loss spike. Since replay increases the number of steps in stage 2, it can perhaps give more time to recover from the loss spike.

We believe further experimentation is necessary to fully understand the nature of this spike, specifically whether it is harmful for model performance and at what data regime it matters the most in.

B.2 Overfitting to rare data

It is known that a fixed un-regularized model has a tendency to overfit to small sample counts. Therefore, it's possible that fine-tuning suffers from this problem. To model this, we set up a simple linear regression toy model.

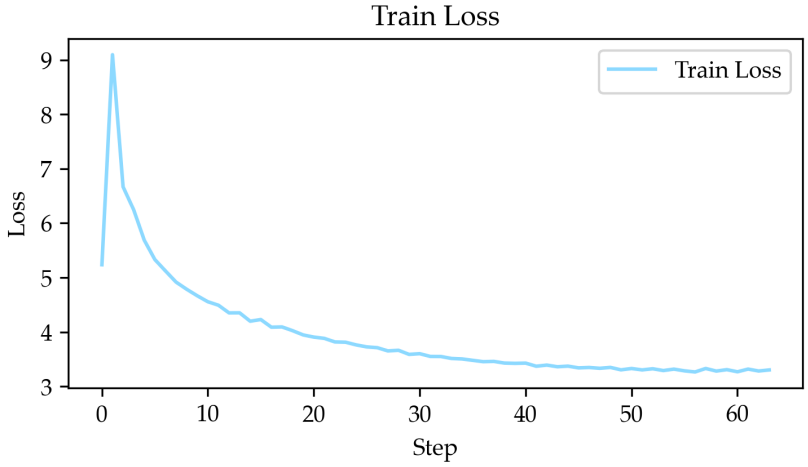


Figure 10: **Train loss spike.** We notice a large loss spike during the first few steps of training across our settings. This represents one barrier to training with a limited number of samples. Replay might help either because (1) there is less distribution shift between stage 1 and stage 2 or (2) there is more time to recover from the spike.

We construct a data distribution in 400 dimensions, governed by a pre-training $\theta_{\text{PT}} \sim \mathcal{N}(0, I)$ and a fine-tuning $\theta_{\text{FT}} \sim \mathcal{N}(\theta_{\text{PT}}, 0.1I)$. We then construct a dataset of N pre-training points and n fine-tuning points. Each point is generated as $(x, \theta^\top x + \epsilon)$ for $x \sim \mathcal{N}(0, I)$ and $\epsilon \sim \mathcal{N}(0, 1)$.

Our training algorithm first "pre-trains" by performing OLS on the pre-training points. For sufficiently large $N >> d$, this will easily fit the pre-training vector θ_{PT}. However, our goal is to learn the fine-tuning vector θ_{FT}. We do this by performing OLS on the residuals of the pre-training fit and the true labels for the fine-tuning points. The fine-tuning now benefits from the pre-training fit bringing the parameters closer to the true fine-tuning vector. When we are in the over-parameterized regime, we use min-norm least squares regression, as double-descent literature has shown this is known to generalize better and is reflective of deep learning inductive bias [Belkin et al., 2019, Hastie et al., 2020]. We then measure error as mean squared error between the learned parameter and true fine-tuning parameter. We visualize these results in Figure 11, purple line. We see that for $n < d$, the model overfits to the noise in the fine-tuning data, resulting in higher error than random guessing. We plot the gray line to track the best possible error achievable by the model for a given n by using any sample count up to n.

We now introduce replay: mix in some pre-training data for the fine-tuning OLS. The replay significantly reduces the overfitting as we see the in the other lines.

In this setting, it is known that the Bayes-optimal estimator involves appropriately tuning the ridge regularization parameter. We visualize differing values of the ridge parameter in Figure 12, orange line. We see that the optimal ridge parameter is non-zero. Moreover, the best ridge parameter achieves much better loss than tuned replay count. If this intuition holds, real language model training should benefit from finding the correct notion of regularization. One might expect the natural analogue of ridge regression for language models is weight decay. As discussed in E.1.3, weight decay does not significantly improve the loss and under-performs optimal replay. This tells us we need to rethink how we regularize fine-tuning to extract the full value of rare data points.

