SCALING LAWS FOR FINE-GRAINED MIXTURE OF EXPERTS

Jan Ludziejewski* University of Warsaw IDEAS NCBR
Jakub Krajewski* University of Warsaw IDEAS NCBR
Kamil Adamczewski IDEAS NCBR
Maciej Pióro IPPT PAN IDEAS NCBR
Michał Krutul University of Warsaw IDEAS NCBR
Szymon Antoniak IDEAS NCBR
Kamil Ciebiera University of Warsaw IDEAS NCBR
Krystian Król University of Warsaw IDEAS NCBR
Tomasz Odrzygóźdź TradeLink
Piotr Sankowski University of Warsaw IDEAS NCBR
Marek Cygan University of Warsaw Nomagic
Sebastian Jaszczur* University of Warsaw IDEAS NCBR

ABSTRACT

Mixture of Experts (MoE) models have emerged as a primary solution for reducing the computational cost of Large Language Models. In this work, we analyze their scaling properties, highlighting certain arbitrary assumptions present in the existing literature. In particular, we introduce a new hyperparameter, granularity, which allows for the optimal adjustment of the size of experts. Subsequently, we present scaling laws for fine-grained MoE, taking into account the number of training tokens, model size, and granularity. Using these scaling laws, we derive the optimal training configuration for a given computational budget. Furthermore, in contrast with previous works, we demonstrate that the gap in efficiency between dense and MoE models grows as we scale up the model size and training budget.

1 INTRODUCTION

In recent years, LLMs achieved exceptional performance in tasks across numerous domains (Chowdhery et al., 2022; Yin et al., 2023; Agostinelli et al., 2023). However, training those massive models incurs high computational costs, measured in millions of GPU-hours (Touvron et al., 2023; Workshop et al., 2023), and leading to non-negligible carbon footprints (Faiz et al., 2024). To combat these obstacles, the research community has been striving to increase the efficiency of LLMs. One promising approach that has lately been gaining visibility is the use of Mixture of Experts (MoE) methods. Models such as Switch (Fedus et al., 2022) and Mixtral (Jiang et al., 2024) have demonstrated that it is possible to achieve comparable effectiveness with significantly lower computational costs.

In the context of the current trend of increasing budgets for training models, a question arises: will MoE models continue to be attractive in the future? This is an important issue, as results from other studies (Clark et al., 2022) suggest that the traditional dense models may outperform MoE as the size of the models increases. We argue that previous claims lose their validity when we relax certain implicit assumptions regarding the training process, present in previous research (Clark et al., 2022). In particular, we refer to the fixed training duration and the constant size of experts in MoE models.

*Equal contribution. Corespondence to Jan Ludziejewski <ludziej@mimuw.edu.pl> , Jakub Krajewski <gim.jakubk@gmail.com> and Sebastian Jaszczur <s.jaszczur@uw.edu.pl>.

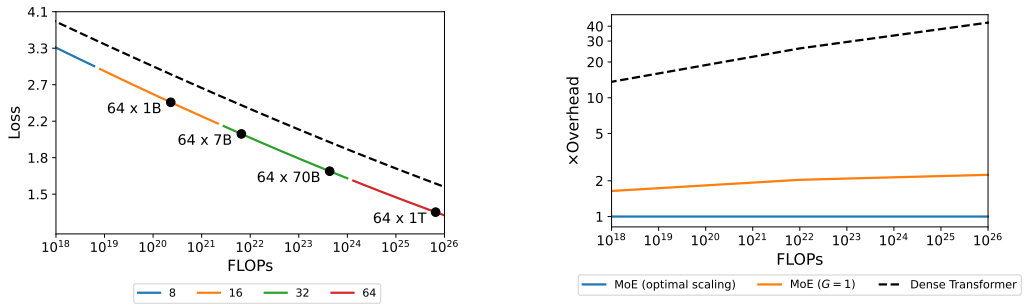


Figure 1: Mixture-of-Experts can be *always* considered more efficient than dense Transformers, regardless of the model size. **(a)** Compute Optimal scaling curves for standard Transformers (the dashed line) and for fine-grained MoE (colors denote optimal granularity for the given FLOPs training budget). **(b)** Relative number of FLOPs needed to train Transformer and Vanilla MoE (MoE with $G = 1$) to achieve the performance of MoE with compute optimal G.

Our results suggest that a compute-optimal MoE model trained with a budget of 10^{20} FLOPs will achieve the same quality as a dense Transformer at a $20\times$ greater computing budget, with compute savings rising steadily, exceeding $40\times$ at 10^{25} FLOPs (see Fig. 1). Our main contributions are:

1. Introducing a new hyperparameter - granularity. Adjusting this parameter allows us to determine the optimal size of experts in MoE models, translating into increased efficiency.
2. Deriving new scaling laws for MoE models by integrating variable training duration, the number of parameters, and granularity, pointing us to the optimal parameters of MoE models.
3. Demonstrating that, with optimal settings, MoE models can always outperform vanilla Transformers at any computing budget - contrary to the results from Clark et al. (2022).

