Scaling Laws for Upcycling Mixture-of-Experts Language Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Pretraining large language models (LLMs) is resource-intensive, often requiring months of training time even with high-end GPU clusters. There are two approaches of mitigating such computational demands: reusing smaller models to train larger ones (upcycling), and training computationally efficient models like mixture-of-experts (MoE). In this paper, we study the upcycling of LLMs to MoE models, of which the scaling behavior remains underexplored. Through extensive experiments, we identify empirical scaling laws that describe how performance depends on dataset size and model configuration. Particularly, we show that, while scaling these factors improves performance, there is a novel interaction term between the dense and upcycled training dataset that limits the efficiency of upcycling at large computational budgets. Based on these findings, we provide guidance to scale upcycling, and establish conditions under which upcycling outperforms from-scratch trainings within budget constraints.

1 Introduction

Large-scale neural network architectures, such as dense transformers [Vaswani et al., 2017], have seen remarkable success across a wide range of tasks, particularly achieving human-level capabilities in natural language processing [Achiam et al., 2023]. However, they often demand an enormous amount of computation, imposing challenges of computational efficiency and scalability. Sparse models like mixture-of-experts (MoE) architectures [Shazeer et al., 2017, Lepikhin et al.], have emerged as an alternative achieving better efficiency-performance trade-off via partial activation (routing) of neural parameters when processing the input. Even so, MoE still requires substantial compute power to reach its full potential [Wei et al., 2024, Dai et al., 2024, Yang et al., 2024].

One direction to further accelerate training convergence is leveraging smaller pretrained models to guide the training of larger MoE models. Komatsuzaki et al. propose upcycling, which reuses the dense checkpoint to continued pretrain the upcycled MoE. The MoE is expected to specialize and optimize routing rapidly by leveraging the pretrained dense model weights.

Despite the promise of efficient MoE training via upcycling, the effectiveness and limitations of this technique remain unclear. While some [Wei et al., 2024, He et al., 2024] have already adopted it to training large-scale MoEs, Muennighoff et al. [2024] reported negative results where upcycling can slow down training convergence. We believe these seemingly contradictory conclusions are due to insufficient comprehensive studies and assessments. There is also a lack of guidance on how and when to upcycle MoE, hampering a wider adoption of this technique.

In this paper, we seek to better understand large language model's (LLM) upcycling to MoE via a series of controlled experiments, spanning up to a few hundred billion (B) training tokens and models up to 7B total parameters. Specifically, we uncover precise power-law scalings for the language

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

modeling performance (cross-entropy loss) with respect to training dataset size (for both dense and upcycled MoE training), and the model configuration, including the total number of parameters (model size). Building on these results, we provide a framework for assessing when upcycling offers advantages over from-scratch training and how performance gains depend on dataset size and model configuration.

Main results. The major technical findings of this paper are summarized below. Let D_1, D_2 be the number of tokens used to train the dense model and upcycled MoE respectively. Denote the cross-entropy test loss of the upcycled MoE by L. We find that the upcycled MoE satisfies the following relation for a wide range of model configuration:

$$L = AD_1^{-\alpha_1} D_2^{-\alpha_2+\alpha_3 \log D_1} + E \tag{1}$$

where α_i's ($i = 1, 2, 3$) are positive scaling exponents, and A, E are constants independent of D_1, D_2.

2 Experimental Design and Results

In this Section, we describe the design of our experiments, before presenting some of the experimental results. Additional details, including ablation studies are available in Appendix C.

Dense models in consideration. We train a suite of dense models with model sizes 15M, 44M, 0.1B, 0.2B, 0.5B and 1B. The model configuration (number of layers, n_{layer}, hidden dimension d_{model}, and MLP hidden dimension, d_{mlp}) used in this paper is summarized in Table 3 of Appendix C.

Learning rate schedule. We are interested in training models with different numbers of training token budget. Previous work [Hoffmann et al., 2022] used the cosine learning rate (LR) schedule and trained separate models for *each* number of training token budget, which is resource-consuming. We instead employ the warmup-stable-decay (WSD) learning rate schedule [Bi et al., 2024, Hu et al., 2024], which requires only a single model sweep with a sufficiently large number of training tokens.

Dataset. We use training dataset derived from the CommonCrawl portion of Slimpajama-DC [Shen et al., 2023], containing 368B tokens in total. The test loss is calculated from the default validation set (0.3B tokens) defined therein. In Appendix D, we train models on two different datasets (Japanese language and source code datasets) to study the scaling behavior across datasets. We show some of the resulting test loss curves of our upcycling experiments in Figure 3 (see model evaluation with standard benchmarks in Appendix C.9).

3 Scaling Laws

3.1 Scaling Law for Dataset Sizes

We fix the model size while varying $D_{1,2}$ to study the scaling behavior with respect to these variables. To determine the functional form of the scaling law, $L(D_1, D_2)$, we require it to satisfy certain properties:

Requirement 1. $L(D_1, D_2)$ follows the power law with respect to D_2, as shown empirically in Figure 1. This aligns with the power-law ansatz, treating upcycling as analogous to standard MoE training with dataset size D_2, initialized with dense parameters rather than random weights:

$$L(D_1, D_2) = L_{D_1}(D_2) = AD_2^{-\alpha} + E \tag{2}$$

Requirement 2. As $D_2 \to 0$, the loss should reduce to the power-law scaling behavior of the dense counterpart with respect to D_1, consistent with the function-preserving initialization of upcycling:

$$\lim_{D_2 \to 0} L(D_1, D_2) = AD_1^{-\alpha} + E$$

We investigate the following functional forms of power law satisfying these requirements:

Multiplicative:

$$L(D_1, D_2) \approx AD_1^{-\alpha_1} D_2^{-\alpha_2+\alpha_3 \log D_1} + E \tag{3}$$

Additive:

$$L(D_1, D_2) \approx AD_1^{-\alpha_1} + FD_2^{-\alpha_2+\alpha_3 \log D_1} + E \tag{4}$$

Both forms include an *interaction* term ($\alpha_3 \log D_1$) in the scaling exponent, capturing the interplay between D_1 and D_2.

Empirical comparisons of functional forms. We empirically compare these functional forms, including the special case where D_1 and D_2 have no interaction. The optimization uses the Huber loss ($\delta = 10^{-3}$) and the BFGS algorithm, fitting the logarithm of the loss via the LogSumExp trick applied to the RHS of Equations 3 and 4. The leave-one-out root mean square error (RMS) serves as the fit metric.

The fit is performed on a 0.1B dense model upcycled to MoE architectures with $n_{\text{expert}} = \{4, 8\}$ and $n_{\text{TopK}} = \{1, 2\}$, trained on a 5×5 grid of D_1, D_2. The fitting results are shown in Table 1, where we can see that the multiplicative functional form (with non-zero interaction) achieves consistently the lowest leave-one-out RMS error across the experimented MoE architectures. Henceforth, we adopt Equation 3 in our scaling laws.

3.1.1 Interpretations

Several quantitative and qualitative observations can be made from the scaling law of dataset sizes. From our fit (Table 8 in Appendix E), we notice a trend $\alpha_2 \gtrsim \alpha_1 \gg \alpha_3$. This means that while increasing either of D_1 and D_2 helps improve performance, as upcycled training has a slightly larger exponent (α_2), increasing D_2 helps train faster.

Upcycled MoE has a better head start (effective scaling factor is smaller). Fixing D_1, we see that the effective scaling factor for D_2 is $AD_1^{-\alpha_1}$. Increasing D_1 lowers the effective scaling factor, and hence the loss of upcycling. Indeed, fixing D_2, we see that the model performs better with increasing D_1 in Figure 3.

Upcycled MoE trains slower with larger sunk cost (effective scaling exponent is smaller). Again fixing D_1, we see that the effective scaling exponent with respect to D_2 is $\alpha_2 - \alpha_3 \log D_1$. This means that the larger the sunk cost (D_1) is, the loss decreases more slowly with D_2, indicating diminishing returns from increasing D_2 at higher D_1 values, agreeing with Figure 1's results.