B.3 Model size

One concern is that the necessity of replay is due to the model size. To see whether any explanation related to model size holds ground, we see whether changing model size changes the necessity of replay. To measure this, we take our joint training setting with a fixed learning rate schedule and $\alpha = 1.0$ (all rare data in stage 2) and vary the model size. When we increase model size, we decrease the learning rate inversely proportionally to the width of the hidden dimension, following folklore scaling practices [Everett et al., 2024]. We visualize the results in Figure 13. We see that across all

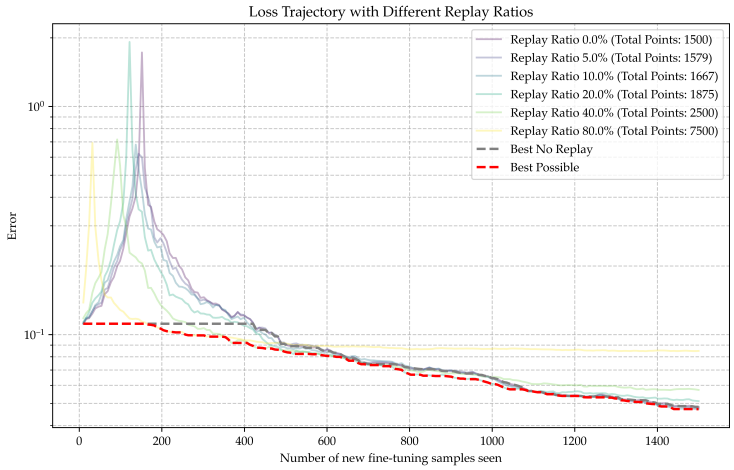


Figure 11: **Linear regression with replay.** We plot the loss of a linear regression model as a function of the number of fine-tuning points n for different values of the replay fraction ρ. We see that for $n < d$, the model overfits to the noise in the fine-tuning data, resulting in higher error than random guessing. We plot the gray line to track the best possible error achievable by the model for a given n by using any sample count up to n. Replay significantly reduces the overfitting, resulting in better MSE than random guessing. We track the best possible error as a function of n for the best ρ as the red line. For a regime of some but not too many rare points, replay improves over standard fine-tuning.

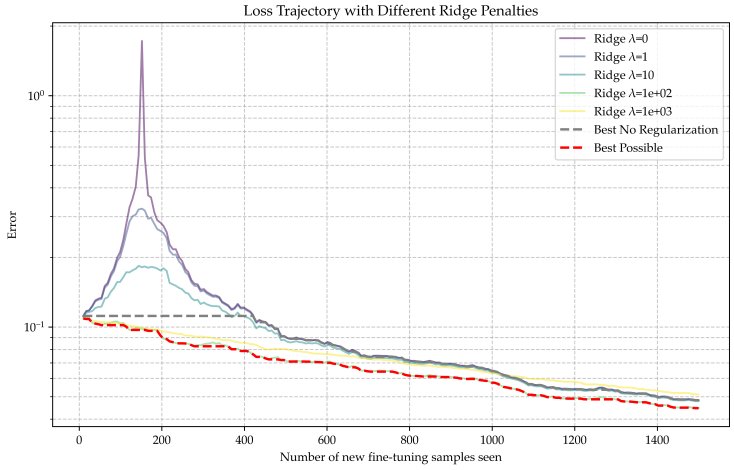


Figure 12: **Linear regression with ridge regularization.** We follow the same setup as in Figure 11, but instead of tuning replay fraction, we tune the ridge regularization parameter λ. We see that the optimal ridge parameter is non-zero and that the best ridge parameter achieves much better loss than tuned replay count.