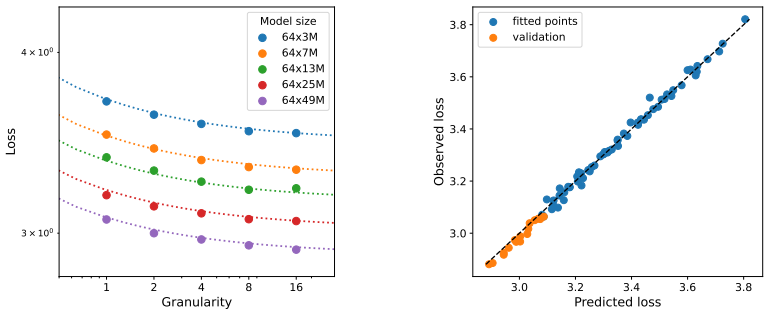
2 Background and Related Work

Mixture of Experts. In the context of language modeling, MoE was first introduced by Shazeer et al. (2017) and later adapted to Transformers (Shazeer et al., 2018; Lepikhin et al., 2020). Fedus et al. (2022) improved the training stability and proposed to route each input to a single expert. Zhou et al. (2022) fixed load balancing issues by designing expert choice routing, which we use in our experiments. Concurrently to our work, Dai et al. (2024) proposed to modify the MoE layer by segmenting experts into smaller ones . Independently, Liu et al. (2023) suggested a unified view of sparse feed-forward layers, considering varying the size of memory blocks. Both approaches can be interpreted as modifying granularity. However, we offer a comprehensive comparison of the relationship between model hyperparameters and derive principled selection criteria.

The core idea behind MoE in Transformers is to replace the feed-forward layer with a set of *experts*. The size of each expert is typically (Fedus et al., 2022; Zhou et al., 2022; Jiang et al., 2024) set to mirror the original dimensions of the layer, with the hidden expert dimension d_{expert} equal to d_{ff}.

Scaling laws. Scaling laws are empirically derived equations relating the loss of a model with variables such as the number of parameters, training samples, or the computational budget. Kaplan et al. (2020) first studied scaling laws for Transformers and observed power law relationships between the perplexity and sizes of the model and the dataset. Hoffmann et al. (2022) extended this work by considering variable LR schedules and formulating a modified functional form of the scaling laws.

Particularly relevant, Clark et al. (2022) studied the scaling of MoE when changing model size and number of experts on a fixed dataset, concluding that routed models are more efficient only until a certain model size. In this work, we challenge that claim by considering a variable, optimal dataset size for both model families (see Section 4.3).



(a) **(b)**

Figure 2: **(a)** Fit of the scaling laws for models trained on 33B tokens. **(b)** Validation of the fit.

3 GRANULARITY

In the standard setting, the inner dimension of each expert network d_{expert} is the same size as the feed-forward layer of the base model, $d_{\text{expert}} = d_{\text{ff}}$. In this work, we suggest an alternative approach where the hidden dimension of the expert is not necessarily set to mirror that of the standard feed-forward layer. Instead, it can be adjusted to a value that is the most effective. This approach allows the configuration of MoE to be articulated in terms of two key hyperparameters: *granularity* (G) and *expansion rate* (E). In the following parts of this work, we will also use the term *active* parameters to refer to the non-embedding parameters used to produce output for a single token, except routing. The number of active parameters is denoted as N_{act}.

Let d_{expert} be the hidden dimension of a single expert. Granularity is defined as $G = \frac{d_{\text{ff}}}{d_{\text{expert}}}$. In other words, granularity denotes the multiplier factor for the change in the size of an expert from the original standard model, defined as $G = 1$. Note that increasing granularity does not affect the number of active parameters since, as G increases, the number of experts that process the token grows proportionally to G. That is, for granularity G, a token is routed to G fine-grained experts, keeping the number of active parameters constant.

We then define the *expansion rate*, which describes the increase in the number of parameters from a standard transformer layer to a MoE layer. Given that, N_{MoE} and N_{ff} denote the total number of parameters in an MoE layer excluding routing, and the standard feed-forward layer, respectively. The expansion rate E is then defined as $E = \frac{N_{\text{MoE}}}{N_{\text{ff}}}$. The definitions of both granularity and expansion rate extend and refine our understanding of the number of experts, symbolized as N_{expert}, $N_{\text{expert}} = G \cdot E$. For non-granular models, where $G = 1$, the expansion rate is equal to the number of experts. Intuitively, increasing granularity for a given expansion rate gives the model more flexibility in mapping datapoints to experts, potentially improving performance.

4 SCALING LAWS

Granularity determines changes in the architecture of MoE. In this section, we aim to derive a parametric scaling law for predicting the final loss value $\mathcal{L}$ based on granularity G, total number of non-embedding parameters N, and number of training tokens D. To this end, we run over 100 experiments on the decoder-only Transformer MoE, with up to 3.7B parameters. The model sizes are reported as $E \times N_{\text{act}}$, expansion rate and active parameters. We begin the analysis with partial observations based on empirical results. We then combine them into a joint equation.