(n_{expert},n_{TopK})	(4,1)	(4,2)	(8,1)	(8,2)
Mul.	**0.0111**	**0.0081**	**0.0105**	**0.0031**
Mul. ($\alpha_3 = 0$)	0.0169	0.0085	0.0180	0.0095
Add.	0.0165	0.00843	0.0167	0.0093
Add.($\alpha_3 = 0$)	0.0196	0.0117	0.0430	0.0113

Table 1: **Multiplicative scaling law with interaction consistently achieves lowest error.** Leave-one-out RMS error for fitting the loss for MoEs upcycled from a dense 0.1B model, with functional forms of Equations 3, 4, and specific cases with $\alpha_3 = 0$. The first and second number in the bracket of the first row indicates the MoE architecture's parameter, n_{expert} and n_{TopK} respectively.

3.2 Joint Scaling Law

In this Section, we show that the upcycled MoE follows a joint scaling law with respect to dataset size and model size, provided the MoE architecture is fixed. For our study, we adopt a widely used MoE configuration ($n_{\text{expert}} = 8, n_{\text{TopK}} = 2$), which has been implemented in several large-scale, publicly available models, including Mixtral-8x7B and 8x22B [Jiang et al., 2024].

The straightforward functional form is as follows.

$$L(D_1, D_2, N_2) = AD_1^{-\alpha_1} D_2^{-\alpha_2 + \alpha_3 \log D_1} + BN_1^{-\beta_2} + E \tag{5}$$

Fitting. Towards this end, we fit three separate scaling laws, corresponding to dense, MoE, and upcycled trainings, incorporating dataset and model sizes (the first two are fitted with the functional form of Equation 7). In Figure 4, we show that the fitted result of upcycling scaling law extrapolates well, achieving validation RMS error of 0.015.

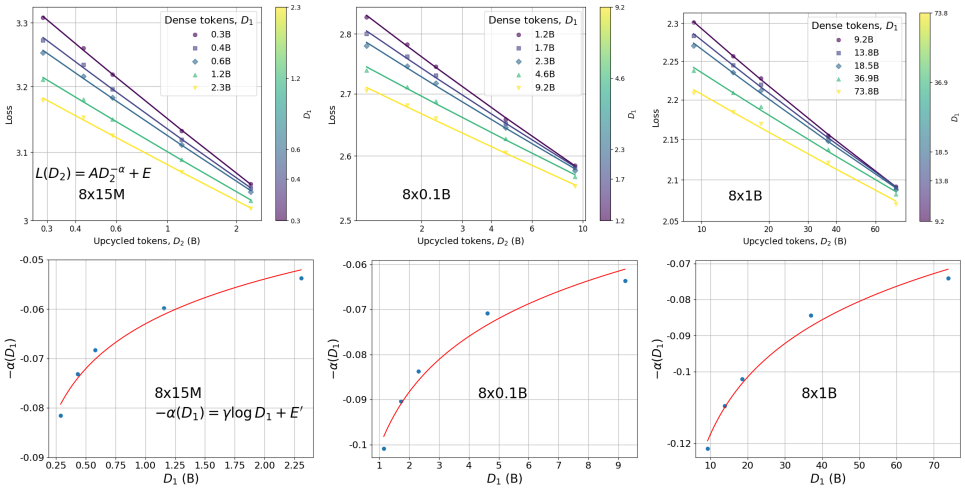

Figure 1: **Top: D_2 has power-law scaling.** We show scaling behavior of upcycled training tokens (D_2) for different values of dense tokens (D_1). **Bottom: Interaction term explains decreasing exponents.** The fitted exponents in the upper plots are used to fit the logarithm of D_1, and are shown to agree well with the functional form.

3.3 Training MoE from Scratch versus Upcycling

Past studies [Komatsuzaki et al., Muennighoff et al., 2024] have explored the efficiency of training an MoE from scratch compared to upcycling a dense model. Komatsuzaki et al. found that upcycling retains an advantage up to 120% of the sunk cost (D_1). For instance, to match the performance of an upcycled MoE trained with an additional 0.4 trillion (T) tokens after 2T dense tokens, training an MoE from scratch would require 2.4T tokens—effectively saving 2T tokens in upcycled training. Conversely, Muennighoff et al. [2024] reported that training from scratch requires less than 100% of the sunk cost under different settings, indicating that it could be more efficient. These seemingly contradictory results suggest that upcycling efficiency depends on both sunk cost and model size.

To investigate this, we define D^* the number of tokens required for training from scratch to match the performance of an upcycled MoE with the same sunk cost, i.e.,

$$L_{N_1}^{\text{MoE}}(D^*) = L_{N_1}^{\text{Upc.}}(D_1 = D^*, D_2 = D^*) \tag{6}$$

Since the above equation involves non-integer polynomial exponents, we solve it numerically and approximate the solution analytically. Figure 5 shows that D^* decreases with increasing model size, with D^* equal to 4B tokens for an 8x1B model. When $D_2 \lesssim D^*$, the required D_{MoE} to catch up is more than 100% of D_1: upcycling remains more efficient than training from scratch. However, for $D_2 \gtrsim D^*$, the efficiency reverses, favoring training from scratch. Our findings indicate that: **Upcycling is more efficient than training an MoE from scratch for lower sunk cost and training token budget, but its efficiency diminishes as the sunk cost or model size increases.**

We consider other application and implication of the joint scaling law in Appendix G.

4 Discussion and Conclusion

In this paper, we have presented compelling evidence that MoE upcycling follows novel scaling laws with dataset and model configuration, revealing trade-offs due to interactions between dataset sizes.

While our empirical formulae successfully capture the observed scaling behavior, the underlying mechanism, particularly the interaction term, remains theoretically unexplained. To our knowledge, formal justification for such interactions is lacking in the literature [Bahri et al., 2024, Paquette et al.]. Future work could explore it through optimization dynamics or loss landscape analysis. Additionally, extending our scaling laws to alternative MoE architectures and encoder-decoder models would help assess their generality and implications.

References

J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Y. Bahri, E. Dyer, J. Kaplan, J. Lee, and U. Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27):e2311878121, 2024.

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

X. Bi, D. Chen, G. Chen, S. Chen, D. Dai, C. Deng, H. Ding, K. Dong, Q. Du, Z. Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. pages 2397–2430. PMLR, 2023.

Y. Bisk, R. Zellers, J. Gao, Y. Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439, 2020.

W. Cai, J. Jiang, F. Wang, J. Tang, S. Kim, and J. Huang. A survey on mixture of experts. *arXiv preprint arXiv:2407.06204*, 2024.

T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

A. Clark, D. de Las Casas, A. Guy, A. Mensch, M. Paganini, J. Hoffmann, B. Damoc, B. Hechtman, T. Cai, S. Borgeaud, et al. Unified scaling laws for routed language models. pages 4057–4086. PMLR, 2022.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

T. Computer. Redpajama: an open dataset for training large language models, October 2023. URL `https://github.com/togethercomputer/RedPajama-Data`.

D. Dai, C. Deng, C. Zhao, R. Xu, H. Gao, D. Chen, J. Li, W. Zeng, X. Yu, Y. Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

W. Du, T. Luo, Z. Qiu, Z. Huang, Y. Shen, R. Cheng, Y. Guo, and J. Fu. Stacking your transformers: A closer look at model growth for efficient llm pre-training. *arXiv preprint arXiv:2405.15319*, 2024.

D. Eigen, M. Ranzato, and I. Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

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. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

A. Hägele, E. Bakouch, A. Kosson, L. Von Werra, M. Jaggi, et al. Scaling laws and compute-optimal training beyond fixed training durations. *Advances in Neural Information Processing Systems*, 37: 76232–76264, 2024.