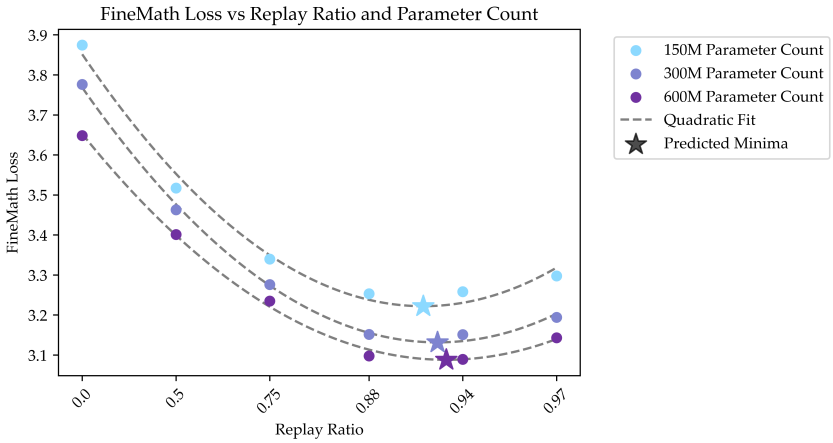


Figure 13: **Model scaling.** We take our standard 4B token training setup and scale the parameter count to 4x parameter count. For a given size, we sweep for the optimal replay fraction ρ while reserving all the rare data for stage 2 ($\alpha = 1.0$). We find that larger models still require replay data to get lower loss.

sizes, the model benefits from replay. Furthermore, the benefit of replay is relatively consistent across model sizes.

One interesting implication of this finding is that one can determine the optimal data schedule for a large model by tuning on a small model. This resembles μP style arguments [Yang et al., 2022, 2024a] for setting layer-wise learning rates at small model counts.

C Note on tuning appendices

This project has spanned a lot of experiments. The order these were conducted in does not reflect the order in which they are presented. At a high level, most of the mid-training/pre-training experiments were done first, giving intuition for what the results would look like for supervised fine-tuning. This means the experiments are more comprehensive for mid-training as it was when there was a worse understanding of the relationship between problem parameters. As the project developed, we were able to reduce search spaces by good priors on what hyperparameters worked (though we always certified they were correct as shared in the plots).

We believe it is more instructive to read the guide for how to set mid-training hyperparameters. We believe the literature lacks rigorous experiments (mostly deciding random hypers on the fly). However, this stage can give large data efficiency gains if done correctly.

D General training settings

We train a 150 million parameter Llama-style language model with context length 4096 for 4B tokens. This is close to Chinchilla optimal scaling [Hoffmann et al., 2022] which prescribes 20x tokens per parameter. We train with batch size 1024 for 1024 steps with weight decay 0.1. We use the Adam optimizer with default parameters. When we tune learning rate, we search for the nearest power of 3 assuming the final loss is convex in the learning rate. For the other models, we scale the learning rate inversely with the model width following the reccomendations of [Everett et al., 2024] (see Table 1).

For our common data, we use C4 [Raffel et al., 2023] since it is filtered but relatively uncurated. For example, C4 filters out all code data by removing documents with curly braces. For our rare data, we have domains representing math (FineMath [Allal et al., 2025]), coding (StarCoder [Li et al., 2023]), and instruction following (Flan [Longpre et al., 2023]). Our validation datasets are always the same distribution as our training data.

Parameter	150M	300M	600M	1.9B
Context Length	4096	4096	4096	4096
Hidden Dimension	512	768	1024	2048
Intermediate Dimension	1792	2688	3584	7168
Attention Heads	8	12	16	16
KV Heads	8	12	8	8
Layers	6	12	24	24
Learning rate	3e-3	2e-3	1.5e-3	7.5e-4

Table 1: Model architecture configurations for different model sizes. All models use the Llama architecture with a standardized context length of 4096 tokens. We default to the 150M model if not specified.

D.1 Model configurations

D.2 Magic number justifications

Selecting a training regime requires setting some arbitrary numbers. We give justification for some here.

- Rare data fraction: Pre-training token counts are on the order of 10 trillion while domains are on the order of 10 billion tokens, motivating our choice of $\approx 0.1\%$. We actually use a rare data fraction of $\frac{1}{1024}$; this better interacts with our block-deterministic data scheduler which draws/shuffles 2048 sequences at a time.
- Replay fractions and rare data allocations: In early experiments we quickly realized that the dependence on these parameters scaled nicely when the replay fractions are spaced out by $\log(1-x)$. Therefore, we equally spaced out values. In plots, we round all values to two decimals. In actuality, our replay fractions were $0.25, 0.5, 0.75, 0.875$ and our rare data allocations were $1.0, 0.5, 0.25, 0.125$. The power of 2 scaling similarly interacts nicely with our block-deterministic data scheduler.
- Model size: 150M reflects a model scale that is large enough to be represantative and scale nicely. It also enabled quicker iteration than larger scale models. During the course of this project, we sanity checked our results held at larger scales, increasing our confidence in using smaller scale models.