Power Law With Respect to Granularity. We first answer the question of whether granular models follow the scaling laws. In Fig. 2 and in Fig. 3, we can notice that increasing granularity results in a lower loss. The returns follow approximately an exponential pattern, converging to a positive constant. The empirical relationship given by Fig. 3, suggests the following power-law dependence of loss on a varying granularity for given N and D, $\mathcal{L}_{N,D}(G) = \frac{g_{N,D}}{G^{\gamma_{N,D}}} + h_{N,D}$.

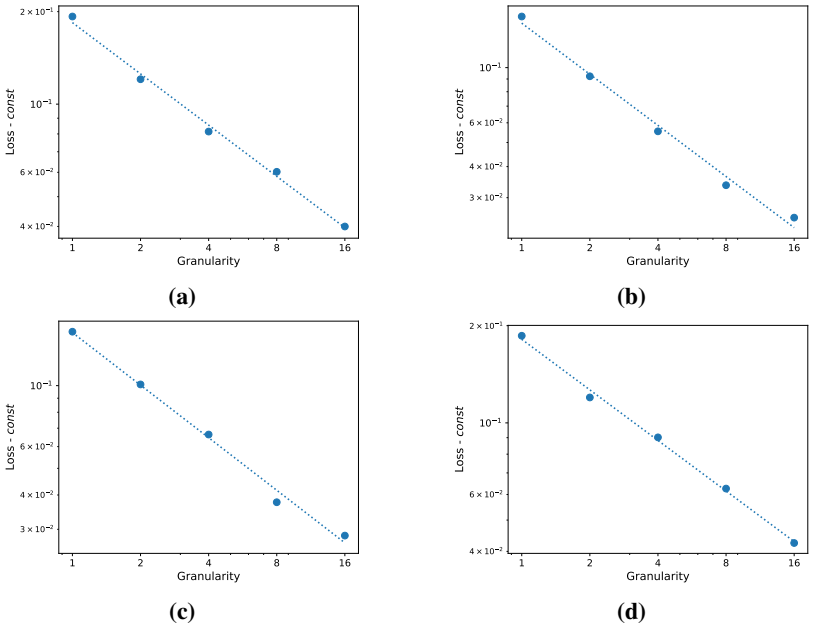


Figure 3: Illustration of scaling granularity when N, D are fixed for: **(a)** $N = 64 \times 25M$, $D = 16B$, $const = 3.12$ **(b)** $N = 64 \times 49M$, $D = 16B$, $const = 3.02$ **(c)** $N = 64 \times 25M$, $D = 32B$, $const = 3.03$ **(d)** $N = 64 \times 49M$, $D = 32B$, $const = 2.88$

Scaling the Model and Dataset Size. The power-law given by equation 2 consists of three terms that describe inherent data entropy and limitations in function representation and data. This derivation is independent of the architecture. In particular, the equation 2 also holds for constant granularity. Empirically, we observe a power law relationship in N and D analogous to that in dense models (see also Fig. 1 in Kaplan et al. (2020)), as depicted in Fig. 5, Appendix, for a fixed value of granularity. Furthermore, the validity of this functional form is verified by fit in Section 4.2.

Since we know that separate scaling laws are valid for given granularities, in the general form, the parameters in equation 2 can depend on the model's granularity, $\mathcal{L}_G(N, D) = c_G + \frac{a_G}{N^{\alpha_G}} + \frac{b_G}{D^{\beta_G}}$.

4.1 The Form of the Joint Scaling Law

Following the above observation that models with constant granularity obey Chinchilla scaling laws given by equation 2, the key question arises as to how the general notion of granularity G can be incorporated into the joint scaling law. The objective is to identify a function that fulfills these criteria, $\mathcal{L}(N, D, G) = \mathcal{L}_{N,D}(G) = \mathcal{L}_G(N, D)$. We aim to determine which of these parameters remain independent of G and present some rationale for the structure of our formula.

Lower Bound. Consider the limit for N and D growing to infinity, $\lim_{\substack{N\to\infty \\ D\to\infty}} \mathcal{L}(N, D, G) = c_G$, with the constant term c_G dependent on granularity. This is contradictory to the fact that it captures the inherent entropy of the dataset. The lower bound of the achievable loss when training bigger models on more samples should not depend on the architecture. Therefore, parameter $c_G = c$ is constant for all granularities.

Granularity and Number of Tokens D. As seen in Fig. 4, Appendix, the benefit of training a model on a larger dataset is almost the same for each granularity value. This suggests that there is no interaction between D and G. Therefore, we can assume that $\frac{b_G}{D^{\beta_G}} = \frac{b}{D^{\beta}}$.

Granularity and Model Size N. We consider α to be a constant that describes how the function scales with N. In this work, we assume polynomial functional forms that rule out the potential dependency of α on G given the form of $\mathcal{L}_{N,D}(G)$. Therefore, the only element dependent on G is a_G:

$$\mathcal{L}(N, D, G) = c + \left(\frac{g}{G^\gamma} + a\right) \frac{1}{N^\alpha} + \frac{b}{D^\beta}. \tag{1}$$

Finally, one could consider omitting the constant a in the equation above, and it would still reduce to $\mathcal{L}_{N,D}(G)$. However, this would mean that a model with infinite granularity and a small number of active parameters can achieve the perfect perplexity of the lower bound. We think that MoE sparse model should not exceed the performance of its dense counterpart matched by a total number of parameters and with all of them activated. This means that constant a can act as a marginal improvement from granularity.