E. He, A. Khattar, R. Prenger, V. Korthikanti, Z. Yan, T. Liu, S. Fan, A. Aithal, M. Shoeybi, and B. Catanzaro. Upcycling large language models into mixture of experts. *arXiv preprint arXiv:2410.07524*, 2024.

T. Henighan, J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T. B. Brown, P. Dhariwal, S. Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

D. Hernandez, J. Kaplan, T. Henighan, and S. McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

J. Hestness, N. Ardalani, and G. Diamos. Beyond human-level accuracy: Computational challenges in deep learning. pages 1–14, 2019.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 30016–30030, 2022.

S. Hu, Y. Tu, X. Han, C. He, G. Cui, X. Long, Z. Zheng, Y. Fang, Y. Huang, W. Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

D. Kocetkov, R. Li, L. Jia, C. Mou, Y. Jernite, M. Mitchell, C. M. Ferrandis, S. Hughes, T. Wolf, D. Bahdanau, et al. The stack: 3 tb of permissively licensed source code. *Transactions on Machine Learning Research*.

A. Komatsuzaki, J. Puigcerver, J. Lee-Thorp, C. R. Ruiz, B. Mustafa, J. Ainslie, Y. Tay, M. Dehghani, and N. Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints.

V. A. Korthikanti, J. Casper, S. Lym, L. McAfee, M. Andersch, M. Shoeybi, and B. Catanzaro. Reducing activation recomputation in large transformer models. *Proceedings of Machine Learning and Systems*, 5:341–353, 2023.

J. Krajewski, J. Ludziejewski, K. Adamczewski, M. Pióro, M. Krutul, S. Antoniak, K. Ciebiera, K. Król, T. Odrzygóźdź, P. Sankowski, et al. Scaling laws for fine-grained mixture of experts. *arXiv preprint arXiv:2402.07871*, 2024.

T. Le Scao, T. Wang, D. Hesslow, S. Bekman, M. S. Bari, S. Biderman, H. Elsahar, N. Muennighoff, J. Phang, O. Press, C. Raffel, V. Sanh, S. Shen, L. Sutawika, J. Tae, Z. X. Yong, J. Launay, and I. Beltagy. What language model to train if you have one million GPU hours? pages 765–782, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.54. URL https://aclanthology.org/2022.findings-emnlp.54.

D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen. Gshard: Scaling giant models with conditional computation and automatic sharding.

J. Liu, L. Cui, H. Liu, D. Huang, Y. Wang, and Y. Zhang. Logiqa: a challenge dataset for machine reading comprehension with logical reasoning. pages 3622–3628, 2021.

L. Liu, Y. J. Kim, S. Wang, C. Liang, Y. Shen, H. Cheng, X. Liu, M. Tanaka, X. Wu, W. Hu, et al. Grin: Gradient-informed moe. *arXiv preprint arXiv:2409.12136*, 2024.

K. M. Lo, Z. Huang, Z. Qiu, Z. Wang, and J. Fu. A closer look into mixture-of-experts in large language models. *arXiv preprint arXiv:2406.18219*, 2024.

I. Loshchilov, F. Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5, 2017.

H. Mikami, K. Fukumizu, S. Murai, S. Suzuki, Y. Kikuchi, T. Suzuki, S.-i. Maeda, and K. Hayashi. A scaling law for syn2real transfer: How much is your pre-training effective? pages 477–492. Springer, 2022.

N. Muennighoff, A. Rush, B. Barak, T. Le Scao, N. Tazi, A. Piktus, S. Pyysalo, T. Wolf, and C. A. Raffel. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 36:50358–50376, 2023.

N. Muennighoff, L. Soldaini, D. Groeneveld, K. Lo, J. Morrison, S. Min, W. Shi, P. Walsh, O. Tafjord, N. Lambert, et al. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

D. Paperno, G. Kruszewski, A. Lazaridou, N.-Q. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, 2016.

E. Paquette, C. Paquette, L. Xiao, and J. Pennington. 4+ 3 phases of compute-optimal neural scaling laws. *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

T. Porian, M. Wortsman, J. Jitsev, L. Schmidt, and Y. Carmon. Resolving discrepancies in compute-optimal scaling of language models. *Advances in Neural Information Processing Systems*, 37: 100535–100570, 2024.

J. S. Rosenfeld, A. Rosenfeld, Y. Belinkov, and N. Shavit. A constructive prediction of the generalization error across scales.

K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

N. Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. 2017.

Z. Shen, T. Tao, L. Ma, W. Neiswanger, Z. Liu, H. Wang, B. Tan, J. Hestness, N. Vassilieva, D. Soboleva, et al. Slimpajama-dc: Understanding data combinations for llm training. *arXiv preprint arXiv:2309.10818*, 2023.

M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

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, June 2023. URL `https://huggingface.co/datasets/cerebras/SlimPajama-627B`.

J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

X. Sun, Y. Chen, Y. Huang, R. Xie, J. Zhu, K. Zhang, S. Li, Z. Yang, J. Han, X. Shu, et al. Hunyuan-large: An open-source moe model with 52 billion activated parameters by tencent. *arXiv preprint arXiv:2411.02265*, 2024.

H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

T. Wei, B. Zhu, L. Zhao, C. Cheng, B. Li, W. Lü, P. Cheng, J. Zhang, X. Zhang, L. Zeng, et al. Skywork-moe: A deep dive into training techniques for mixture-of-experts language models. *arXiv preprint arXiv:2406.06563*, 2024.

J. Welbl, N. F. Liu, and M. Gardner. Crowdsourcing multiple choice science questions. *W-NUT 2017*, page 94, 2017.

A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

B. Zhang, Z. Liu, C. Cherry, and O. Firat. When scaling meets llm finetuning: The effect of data, model and finetuning method.

P. Zhang, G. Zeng, T. Wang, and W. Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

A Preliminaries

A.1 Model details

Dense model. Our dense models are decoder-only transformers pretrained with an autoregressive language modeling objective. The architecture is most similar to Llama2 models [Touvron et al., 2023], incorporating advances such as SwiGLU [Shazeer, 2020] and rotary position embedding [Su et al., 2024]. We use the Llama tokenizer of vocabulary size 32,000.

Mixture-of-Experts. The MoE model in consideration is the same as our dense model, but with all MLP blocks replaced by multiple blocks (experts) with the same configuration [Fedus et al., 2022]. A router consisting of a single-layer MLP outputs the routing probability of the tokens to the experts.

The model configuration has two key parameters: n_{expert}, representing the number of experts, and n_{TopK}, which specifies how many of the highest-probability experts each token is routed to at each layer. The output of the experts is linearly combined and passed to the next layer.

The MoE and its corresponding dense model with model size N_{dense}, consisting of n_{expert} experts, is denoted with a prefix "n_{expert}", e.g., 8x1B where the dense correspondent is of 1B in model size. We refer to the number of non-embedding model parameters (that is, total number of parameters minus the number of embedding and language modeling head parameters) used for computation per token as the number of *active parameters*.

Upcycling. The upcycling scenario assumes that one is given a pretrained dense model and would like to train an MoE with the same configuration but replacing the MLPs with the MoE modules [Komatsuzaki et al.]. By replicating the dense model's MLP weight n_{expert} to form the experts, the knowledge from the dense model can be reused to accelerate the training of the MoE compared to training the MoE from scratch (from random parameter initialization). Other modules' weights are copied from the dense counterparts directly, with the router's initial weights randomized. See Figure 2 for an illustration of upcycling. We employ this technique for our study.

A.2 Power-law Ansatz

There is an extensive literature showing that the loss of training deep learning models has a simple power-law behavior: $L = \frac{A}{X^\alpha} + E$, for single variable X, including model size and dataset size [Hestness et al., 2017, 2019, Rosenfeld et al., Henighan et al., 2020]. [1] We use this ansatz in this work.