E Mid-training experiments

E.1 Fine-tuning baseline

E.1.1 Repetitions

We try varying the number of repetitions of the rare data during mid-training. For this tuning, since we do not know the learning rate and schedule yet, we tune across the two most promising learning rates of 1e-3 and 3e-3 with no learning rate cooldown (as this is closer to our final learning rate schedule than full decay). We visualize the best of both in Figure 14. We find that we can tolerate up to 32 repetitions of the rare data before overfitting across all domains.

E.1.2 Learning rate cooldown

We vary the cooldown duration of a standard WSD learning rate schedule with 10 step warmup while fixing all 32 repetitions of the rare data to appear at the end of training. We also vary the learning rate to be 1e-3, 3e-3, and 1e-2. For all training runs, 3e-3 did best and 1e-2 always diverged, so we can safely only look at WSD with learning rate 3e-3. We visualize the final result in Figure 3. We find it critical to have a cooldown period as opposed to the more conventional long cooldown period (cooldown duration 0.99).

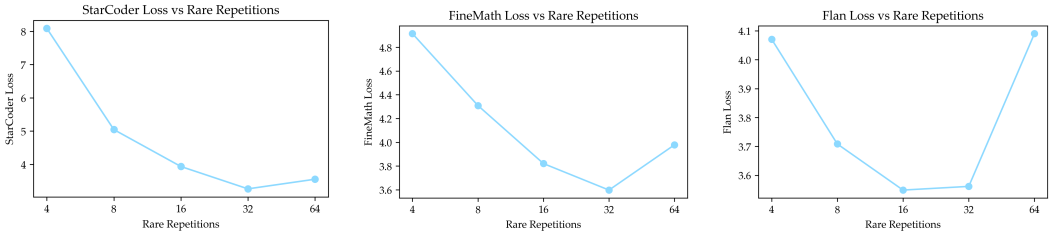


Figure 14: **Mid-training tuning repetitions.** We show that across our mid-training domains, we can tolerate maximum 32 repetitions of the original data before overfitting to the rare data. Note that the x-axis is more comprehensive compared to 16.

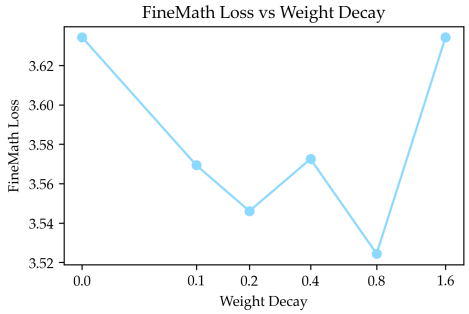


Figure 15: **Tuning weight decay.** We tune the weight decay for the fine-tuning baseline fixed to our above repetition count and learning rate cooldown. The effect is small and noisy, so we default to 0.1, at the upper range of weight decays used in the literature.

E.1.3 Tuning weight decay

We try tuning the weight decay for the fine-tuning baseline fixed to our above repetition count and learning rate cooldown. We visualize the results in Figure 15. We find that weight decay does give minimal but noisy effect on loss. Due to this variance, we decide to stay closer to the range of weight decays used in the literature which are usually around 0.05 (for example, Table 10 and 11 in the appendix of Li et al. [2025]) However, since there does seem to be a benefit of slightly higher weight decay, we use 0.1 for all of our mid-training experiments. We also carry this choice to our pre-training experiments.

F Post-training experiments

F.1 Fine-tuning baseline

F.1.1 Repetitions

We try varying the number of repetitions of the rare data during fine-tuning. We visualize the loss for different repetition counts in 16. We find that we can tolerate up to 64 repetitions of the rare data before overfitting across all domains.

F.1.2 Learning rate

For the model pre-trained on C4, we tried different learning rates for 1000 training steps and picked the model with the best C4 loss. This ended up being 1e-3 (which acheived 4.12 loss, relative to 3e-3 which achieved 4.22 and 3e-4 which achieved 4.66). This learning rate was used across all pre-trained models, which ranged from 512 to 992 steps. Note that the optimal learning rate is different for this setting compared to the mid-training experiments because we're using a different learning rate.