4.2 Fitting the Parametric Scaling Law

Subsequently, we fit parameters in equation 1 to describe the scaling of MoE. We depict the fit of the equation in Fig. 2 (and Fig. 6, Appendix). We generally observe a good fit, with RMSE $= 0.015$. The exact values are given in Table 3, Appendix D. We validate the stability of the fit by excluding the top 20% of models with the lowest perplexity and finding the coefficients based on the remaining experiments. We observe that the formula remains almost unchanged in this scenario (see Table 4 in Appendix E). The validation RMSE is 0.019. Results are depicted in Fig. 2 b).

4.3 MoE is Always More Efficient

Subsequently, we find optimal N, D, G for a given computational budget F. This can be done by solving the following optimization problem, minimize $\mathcal{L}(N, D, G)$ subject to FLOPs$(N, D, G) = F$.

Contrary to the results from Clark et al. (2022), in Fig. 1, we can see that Mixture-of-Experts can always be considered more efficient than dense Transformers, regardless of the model size. According to observations from Appendix D.1, MoE models scale better with optimal training. However, for short training schedules, they may under-perform dense models. This means that for constant training time and increasing model size, there exists a point where both models will become very under-trained, in which scenario, dense models surpass MoE. This shows why in Clark et al. (2022), where varying the number of training tokens has not been considered, MoE was predicted to be underperforming for models bigger than $1T$. However, when all training hyper-parameters N, D, G are properly selected to be compute-optimal for each model, the gap between dense and sparse models only increases as we scale. See Appendix F for the details of the compute-optimal setup derivation.

5 Conclusions

This study introduces a novel hyperparameter, granularity (G), and underscores the significance of adjusting it to optimize the efficiency of MoE models. A central finding of this research is that a standard granularity of $G = 1$ is suboptimal across a broad range of FLOPs, leading to the recommendation of using higher granularity to enhance MoE model performance and efficiency. Simultaneously, this work emphasizes the importance of varying training duration for compute-optimal settings. Both granularity and variable training length are incorporated into new scaling laws, confidently demonstrating that MoE models outperform dense transformers at any computing budget. This work not only sheds new light on the scaling laws applicable to MoE models but also provides practical guidance for improving computational efficiency in LLMs.

ACKNOWLEDGMENTS

We would like to express sincere gratitude to Piotr Miłoś and Tomasz Trzciński for valuable feedback and to Aleksandra Weglarz for her help with graphic design.

This work was funded by IDEAS NCBR, which also provided significant computational resources a supportive research environment and direction. The research was supported by PL-Grid infrastructure (grant PLG/2023/016148). We also benefited from the Entropy cluster (hosted at the Faculty of Mathematics, Informatics and Mechanics of the University of Warsaw) funded by NVIDIA, Intel, the Polish National Science Center grant 2022/45/N/ST6/02222, and ERC Starting Grant TOTAL. Marek Cygan was partially supported by an NCBiR grant POIR.01.01.01-00-0392/17-00.

REFERENCES

Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023.

Richard P. Brent. An algorithm with guaranteed convergence for finding a zero of a function. *Comput. J.*, 14:422–425, 1971. URL https://api.semanticscholar.org/CorpusID:10312755.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George van den Driessche, Eliza Rutherford, Tom Hennigan, Matthew Johnson, Katie Millican, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. Unified scaling laws for routed language models, 2022.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts, 2022.

Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Osi, Prateek Sharma, Fan Chen, and Lei Jiang. Llmcarbon: Modeling the end-to-end carbon footprint of large language models, 2024.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022.

Elias Frantar, Carlos Riquelme, Neil Houlsby, Dan Alistarh, and Utku Evci. Scaling laws for sparsely-connected foundation models, 2023.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022.

Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964. doi: 10.1214/aoms/1177703732. URL https://doi.org/10.1214/aoms/1177703732.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.

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

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020.

Zeyu Leo Liu, Tim Dettmers, Xi Victoria Lin, Veselin Stoyanov, and Xian Li. Towards a unified view of sparse feed-forward network in pretraining large language model, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake Hechtman. Mesh-tensorflow: Deep learning for supercomputers, 2018.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

BigScience Workshop, :, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral,

Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névéol, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Daniel McDuff, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Periñán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A Castillo, Marianna Nezhurina, Mario Sänger, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan

Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. Bloom: A 176b-parameter open-access multilingual language model, 2023.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models, 2023.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing, 2022.