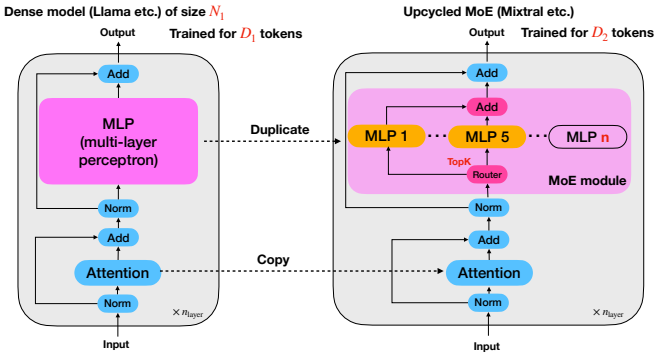

Figure 2: **Upcycling and factors affecting MoE's performance.** Upcycling involves initiating the weights of the MoE (activating n_{TopK} experts per token) by reusing the weights (duplicating the weights of MLPs n_{expert} times) of an existing dense transformer of size N_1 that has been trained for D_1 tokens. The (upcycled) MoE is further trained for D_2 tokens. Language modeling performance improves when scaling these factors. We study and develop formulae (scaling laws) consisting of these factors to predict the empirical performance.

[1] The actual form we assume is $L = \frac{A}{(X+1)^\alpha} + E$ such that the loss is finite at the limit $X \to 0$. However, since the values of X we consider are often much larger than 1 (10^6 or more), we approximate it as $L = \frac{A}{X^\alpha} + E$ for notational convenience.

Hoffmann et al. [2022] have shown that when training a dense transformer, the cross-entropy loss is well-described by the following "Chinchilla" scaling law:

$$L = AD^{-\alpha} + BN^{-\beta} + E \tag{7}$$

The first and second terms quantify the limitation of learning due to limited dataset and model sizes respectively. The scaling exponents, α, β control how fast the loss decreases with respect to dataset and model sizes respectively. E is a constant: it is the irreducible loss representing the inherent entropy of text. All parameters are to be fitted with experimental observations. We also assume Equation 7 for models trained from scratch.

B Related Work

Mixture-of experts. While interests in developing open MoE LLMs are relatively recent [Hu et al., 2024, Yang et al., 2024, Dai et al., 2024, Liu et al., 2024, Sun et al., 2024], the use of MoE in deep learning can be traced to early 2010s [Eigen et al., 2013, Bengio et al., 2013]. See Cai et al. [2024] for a detailed survey of modern MoE models.

Upcycling. Leveraging pretrained models to expedite the training of larger dense models is also known as model growth [Chen et al., 2015]. In the context of training MoEs reusing dense pretrained models, Komatsuzaki et al. are the first studying such a scenario with encode-decoder models. There are studies [Hu et al., 2024, Yang et al., 2024, Lo et al., 2024, Wei et al., 2024, Muennighoff et al., 2024, He et al., 2024] offering insights into upcycling decoder-only transformers, systematic investigation has not been presented. Wei et al. [2024] made only a rough recommendation: use upcycling when the budget for training is smaller than twice the budget used for dense training, while we provide a more general guideline.

Scaling laws. Power-law scaling appears in a variety of natural and human-made phenomena. Scaling studies that are perhaps closest to our work are those considering two-stage training, e.g., transfer learning, fine-tuning, and model growth [Mikami et al., 2022, Zhang et al., Du et al., 2024]. A notable similar phenomenon in transfer learning is *ossification*, where pretraining can hurt fine-tuning performance [Hernandez et al., 2021]. However, to our knowledge, no prior work incorporates an interaction term (α_3 in Equation 1) to capture such effects.

In the context of MoE, Clark et al. [2022] studied how different architectural choices affect MoE's scaling, while Krajewski et al. [2024] investigated fine-grained experts' scaling behavior. The latter work fits a joint scaling law with respect to dataset and model sizes, but we find differences in the obtained parameters. It is likely due to several differences in methodology: the largest dense model they experimented with was smaller (85M), and they did not tune the LR for each model size. Nevertheless, our findings that MoE scales better than its dense counterpart with sufficiently large computational budget do agree with theirs.

C More on architecture and experimental design

	A	B	α/α_1	α_2	α_3	β
Dense	8.83	12.3	0.088	–	–	0.116
MoE	32.0	7.05	0.161	–	–	0.080
Upcycled	16.3	8.53	0.043	0.085	7.98e-4	0.112

Table 2: **Fitted parameters for joint scaling laws.** Note that for MoE, we fit the parameters with variable N_1 (instead of N_2) for comparison conveniences. The irreducible loss applicable to all laws, E, is fitted to be 0.165.

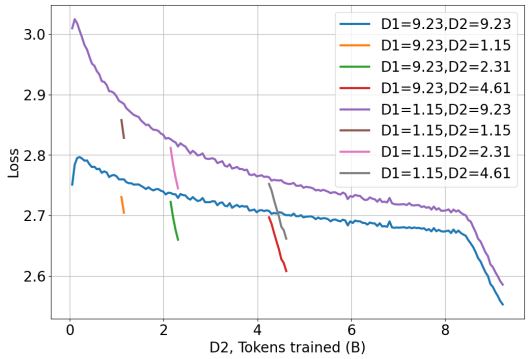

Figure 3: **Loss curves of upcycling.** Intermediate test losses of the 8x0.1B MoE (2 experts activated per token) trained for a variety of total number of tokens, D_2, when upcycled from a dense model pretrained with various numbers of training tokens (D_1) in B.

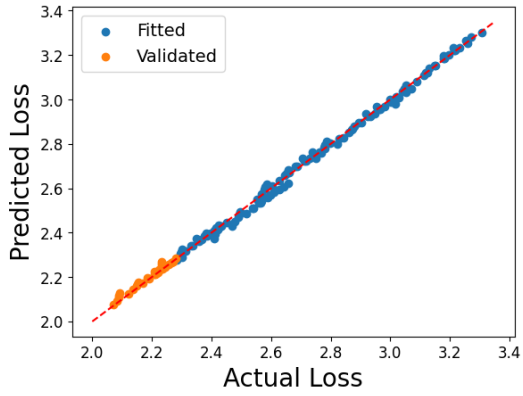

Figure 4: **Fits of the joint upcycling scaling law.**

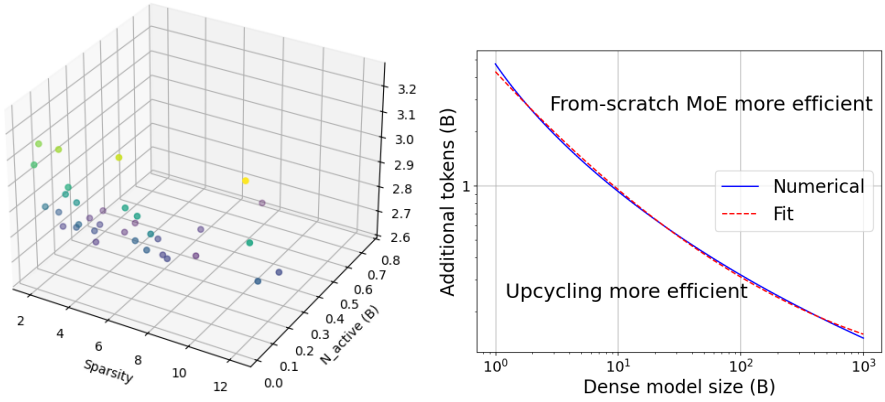

Figure 5: **Left: Upcycling improves with sparsity and the number of active parameters.** We find that upcycling to MoE which is sparser and has more active parameters improves performance. The z-axis shows the value of cross-entropy loss. See Section **??** for details. **Right: Efficiency of upcycling diminishes with sunk cost and model size.** For all additional token budgets for upcycling a Mixtral-like MoE above the curve(s), training MoE from scratch is more efficient, whereas for all token budgets below it, upcycling is more efficient. Shown are the numerical (blue) and analytical (red) solutions of Equation 11. See Section G.1 for details.

C.1 Megatron-LM configuration

Infrastructure. Our experiments are performed on multiple nodes, each consisting of 8 NVIDIA H100 80 GB GPUs, interconnected via InfiniBand HDR. The software we use for training is the Megatron-LM library [Shoeybi et al., 2019].