G Data efficiency

It is important to compare how well two training algorithms leverage the same amount of fixed samples. Though we can compare the direct loss, this gives a misleading impression of how what the

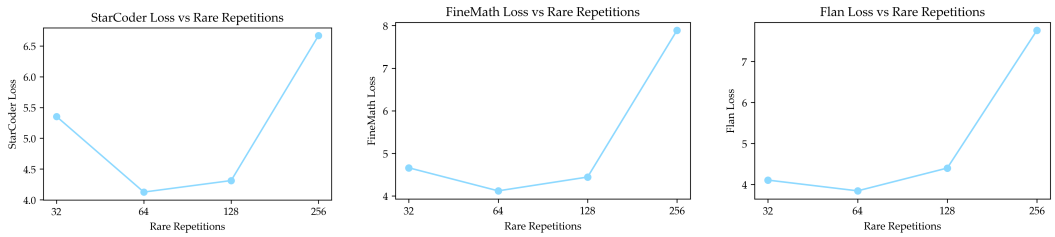


Figure 16: **Fine-tuning repetitions.** We show that across our fine-tuning domains, we can tolerate maximum 64 repetitions of the original data before overfitting to the rare data.

actual improvement is. To bring this into a human-friendly metric, we introduce the data efficiency multiplier. We use the following procedure:

1. **Fix a reference training algorithm.** To enable comparison across a broad suite of possible training strategies without having to refit scaling laws, we first fix a reference training algorithm S_{ref} that characterizes one natural usage of the training data. We detail our choice of reference algorithm below.
2. **Build a scaling law.** We now build a reference scaling law to characterize the loss of the reference algorithm as a function of the number of rare tokens it gets to see. Since the model size is fixed, we fit a power law from number of data points seen to loss (more details below).
3. **Effective data points.** For any given strategy S, we can find it's loss. With this loss, we can see how many data points $D(\mathsf{S})$ it would take for the reference algorithm to match the loss of S. If this number is high, then S is very data efficient.
4. **Normalize for reference.** However, our current metric strongly depends on the choice of reference algorithm. To make this metric useful regardless of the reference algorithm, we always report a *data efficiency improvement*. Specifically, we only use this metric to compare the data efficiency of two strategies S_1 and S_2. We then report the data efficiency improvement of S_2 over S_1 as $\frac{D(\mathsf{S}_2)}{D(\mathsf{S}_1)}$. A large improvement means that S_2 is much more data efficient than S_1.

We now go into details about how we fit the scaling law.

G.1 Scaling law formulation

We fix the model size and total number of tokens. We then fit a power law from number of rare data points given to loss. Our power law has the form $L(D) = aD^b + c$ for loss L, number of rare data points D, and free variables a, b, c. We use `scipy.optimize.curve_fit` to fit the scaling law.

G.2 Training runs

We fix learning rate schedule to be cosine and tune learning rate to be 0.003. When tuning epoch count, we found that the model could not tolerate more than 32 epochs of rare data at larger rare data fractions. Therefore, we fix this epoch count. We also mix data uniformly throughout training instead of using a dedicated data schedule. We train for a total of 4B tokens and vary the number of rare tokens to be 4M, 8M, 16M, 32M, 64M tokens. Since we train for 32 repetitions, our largest rare data run trains on rare tokens for 50% of training.

H WSD tutorial

It turns out that the correct learning rate schedule is critical for improving rare data efficiency. Though annealing-based learning rate schedules give the largest benefit for data ordering, we have relatively little intuitive/empirical understanding of how to properly use them. We will provide a quick introduction to how they work in the un-ordered setting, and then show how to use them in the ordered setting.

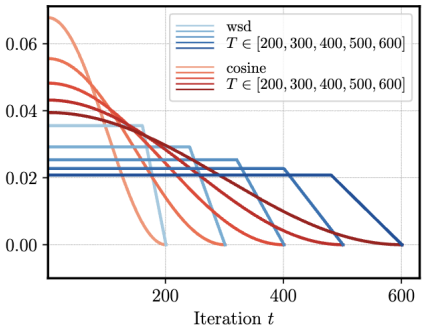


Figure 17: **Learning rate schedule.** This figure shows the shape of a WSD learning rate schedule in contrast to a cosine learning rate schedule (both without warmup). Figure is taken from Schaipp et al. [2025], Figure 2 left.