A Architecture and Training Setup

All of the models considered in this work are decoder-only Transformers trained on the C4 dataset (Raffel et al., 2023). We use GPT2 tokenizer (Radford et al., 2018). Each batch consists of 0.5M tokens packed into 2048 sequences. Our optimizer is AdamW (Loshchilov & Hutter, 2019), with a weight decay of 0.1. In each training run, we use the maximum learning rate of $2\mathrm{e}{-4}$, with linear warmup for 1% steps and cosine decay to $2\mathrm{e}{-5}$. To improve stability, we initialize weights using the truncated normal distribution with reduced scale, as advised in Fedus et al. (2022). The models are trained using mixed precision; we always keep the attention mechanism and router in high precision. We assume the *infinite data* regime, as the number of training tokens for any of the runs in less than the number of tokens in the corpus. We follow Hoffmann et al. (2022) and perform our analysis on the smoothed training loss.

In MoE, we use the Expert Choice routing algorithm, as it guarantees a balanced expert load without tuning additional hyperparameters. To maintain compatibility with autoregressive language modeling, we apply the recipe described in Zhou et al. (2022): tokens are grouped by position across different sequences. The group size is always set to 256. We replace each feed-forward layer with MoE and activate an average of $8d_{\text{model}}^2$ parameters per token in each MoE layer (except routing). In the router, softmax is performed over the expert dimension, while we choose tokens over the token dimension, as this leads to the best performance (as opposed to performing softmax over the token dimension). We put an additional layer normalization before the output of MoE layer. This gives a small improvement for standard MoE, but is crucial for the performance of models with $G > 1$.

Table 1 and table 2 list the considered architecture and training variants for dense and MoE models, respectively.

Table 1: Architecture and training variants (MoE models).

#parameters (nonemb)	d_{model}	n_{blocks}	n_{heads}	D (in #tokens)	G
64x3M	256	4	4	16B, 33B, 66B	1, 2, 4, 8, 16
64x7M	384	4	6	16B, 33B, 66B	1, 2, 4, 8, 16
64x13M	512	4	8	16B, 33B, 66B	1, 2, 4, 8, 16
64x13M	512	4	8	130B	1, 2, 4
64x25M	512	8	8	16B, 33B,	1, 2, 4, 8, 16
64x25M	512	8	8	66B	1, 2, 4, 8
64x49M	640	10	10	16B, 33B	1, 2, 4, 8, 16
64x49M	640	10	10	66B	1, 2, 4
64x85M	768	12	12	33B	1, 2, 4

Table 2: Architecture and training variants (dense models).

#parameters (nonemb)	d_{model}	n_{blocks}	n_{heads}	D (in #tokens)
3M	256	4	4	16B, 24B, 33B, 66B
6M	256	8	4	16B, 24B, 33B, 66B
13M	512	4	8	16B, 24B, 33B, 66B
25M	512	8	8	16B, 24B, 33B, 66B
49M	640	10	10	16B, 24B, 33B, 66B
85M	768	12	12	16B, 33B

B Scaling Laws Background

B.1 Transformer Scaling

Large Transformer-based models are known to approximately obey the power-law relationship between final loss $\mathcal{L}$, model size N, and number of training tokens D. This relationship is often called *Chinchilla scaling laws* described in Hoffmann et al. (2022) as

$$\mathcal{L}(N, D) = c + \frac{a}{N^{\alpha}} + \frac{b}{D^{\beta}}. \tag{2}$$

The power-law formula is composed of three distinct terms that characterize the intrinsic entropy of data, constraints of the model, and limitations in the training data. The term c represents the minimum possible error intrinsic to the data. The remaining two terms are suboptimality terms, which address the limitations in function representation owing to the size of the model and in data signified by the number of tokens. In the limit, with infinite data and model size, the loss is reduced to c.

B.2 Mixture of Experts

For MoE Transformer-based models, Clark et al. (2022) formulated the final loss for a constant dataset size D of 130B tokens, allowing for variations in the expansion rate E, as:

$$\mathcal{L}(N, E) = \left(\frac{10^{d/a}}{N}\right)^{a} \left(\frac{1}{E}\right)^{b+c\log N}. \tag{3}$$

However, this result has a notable limitation as it can be applied only to the original dataset size. The scalability and effectiveness are constrained in this scenario because it is crucial to align the number of training samples with the available computational resources for optimal use. As per Kaplan et al. (2020) and Hoffmann et al. (2022), maintaining a constant dataset size while scaling up the neural network size leads to undertraining, resulting in a model that does not perform to its full potential.

C Visualizations for the Derivation of the Scaling Law

D Fitting the parametric scaling law

This section details results of scaling laws. In addition to MoE, we also perform fitting for dense Transformer given by equation 2. Similarly to Hoffmann et al. (2022), we use Huber loss (Huber, 1964), with $\delta = 0.1$. The optimization is performed using the BFGS algorithm. We include a weight decay of $5e-4$ to enhance generalization. We start with fitting parameters in equation 1 and then find architecture-dependent coefficients α, β, A and B in equation 2. The values are presented in Table 3.