We use and modify the Megatron-LM (core v0.8.0) library for our experiments. [2] Models are trained with data type bfloat16. Except for the largest MoE we train (8x1B), which has tensor parallelism configured to be 2, all models are trained with data and sequence parallelisms only [Korthikanti et al., 2023]. Other optimization libraries used include FlashAttention [Dao et al., 2022] and TransformerEngine. [3] See the example scripts included in the supplementary material.

C.2 Model configuration

Let us elaborate more on our architectural choices. The intermediate hidden dimension size, d_{MLP}, is set to be $4d_{\text{model}}$. We do not implement bias in the linear layers. We also do not use techniques geared for treating training instabilities (which we did not encounter in our study), such as Z-loss or QK normalization. Efficiency-motivated implementations like grouped query attention are not considered as well for simplicity. The number of attention head is chosen to increase with model size following practices in the LLM literature. Other designs of the architecture follow Llama2's closely [Touvron et al., 2023]. See Table 3 for the model configurations. They are selected such that the ratio $n_{\text{layer}}/d_{\text{model}}$ lies in the range 32 to 64, as in Kaplan et al. [2020]. We use the smaller models for ablation studies.

C.3 MoE configuration

Let us describe the routing mechanism within the MoE module studied in this work. Denote the number of experts by n_{expert}, the number of activated experts by n_{TopK}, and the output of expert i by $O_{\text{exp},i}$. At each layer, the output tokens x of the attention layer are passed to a router, which consists of a single-layer perceptron with weight W, responsible for calculating $(G_1(x), G_2(x), ..., G_{n_{\text{expert}}}(x))$, where

$$G(x) = \text{Softmax}(\text{TopK}(W \cdot x)) \tag{8}$$

The TopK operation ensures that only n_{expert} experts are activated for each token, resulting in sparse computation. The output of the MoE module, O_{MoE}, is the weighted expert outputs which can be written as follows:

$$O_{\text{MoE}} = \sum_{i=1}^{n_{\text{expert}}} G_i(x) O_{\text{exp},i}(x)$$

Note that this is also known as token-choice algorithm. Furthermore, we do not use the token-dropping mechanism as in Fedus et al. [2022]. We also do not study MoE variants such as shared experts [Dai et al., 2024] and fine-grained experts [Krajewski et al., 2024], as upcycling these variants is not straightforward.

Let us move to discussing the load-balancing loss. It has the form [Fedus et al., 2022]:

$$\mathcal{L}_{\text{aux}} = \frac{4\eta}{T^2} \cdot \sum_{i=1}^{n_{\text{experts}}} \left(\left(\sum_{j=1}^{T} P_{j,i} \right) \cdot Q_i \right),$$

where η is the coefficient for the auxiliary loss, T is the number of tokens, $P_{j,i}$ the router output probability for token j to be assigned to token i, and Q_i is the number of token assigned to expert i. We ablate the coefficient in Appendix C.7.

C.4 Ablation of learning rate schedules

Here, we compare the performances of using WSD and the commonly used learning rate (LR) cosine schedules. Dense model and MoE used in our ablation are 0.1B and 8x44m respectively, with training configuration given in Table 6. We can see from Figure 6 that both schedules yield similar performances.

[2] `https://github.com/NVIDIA/Megatron-LM`

[3] `https://github.com/NVIDIA/TransformerEngine`

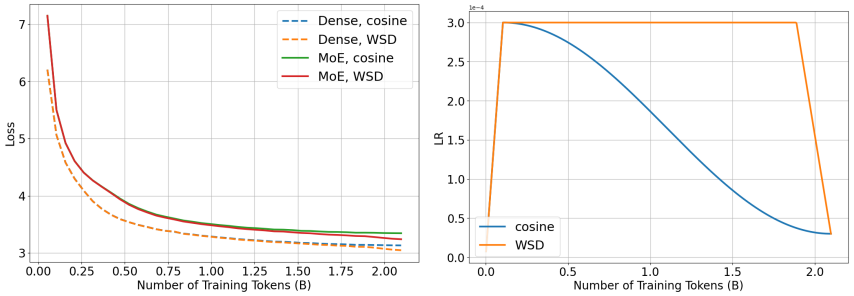

Figure 6: **Comparing WSD and cosine schedules of learning rate.** Left: we see that different schedules cause little differences between the losses, for both dense and MoE training. Right: the learning rate schedules in use are shown.

C.5 Training configuration

The common setup of training is shown in Table 4, and the model-dependent setup (warmup iteration, standard deviation of the normal distribution for initializing weights, maximum iteration run, batch size, tuned LR) is shown in Table 5. As described in the main text, we use the WSD schedule for training. The number of warmup steps of the WSD LR schedule is set to be roughly the same as the total model size [Porian et al., 2024]. Linear decay to 10% of the maximum LR value is used in the last stage of the schedule, with the length set to be around 10% of the training length, following Hägele et al. [2024].

Logarithmically-spaced intermediate checkpoints are saved and used to emulate different numbers of token budget. We also increase both the training length and batch size with model size following common practices without performing precise tuning.

Model	n_{layer}	d_{model}	n_{head}	N_{dense}	$N_{\text{MoE}}^{\text{total}}$ (8 experts)
15M	9	320	4	14,751,680	92,189,120
44M	12	480	8	44,248,800	276,538,080
0.1B	15	640	8	98,323,840	614,496,640
0.2B	21	832	8	232,623,040	1,453,845,952
0.5B	26	1120	16	521,889,760	3,261,732,320
1B	30	1504	16	1,085,859,424	6,786,500,704

Table 3: Dense models used in our study and their parametric details. Note that d_{MLP}, is set to be $4d_{\text{model}}$. In the last column, we show the total non-embedding model parameters of the corresponding MoE with 8 experts.

Configuration	Details
Context length	1,024
Embedding tying	False
Optimizer	AdamW [Loshchilov et al., 2017]
Adam β_1	0.9
Adam β_2	0.95
Adam ϵ	1e-8
Weight decay	0.1
Gradient clipping	1.0

Table 4: Training configuration used throughout the paper.

Model	warmup iter.	init. size	Max iter.	batch size	LR (8x)
15M	200	0.035	17,600	128	8e-3 (2e-3)
44M	200	0.029	17,600	256	4e-3 (2e-3)
0.1B	200	0.025	17,600	512	4e-3 (2e-3)
0.2B	400	0.022	35,200	512	2e-3 (2e-3)
0.5B	800	0.019	70,400	512	4e-4 (4e-4)
1B	800	0.016	70,400	1024	4e-4 (4e-4)

Table 5: **Model-dependent training configuration**. "init. size" refers to the standard deviation of the normal distribution used for initializing the weights. "Max iter." refers to the maximum iteration run on the model. The MoE counterpart uses the same configuration except for the learning rate (last column).

Configuration	Details
Batch size	512
train iter.	4,000
Warmup iter.	200
Auxiliary loss coeff.	10^{-3}

Table 6: Training configuration for ablation studies.

C.6 Upcycled training's configuration

We initialize the router weights from a normal distribution with zero mean and variance of $2/5d_{\text{model}}$ [Le Scao et al., 2022, Chowdhery et al., 2023, Zeng et al., 2022] (the same initialization is used for from-scratch trainings).

Regarding the learning rate (LR) of upcycled training, there are several choices: using the LR at the end of dense training with a constant LR schedule, i.e., treating upcycled training as a kind of fine tuning; using the LR of dense/MoE training [Komatsuzaki et al., He et al., 2024]. We consider these three choices without retuning the LR. With the other training settings are set to be the same as the one used in training the dense models, including the use of the WSD LR schedule, we find that using the MoE LR leads to better performance. See Figure 7.

As a side note, we observe that the loss for the constant LR schedule decreases monotonically, while the loss increases initially for other cases; there is a *rewarming* stage when using the WSD LR schedules, which can also be observed in Figure 3.