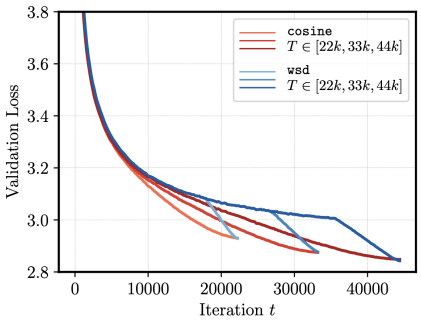


Figure 18: **Loss curves.** This figure shows the loss curves for a WSD learning rate schedule and a cosine learning rate schedule. Note that the loss of a WSD schedule initially makes slower progress than a cosine schedule and then makes up for it with much faster loss improvement at the end. Figure is taken from Schaipp et al. [2025], Figure 1 left.

H.1 What is WSD?

Warmup-Stable-Decay (WSD) is a learning rate schedule with three phases:

1. Warmup: The learning rate is increased linearly from 0 to a peak value.
2. Stable: The learning rate is held constant at the peak value.
3. Decay: The learning rate is decayed linearly from the peak value to 0.

We visually depict this learning rate schedule (without warmup) in Figure 17, left.

H.2 Standard training (random order)

Though warmup is important, the exact duration of the warmup is not critical. In contrast, the decay period is critical to the final loss. We visualize the loss curves for a WSD learning rate schedule and a cosine learning rate schedule in Figure 18, right. Notably, as soon as the learning rate starts decaying, the loss improvement is much faster. This is contrast to cosine learning rate schedules where the loss improvement actually slows down at the end of training (with a characteristic curl up). These different rates of decrease have historically been really important details: for example, fitting the scaling laws to intermediate checkpoints gives incorrect scaling laws since the models have been annealed for different durations [Hoffmann et al., 2022].

Why does this happen? One nice intuitive picture is given by the river valley landscape explanation in Wen et al. [2024]. They posit that the loss landscape looks like a single river flowing down in the middle of a valley (Figure 19). In this picture, you would like to both get to the bottom of the valley and go far down the river. A standard learning rate schedule slowly descends the valley while also making progress along the river direction. The paper's central claim is that WSD instead stays at the top of the valley but continues to make progress along the river direction. Then, when one anneals the learning rate, it starts descending donw the valley, revealing the true progress made by the model not captured by the loss. It is noted in the Edge of Stability literature [Cohen et al., 2022, 2024] that staying at high learning rate is better for performance even though there is large oscillation in the loss.

Though this picture is helpful visually, there is an even simpler theoretical picture. The works Defazio et al. [2024], Schaipp et al. [2025] show that the simple theoretical model of non-smooth convex optimization predicts the shape of the loss curve. Specifically, the upper-bound on the loss from

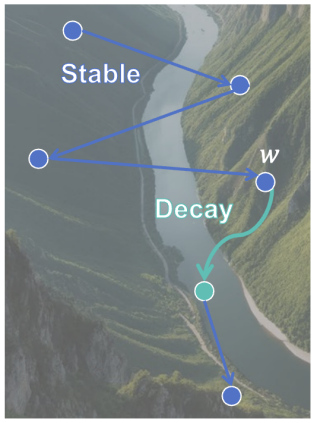


Figure 19: **River valley landscape.** This shows the intuitive picture of the river valley landscape. WSD makes progress along the river direction while making all the hill progress at the very end. Figure is taken from Wen et al. [2024], Figure 2 left.