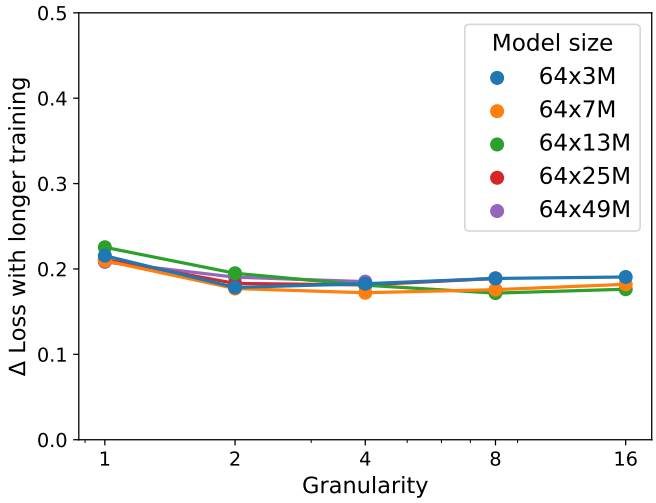


Figure 4: The difference in the loss between training for 16B and 65B tokens for all model sizes and granularity values. The model size is reported as the expansion rate and the number of active parameters.

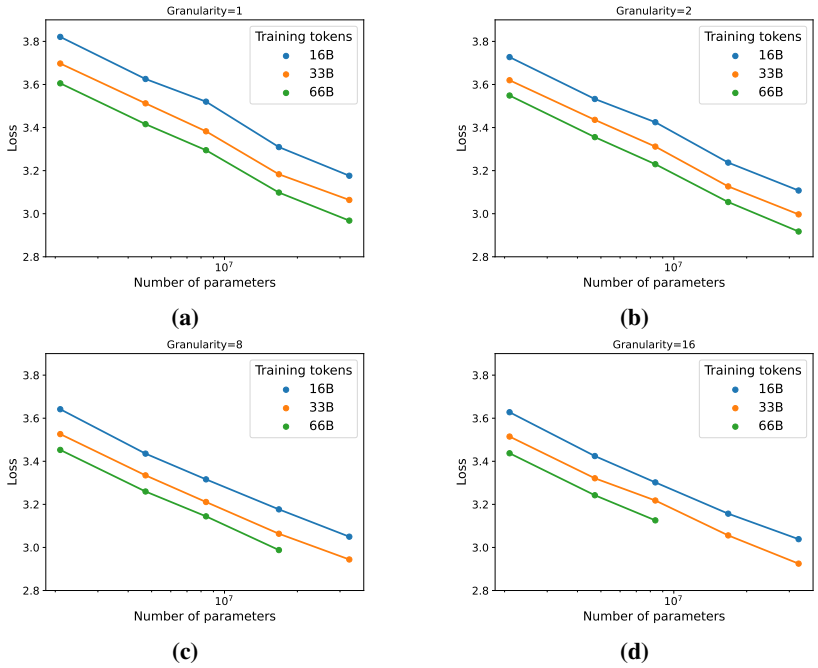


Figure 5: Illustration of scaling N and D for constant granularity value of: **(a)** $G = 1$ **(b)** $G = 2$ **(c)** $G = 8$ **(d)** $G = 16$.

D.1 MoE Scaling Properties

Comparing the part of the formula that approximates underfitting (that is, dependent on training tokens) in MoE ($30.8D^{-0.147}$) and Transformer ($26.7D^{-0.127}$), we can infer that MoE models need longer training to perform competitively but scale better after reaching that point. Nonetheless, this moment may still precede the compute-optimal for both models. On the other hand, we can see that the exponent on dense models $\alpha = -0.126$ scales better with a total number of parameters than the

Table 3: Values of the fitted coefficients.

Model	a	α	b	β	g	γ	c
MoE	18.1	0.115	30.8	0.147	2.1	0.58	0.47
Dense	16.3	0.126	26.7	0.127	-	-	0.47

MoE counterpart $\alpha = -0.115$. This should not be surprising since dense models use all parameters on each token contrary to MoE, which gains a computational advantage by activating only a subset of them. Therefore, the fair comparison of the performance has to take into account FLOPs used by each model type.

E Validation of the Scaling Law

Figure 6 presents the quality of the fit of the scaling law.

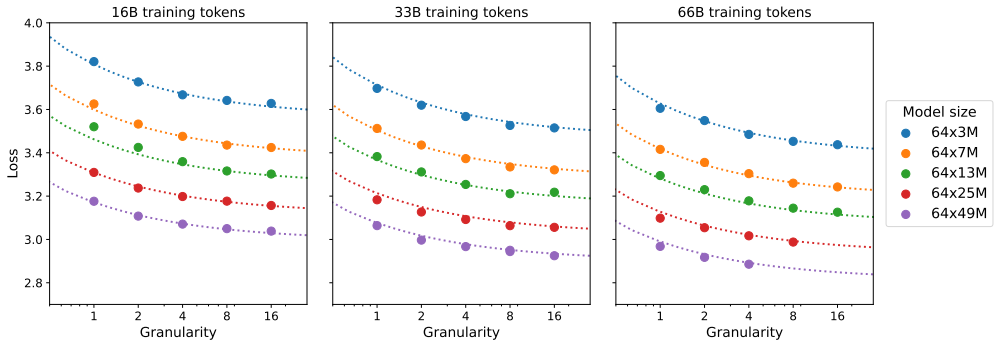


Figure 6: We present the fit of the scaling law compared to experimental results.