Let us further comment on alternative upcycling methods that could potentially further accelerate training. Adding some form of noises to the dense MLP weights, or modifying partially the weights would intuitively help upcycled training generalize faster. Our preliminary experiments however did not see any advantages of doing so. Note that this observation is also consistent with previous negative reports [Komatsuzaki et al., Wei et al., 2024, Muennighoff et al., 2024]. Henceforth, we simply copy the weights directly from the dense model to perform upcycled training.

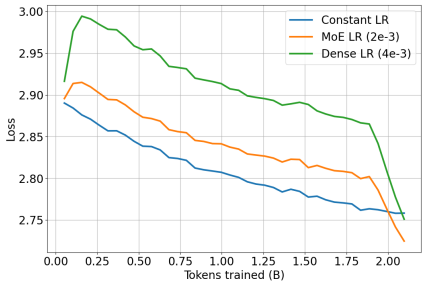

Figure 7: **Ablation of LR when upcycling an 8x0.1B MoE.** We compare the performance of upcycled training (from a dense model trained for 2B tokens) using constant LR (2×10^{-4}, LR at the end of dense training), LR used for dense training, and LR used for MoE training. We find that the latter works the best. The training setup follows Table 6.

C.7 Ablation of auxiliary coefficients for load-balancing loss

The auxiliary loss affects the load-balancing loss (smaller values meaning more balanced expert usage) as well as the cross-entropy loss. We plot these losses in Figure 8 for training a 8x0.1B MoE.

We see that, setting the coefficient to be too small leads to imbalance in expert usage, while setting the coefficient to be too large interferes with the cross-entropy loss. We set it to around 10^{-3} which gives the right balance. We did not make finer tuning of the coefficients and adopted 10^{-3} in our experiments.

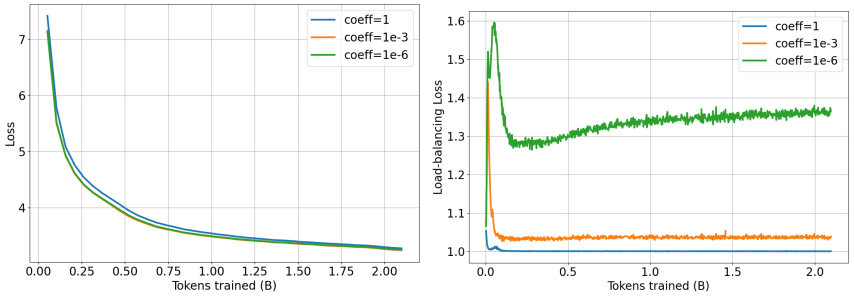

Figure 8: **Ablating auxiliary coefficients.** Left: Cross-entropy losses, where it can be seen that auxiliary coefficient of 1 performs worst. Right: Load-balancing losses, where the larger the coefficient is, the smaller the load-balancing loss becomes. Setting the coefficient to be 10^{-3} gives the right balance between these two losses.

C.8 Ablation of dataset repetition for upcycling

In our experiments, we have used the same dataset for training both dense and upcycled models, following practices of [He et al., 2024]. As an ablation, we split our dataset to two non-overlapping portions, and perform the two-stage training with the distinct datasets. In Figure 9, we find that the difference in performance is very small. This also aligns with previous investigation on data repetition, where it is shown that there is a little difference in performance up to 4 times of repetition [Muennighoff et al., 2023].

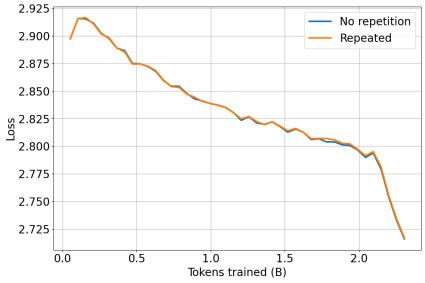

Figure 9: **Ablation of data repetition when upcycling an 8x0.1B MoE.** We do not observe notable difference in the loss. The training setup follows Table 6.

C.9 Evaluation with standard benchmarks and comparison with other existing models

We compare the performance of our trained 1B models against existing models with similar sizes, Pythia [Biderman et al., 2023] and TinyLlama [Zhang et al., 2024], based on standard natural language processing benchmarks, ARC [Clark et al., 2018], lambada [Paperno et al., 2016], logiqa [Liu et al., 2021], piqa [Bisk et al., 2020], sciq [Welbl et al., 2017], and winogrande [Sakaguchi et al., 2021].

Table 7 shows the results. First, we see that the dense model perform similarly to the open models, indicating that our models have been trained correctly. Second, we see that the upcycled model achieves overall the best performance (note that the total tokens used for dense and upcycled training

are around 100B, similar to those under comparison), also indicating that upcycling has progressed correctly and improved the scores.

Models	**Pythia-1B**	**TinyLlama-1.1B**	**1B**	**Upcycled 8x1B**
Datasets	Pile	Slimpajama & Starcoder	Slimpajama	Slimpajama
Tokens	100B	103B	74B	37B
ARC-c	25.59	24.32	27.65	**30.12**
ARC-e	47.26	44.91	52.10	**56.14**
lambada	**53.52**	-	45.08	49.72
logiqa	**29.49**	-	26.11	27.65
piqa	**69.31**	67.30	65.89	67.19
sciq	77.3	-	78.10	**82.00**
winogrande	51.22	53.28	54.93	**57.77**
Avg.	50.53	-	49.98	**52.94**

Table 7: **Benchmarks' performance comparison across models.** Reported scores are accuracies (normalized by byte length whenever applicable). The first two columns are scores of existing models. The last two columns are evaluation results of models trained in this work. The upcycled 8x1B model is upcycled from the 1B model. Our models are evaluated with the LM Evaluation Harness v0.4.0 library [Gao et al., 2024].

C.10 Estimated Total GPU hours

Instead of reporting the actual runtimes on our cluster, which varied in our experiments due to many factors affecting the cluster (number of available nodes, congestion, etc.), we give a theoretical estimate of total GPU hours used for obtaining the joint scaling law, which involves running the largest tested model with most training tokens in this paper.

The estimate is as follows. We calculate the FLOPs for training the dense, MoE, upcycled MoE models with maximum iterations using the $6ND$ approximation, ignoring the additional FLOPs required to continued pretrain models with shorter iterations (as we can reuse the intermediate checkpoints). We further assume that the per-second TFLOPs of the GPU is 400, and is the same for both dense and MoE models. [4] We obtain,

Dense model: 6.14×10^{17} FLOPs

MoE: 1.08×10^{18} FLOPs

Upcycled MoE: 5.38×10^{18} FLOPs

The total GPU hours are henceforth approximately 4,900. We note that the exact total GPU hours used are larger as we have run various additional studies as detailed in the paper. We further note that using the cosine learning rate schedule would cost about twice more GPU hours when varying the number of training token budget.

D Generalization across other datasets

We show that the scaling behavior studied in the main text generalizes across datasets in use. Aside from Slimpajama [Computer, 2023, Soboleva et al., 2023, Shen et al., 2023] used in the main text, we test with two additional datasets: Japanese portion of the CommonCrawl corpus, and the Stack code dataset [Kocetkov et al.]. We use a tokenizer of enlarged vocabulary size of 102,400 when training with the former dataset. [5] We upcycle a 0.1B model with the training configuration set to the same as those used for the experiments presented in the main text. The results are presented in Figure 10, where the scaling behaves similarly to those presented in Figure 1, suggesting that our scaling law for

[4] `https://github.com/NVIDIA/Megatron-LM/blob/main/megatron/core/transformer/moe/README.md`. Note that MoE requires more GPU memory to store its total model parameters, inducing overhead that may slow down training.

[5] `https://huggingface.co/sbintuitions/sarashina2-70b`

dataset are independent of dataset in use. We however do not further study scaling law for model configuration and the joint scaling law due to computational constraints.