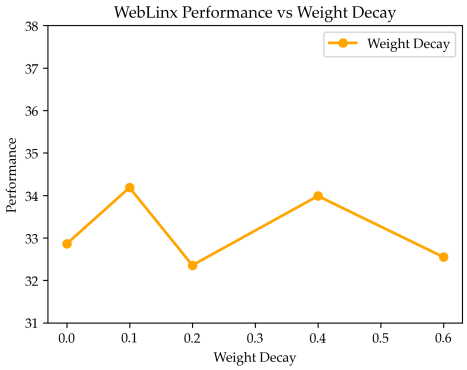


Figure 20: **Weblinx weight decay ablation.** We fine-tune Llama 3.1-8B Instruct on Weblinx demonstrations. We find that without replay, tuning weight decay gives little gains, improving by less than 2% over the baseline.

standard online convex optimization arguments applied to the last iterate of training matches the "shape" of the WSD loss curve.

H.3 Ordered training

Why does it matter that WSD decreases loss faster at the end of training? Intuitively, if the loss is decreasing faster, placing high quality data at the end of training is more important. We algorithmically leverage this intuition by placing the rare data at the end of training with WSD. In some earlier experiments, we found that when using a cosine learning rate schedule, it actually hurt to keep rare data at the end of training relative to placing it uniformly throughout training.

I Web agents

We train with the same hyperparameters as the original Weblinx paper on a subset of the demonstrations from the original paper. We use the same evaluation protocol and metrics as the original paper by combining the validation and in-distribution test set. We defer to the original paper for more details on the data and evaluation. When we specify replay fraction, we are doing the replay on a document level instead of a token level. This doesn't have large implications since all peak at an intermediate value and fine-tuning isn't computationally intensive.

We provide an additional ablation on tuning weight decay while fine-tuning on web agents data. We find that without replay, this gives little gains, improving by less than 2% over the baseline. We provide the results in Figure 20.

J Basque

We tune the learning rate to be 1e-5 for fine-tuning on Basque. We find the gain in accuracy to be real across different token counts, displayed for 40M tokens and 200M tokens in Figure 21.

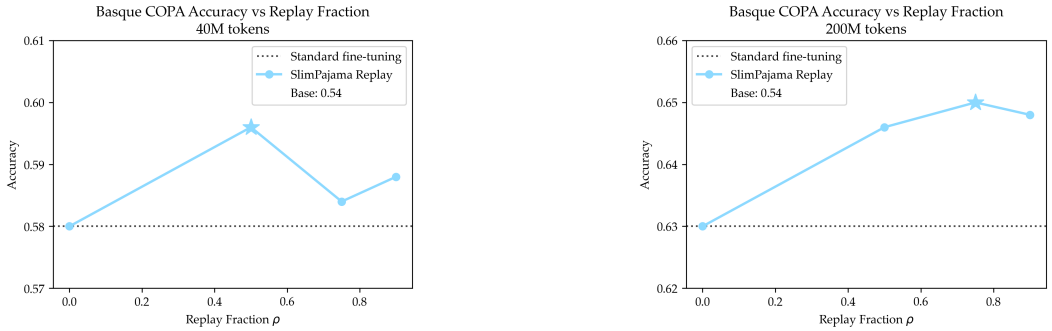


Figure 21: **Basque training with different token counts.** We try Basque training with 40M and 200M tokens to confirm that the gain in accuracy is real.

When tracking Basque loss, we find that there is a spike in loss at the start of training only if the learning rate is sufficiently high. In practice, it is worth using training with this higher learning rate for best Basque loss/accuracy. We find that the loss improvement of replay decreases as we increase the total token count. However, there continues to be an accuracy gain as you increase the token count, showing that replay may be even more important for evaluation metrics.

K Detailed related work

K.1 Repeating data

Prior work on data-constrained scaling laws [Muennighoff et al., 2023, Goyal et al., 2024] predict that as you continue to repeat data, loss improves at a diminishing rate. However, the specific decay formulation predicts that it will asymptote at a particular value.

Specifically, the simplest decay formulation presented in both works estimates that when you see n data points for the second time, it is like effectively training on $n\delta$ data points for decay factor δ. For the k-th repetition, it is like training on $n\delta^{k-1}$ data points. In the infinite data limit, these scaling laws predict that the loss will asymptote at a particular value, specifically at the loss of seeing $\frac{n}{1-\delta}$ fresh data points once.

In our experiments, we found this not to be the case. If we repeated a rare domain too many times, the loss would start going up. This is true whether it is fine-tuning or it is generic pre-training data. This means we think more carefully about how to leverage rare data.