In Table 4, we provide coefficients of the scaling law fitted with 20% of datapoints with the lowest perplexity excluded for the purpose of validation.

Table 4: Values of the fitted coefficients.

Model	a	α	b	β	g	γ	c
MoE	17.6	0.114	26.7	0.140	2.07	0.570	0.472

F Compute-optimal Formula

The main component responsible for higher costs is the increase in routing operations due to a larger pool of granular experts. This increase is proportional to the value of G. For standard, non-granular MoE models ($G = 1$), the routing overhead still exists, although it has been considered negligible. Taking into account the routing operation overhead, the number of used FLOPs F is described by the following formula, $F = (12{d_{\text{model}}}^2 c_f + d_{\text{model}} EGc_r) \cdot D \cdot n_{\text{blocks}}$, given expansion rate E, granularity G, and constants that denote FLOPs per active parameter ratio, respectively, within routing (c_r) and within the rest of the network (c_f). The derivation of this formula can be found in Appendix J.

Given all the constraints, we need to solve the following optimization problem: given F,

$$\begin{aligned} \underset{N,D,G}{\text{minimize}} \quad & \mathcal{L}(N, D, G) \\ \text{subject to} \quad & F = (12{d_{\text{model}}}^2 c_f + d_{\text{model}} EGc_r) \cdot D \cdot n_{\text{blocks}} \\ & N = d_{\text{model}}^2 \cdot (8E + 4) \cdot n_{\text{layers}}, \\ & d_{\text{model}} = 64 \cdot n_{\text{layers}}. \end{aligned}$$

All these constraints are reducible to a one-dimensional optimization problem, which is, however, hard to solve analytically. Therefore, we approximate the solution using Brent's method (Brent, 1971). The results of this optimization for varying FLOPs budgets are plotted in Fig. 1 while the optimal configurations of parameters for selected model sizes are presented in Table 5. To validate the uncertainty of these predictions, we follow Hoffmann et al. (2022) and calculate the 10th and 90th percentiles estimated via bootstrapping data (see Appendix G for the detailed results).

Table 5: Compute optimal training hyper-parameters for MoE models. Optimal N and D follow approximately similar relation to these of Hoffmann et al. (2022) for active parameters around the range of $1B$ to $10B$ parameters, requiring comparably longer training for smaller models and shorter for bigger ones. Note that this also considers optimal granularity and its FLOPs cost.

N	D	G	FLOPs	Loss
64 x 100M	4.37B	8	2.95e+18	3.133
64 x 1B	28.94B	16	1.93e+20	2.491
64 x 3B	72.90B	16	1.41e+21	2.245
64 x 7B	137.60B	32	6.46e+21	2.076
64 x 70B	941.07B	32	4.16e+23	1.694
64 x 300B	2.96T	64	5.69e+24	1.503
64 x 1T	7.94T	64	4.97e+25	1.367

The term ${12d_{\text{model}}}^2$ is the number of active parameters within a transformer block, while $d_{\text{model}} EGc_r$ is the number of active parameters within a routing network. The in-depth analysis of constants c_r and c_f can be found in Appendix J. We exclude embedding and unembedding from the FLOPs calculations, following Hoffmann et al. (2022).

Observe that, in contrast to scenarios where routing operations are omitted, the FLOPs calculation that incorporates routing overhead relies on both d_{model} and n_{blocks}. Consequently, an additional condition is required to determine the scaling of d_{model} and n_{blocks} in relation to an increase in N, the number of parameters. It is noted that minor variations in the depth-to-width ratio are not significant (Kaplan et al., 2020). Following this analysis, we opt to adopt the assumption that $d_{\text{model}} = 64 n_{\text{blocks}}$.

The total number of parameters in the feed-forward layer, excluding the routing matrix, is $2Ed_{\text{ff}}d_{\text{model}} = 8E{d_{\text{model}}}^2$, and $4{d_{\text{model}}}^2$ in attention (key, query, value, and output projection). This results in the following formula for $N = {d_{\text{model}}}^2 \cdot (8E + 4) \cdot n_{\text{blocks}}$.

G Reliability of Compute Optimal Formula

In this section, we assess the stability of our predictions presented in Appendix F. Similarly to Hoffmann et al. (2022) we calculate the 10$^{\text{th}}$ and 90$^{\text{th}}$ percentiles estimated via bootstrapping data (80% of the data is sampled 100 times). See Table 6 for the details.

H Discussion

Extreme Granularity. In Section 4, we argue that model performance improves with increasing granularity. This postulate largely aligns with the empirical findings of our study. Nonetheless, at exceedingly high granularity levels, such as $G = 64$ in models characterized by $d_{\text{model}} = 256$ and $E = 64$, there is an observable decline in performance. This phenomenon is particularly evident

Table 6: 10^{th} and 90^{th} percentiles estimated via bootstraping data.