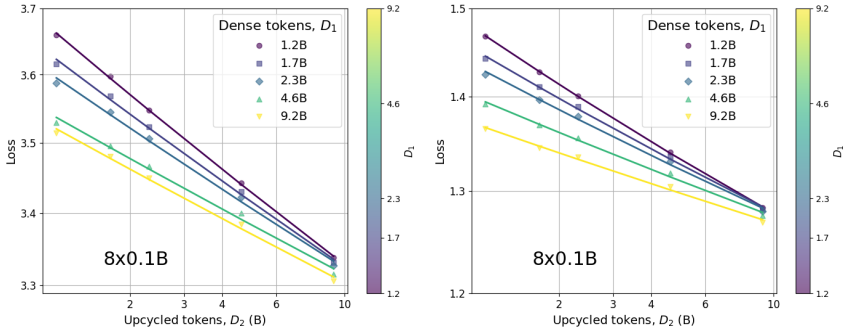

Figure 10: **Scaling behavior generalizes across datasets used.** Left: Japanese dataset. Right: Code dataset.

E More results on scaling law for dataset sizes

E.1 Fitting across architectures

Figure 8 shows the fitting of the scaling law for dataset sizes across architectures (fixing dense model size to 0.1B).

$(n_{\text{expert}},n_{\text{TopK}})$	(4,1)	(4,2)	(8,1)	(8,2)
α_1	0.28	0.19	0.29	0.18
α_2	0.29	0.20	0.30	0.20
α_3	0.01	0.008	0.01	0.008

Table 8: **Fitted exponents of the scaling law for dataset sizes across architectures.**

E.2 More validation of Equation 1

We first make an observation from Figure 1 that the slope of the fitted lines, that is, the scaling exponent as in Equation 2 is decreasing with D_1. Let us consider the scaling factors A fitted in Figure 1. The scatter plot of A's in Figure 11 leads us to deduce that the following relation:

$$A \propto D_1^{-\eta} \tag{9}$$

Indeed, we find that the above equation fits well in the Figure. Substituting it to Equation 2, we obtain at the RHS a term of the form $D_1^{-\eta} D_2^{-\alpha}$, indicating that Equation 1 arises not only from first principles as in the main text, but from empirical observations as well.

E.3 Bilinear form

We would also like to note that by taking the logarithm of the first term of Equation 3, we have $\alpha_1 \log D_1 + \alpha_2 \log D_2 + \alpha_3 \log D_1 \log D_2$, which is *bilinear* in terms of $\log D_1$ and $\log D_2$ (that is, linear with each variable separately), indicating that the multiplicative term with interaction is a natural generalization of the single-variable power law.

F More Results on the Joint Scaling Law

F.1 Validating Requirement 3

We show in Figure 12 that the upcycled model empirically satisfies power-law scaling with respect to N_1, i.e., Requirement 3, repeated below for convenience.

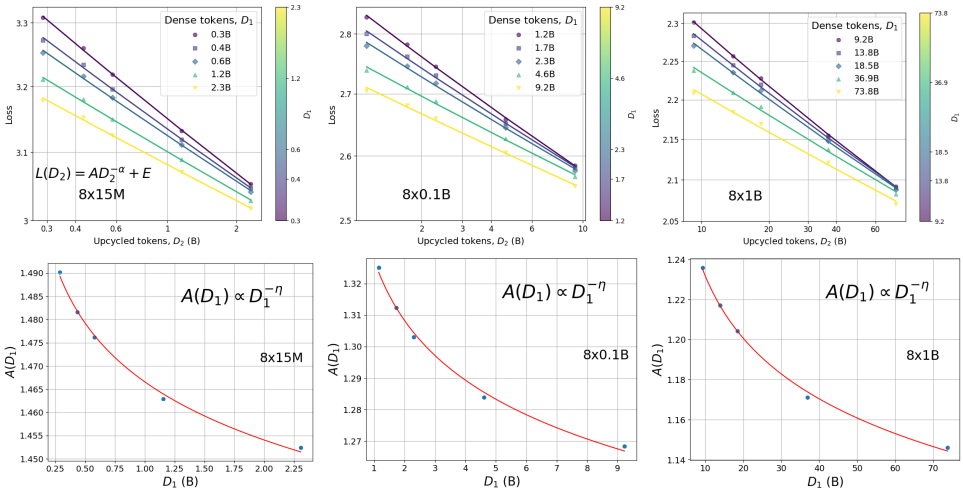

Figure 11: **Top: Fitting plots same as those in Figure 1. Bottom: Fitting scaling factor has power-law behavior.** The fitted scaling factors are shown to fit well with Equation 9.

Requirement 3. $L_{D_1,D_2}(N_1) = B_{D_1,D_2} N_1^{-\beta} + E_{D_1,D_2}$.

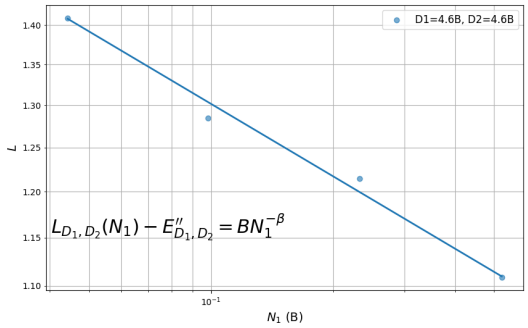

Figure 12: **Upcycled model has power-law behavior with respect to N_1.**

F.2 Another functional form satisfying Requirements 3 and 4

For convenience purposes, let us repeat Requirement 4:

Requirement 4. $\lim_{D_2 \to 0} L(D_1, D_2, N_1) = AD_1^{-\alpha} + BN_1^{-\beta} + E$, i.e., approaching Chinchilla scaling law.

There is another functional form satisfying Requirement 3 and 4:

$$L(D_1, D_2, N_2) = \frac{A}{D_1^{\alpha_1}} + \frac{B}{D_2^{\alpha_2} N_1^{\beta_2 - \alpha_3 \log D_2}} + E \tag{10}$$

Fitting the above equation however yields negative exponents, which are, empirically, $\alpha_1 = 0.10, \alpha_2 = -0.07, \beta = -0.08, \alpha_3 = -0.008$. We reject the hypothesis of this functional form as the loss is predicted to increase with dataset/model sizes, violating the power-law ansatz as well as empirical observation.

F.3 Why fitting scaling laws separately?

We provide reasoning on why we fit three scaling laws separately, instead of fitting an unified scaling law which holds for all three dense, MoE from-scratch, and upcycled training scenarios.

The scaling law for MoE trained from scratch is expected to have parameters different from those for upcycled training (that is, they are not equal to each other at $D_1 \to 0$), because upcycled MoEs are initialized with identical MLP/expert weights from the dense models, while MoEs trained from-scratch have different initial (random) MLP/expert weights. Thus, we expect them to scale differently.

Similarly, we expect the scaling law for dense model has parameters different from those for upcycled training at the limit $D_2 \to 0$. As can be seen from Figure 3, the upcycled training undergoes a *rewarming* phase in the beginning, where the loss increases initially before decreasing. This causes a deviation from the dense model scaling law at $D_2 \to 0$, although we still expect it to correlate with the dense model scaling law, i.e., Requirement 2 should hold based on function preservation and (empirical) observations that rewarming does not completely de-correlate the loss behaviors.

Finally, our preliminary investigations also show that unified scaling law does not fit well. We consequently consider them separately. Moreover, we note that there can be other possibilities of functional forms that we do not explore further.

G More implication of the Joint Scaling Law

G.1 Training MoE from Scratch versus Upcycling

Past studies [Komatsuzaki et al., Muennighoff et al., 2024] have explored the efficiency of training an MoE from scratch compared to upcycling a dense model. Komatsuzaki et al. found that upcycling retains an advantage up to 120% of the sunk cost (D_1). For instance, to match the performance of an upcycled MoE trained with an additional 0.4 trillion (T) tokens after 2T dense tokens, training an MoE from scratch would require 2.4T tokens—effectively saving 2T tokens in upcycled training. Conversely, Muennighoff et al. [2024] reported that training from scratch requires less than 100% of the sunk cost under different settings, indicating that it could be more efficient. These seemingly contradictory results suggest that upcycling efficiency depends on both sunk cost and model size.

To investigate this, we define D^* the number of tokens required for training from scratch to match the performance of an upcycled MoE with the same sunk cost, i.e.,

$$L_{N_1}^{\mathrm{MoE}}(D^*) = L_{N_1}^{\mathrm{Upc.}}(D_1 = D^*, D_2 = D^*) \tag{11}$$

Since the above equation involves non-integer polynomial exponents, we solve it numerically and approximate the solution analytically (Equation **??**). Figure 5 shows that D^* decreases with increasing model size, with D^* equal to 4B tokens for an 8x1B model. When $D_2 \lesssim D^*$, the required D_{MoE} to catch up is more than 100% of D_1: upcycling remains more efficient than training from scratch. However, for $D_2 \gtrsim D^*$, the efficiency reverses, favoring training from scratch. Our findings indicate that: **Upcycling is more efficient than training an MoE from scratch for lower sunk cost and training token budget, but its efficiency diminishes as the sunk cost or model size increases.**

G.2 Compute Allocation and Compute-Optimal Upcycling

Given a fixed total floating-point operation (FLOPs) budget C, we analyze how to optimally allocate compute between dense training and upcycled training. Specifically, we solve:

$$\min_{D_1, D_2, N_1} L(D_1, D_2, N_1) \text{ s.t. } \mathrm{FLOPs}(D_1, D_2, N_1) = C$$

We find that for a given compute budget without any pretrained models, training a compute-optimal dense or MoE model from scratch outperforms the two-stage dense-to-upcycled MoE approach (see Figure 14 in Appendix G.4). This suggests: **If no pretrained model exists, direct training is preferable to upcycling for optimal performance.**

In scenarios where a pretrained dense model is already available, we can determine the compute-optimal scaling of MoE upcycling. The compute cost of upcycled training is approximated as $C_2 = 6N_2D_2$ [Kaplan et al., 2020]. Optimizing the compute leads to the scaling relations (see Appendix G.4 for derivation):

$$D_2^{\mathrm{opt}} \propto C_2^{\frac{\beta}{\beta+\alpha_{\mathrm{eff}}}}, \quad N_1^{\mathrm{opt}} \propto C_2^{\frac{\alpha_{\mathrm{eff}}}{\beta+\alpha_{\mathrm{eff}}}} \tag{12}$$

where $\alpha_{\text{eff}} := \alpha_2 - \alpha_3 \log D_1$. Notably, as D_1 increases, α_{eff} decreases, meaning **larger pretrained models require disproportionately more tokens for efficient upcycling.**

For instance, applying this to the Llama2 models (7B, 13B, 70B), which were trained on 2T tokens [Touvron et al., 2023], we estimate that compute-optimal upcycling follows the scaling $D_2 \propto N_1^{1.8}$, indicating that larger models require nearly quadratic increases in upcycling data.

Overall, we find that upcycling is inefficient relative to from-scratch trainings when considering compute optimality.

G.3 From-scratch training vs Upcycling

We have shown in the main text that upcycling is only effective when the sunk cost or the model size is small. We further visualize this by plotting the losses for model sizes 1B, 7B and 70B, fixing D_1 to various values, with respect to D_{MoE} and D_2 for from-scratch and upcycled training respectively in Figure 13.

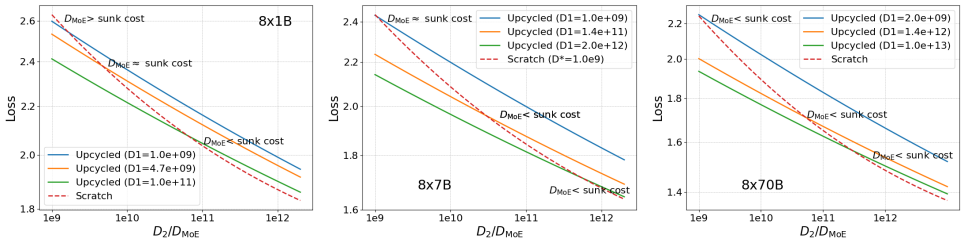

Figure 13: **Token budget of from-scratch MoE training and when it catches up with upcycled MoE's performance.** We compare loss-versus-token plots of from-scratch and upcycled MoE trainings at various dense training budgets (sunk costs) and model sizes. Upcycling is considered to be efficient only when $D_{\text{MoE}} >$ sunk cost. We observe that the efficiency of upcycling diminishes with sunk cost and model size.

G.4 More on compute optimality of upcycling

As mentioned in the main text, while training the Mixtral-like MoE from scratch is shown to be more efficient in the long run, it requires more compute (1.75 times larger than dense training). There is a compute-performance trade-off between these stages of training.

We show in Figure 14 that the optimization results in Section G.2 that training from scratch can always be considered to be more performant.

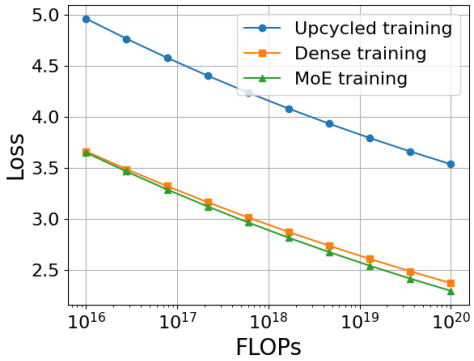

Figure 14: **Compute-optimal training.** Upcycled training performs worse even when D_1, D_2, N_1 are allocated optimally, compared to compute-optimal training of dense or MoE models.

G.5 Derivation of compute-optimal formula

As in the main text, we want to scale N_2, D_2 optimally, $N_2^{\text{opt}}, D_2^{\text{opt}}$, given a FLOPs budget and fixing D_1, while minimizing the loss L, which we write as $L_{D_1}(D_2, N_1)$. This is equivalent to solving the following:

$$\left.\frac{\partial}{\partial D_2} L_{D_1}(D_2, C_2/10.5D_2)\right|_{D_2=D_2^{\text{opt}}} = 0,$$
$$\left.\frac{\partial}{\partial N_1} L_{D_1}(C_2/10.5N_1, N_1)\right|_{N_1=N_1^{\text{opt}}} = 0$$

where we have used $N_2 = 1.75N_1$ and $C_2 = 6N_2D_2$. Solving the above equations leads to

$$D_2^{\text{opt}} = G\left(\frac{C_2}{10.5}\right)^a,$$
$$N_1^{\text{opt}} = G^{-1}\left(\frac{C_2}{10.5}\right)^b$$

where

$$G \coloneqq \left(\frac{A_{\text{eff}}\alpha_{\text{eff}}}{B\beta}\right)^{1/(\alpha_{\text{eff}}+\beta)}$$
$$a \coloneqq \frac{\beta}{\alpha_{\text{eff}}+\beta}$$
$$b \coloneqq \frac{\alpha_{\text{eff}}}{\alpha_{\text{eff}}+\beta}$$
$$A_{\text{eff}} \coloneqq AD_1^{-\alpha_1}$$
$$\alpha_{\text{eff}} \coloneqq \alpha_2 - \alpha_3 \log D_1$$

We can henceforth relate D_2^{opt} and N_1^{opt} via

$$D_2^{\text{opt}} = G\left(GN_1^{\text{opt}}\right)^{a/b} \propto \left(N_1^{\text{opt}}\right)^{\beta/\alpha_2 - \alpha_3 \log D_1}$$

and

$$N_1^{\text{opt}} = G^{-1}\left(G^{-1}D_2^{\text{opt}}\right)^{b/a} \propto \left(D_2^{\text{opt}}\right)^{(\alpha_2 - \alpha_3 \log D_1)/\beta}$$