N	D	G
64 x 100M	(2.97B, 5.98B)	(8, 8)
64 x 1B	(21.17B, 40.73B)	(16, 16)
64 x 3B	(50.20B, 105.88B)	(16, 32)
64 x 7B	(101.06B, 205.40B)	(32, 32)
64 x 70B	(638.49B, 1.59T)	(32, 64)
64 x 300B	(1.99T, 5.62T)	(64, 64)
64 x 1T	(5.29T, 16.87T)	(64, 64)

in scenarios where the number of parameters in the routing mechanism exceeds active parameters in actual experts. Additionally, as described in Section 4.3, the utility of such high granularity is predominantly restricted to models of substantial size. In alignment with the principles outlined in Hoffmann et al. (2022), this research focuses more on findings that can be broadly applied rather than delving into the specific details of these corner-case situations. However, it is hypothesized that the efficiency of models with significantly high granularity could be potentially enhanced through careful expert initialization or modifications to the routing algorithm. These ideas are set aside to be investigated in future studies.

Varying Expansion Rate. In this study, due to computational resources constraint, we focus on $E = 64$, as recommended by Clark et al. (2022). This value of E was also used for the largest models in other works (Du et al., 2022; Zhou et al., 2022) and the best-performing configuration in Fedus et al. (2022). Nonetheless, we acknowledge the importance of considering different expansion rates, as different levels of E may be chosen based on factors like the target size of the model in memory. Therefore, in Appendix I, we present the results of the study for $E = 16$ and show that the main findings of this work are still valid in such cases.

Including E in the formula. Another possible advancement would be to unify all of the factors N, D, G and E in one formula. While this would open the possibility of studying the relationships between coefficients in more detail, it would also be hard to practically recommend the optimal configuration in such a scenario using only FLOPs. This is because larger values of E typically lead to better performance but also incur additional memory requirements. Therefore, the choice of expansion rate may be heavily dependent on the available hardware configuration. We leave a detailed study of these factors for future work.

Modeling the cost of granularity. It is important to note that the exact estimation of the training cost of MoE models is dependent on the training setup, hardware, and implementation. Specifically, increasing G can lead to higher transfer costs, depending on the adopted model of distributed training. Therefore, the precise selection of hyperparameters should be made considering these factors. In this work, we model the cost of operations using FLOPs, which is common in the Scaling Laws literature (Kaplan et al., 2020; Hoffmann et al., 2022; Frantar et al., 2023). Additionally, we would like to note that in our setup, we observe significant gains of granular models measured as wall-clock time needed to achieve given perplexity (see Fig. 7 for an example).

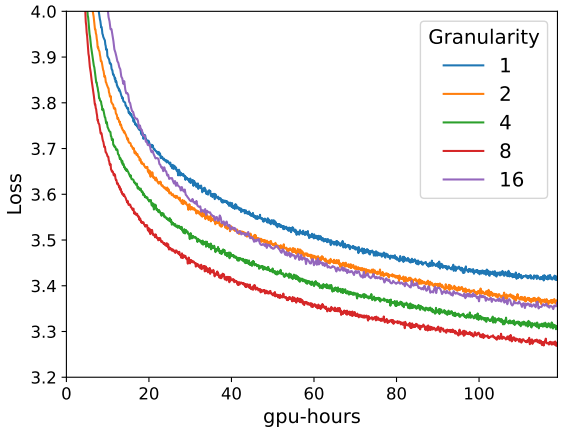


Figure 7: Training loss curves for model with $N = 64 \times 7M$, $D = 66B$ tokens, measured on NVIDIA A100 GPU. We can see that the model with $G = 8$ achieves the best performance.

I Varying Expansion Rate

In this section, we provide results for $E = 16$. The training procedure is the same as described in App. A. The models considered in this part are listed in Table 7.

Table 7: Architecture and training variants (MoE models).

#parameters (nonemb)	d_{model}	n_{blocks}	n_{heads}	D (in #tokens)	G
64x3M	256	4	4	8B, 16B, 33B	1, 2, 4, 8, 16
64x7M	256	8	4	8B, 16B, 33B	1, 2, 4, 8, 16
64x13M	512	4	8	8B, 16B, 33B	1, 2, 4, 8, 16
64x13M	512	4	8	66B	1, 2, 4
64x25M	512	8	8	8B, 16B, 33B	1, 2, 4, 8, 16
64x49M	640	10	10	8B	1, 2, 4, 8, 16

We fit Eq. 1 using the same procedure as described in Section 4.2. The results are detailed in Table 8.

Table 8: Values of the fitted coefficients.

Model	a	α	b	β	g	γ	c
MoE ($E = 16$)	19.64	0.124	57.07	0.169	1.18	0.986	0.472

Using the coefficients and FLOPs calculation formulas, we can derive the compute optimal training parameters. The results are presented in Table 9.

We can observe that similarly to the case when $E = 64$, larger compute budgets imply larger optimal values of G. Note that the values for 10^{th} and 90^{th} percentiles form larger intervals in this case, as in this part we run a smaller number of experiments and keep shorter training durations. However, we believe that this preliminary study forms a valuable addition to the results in the main part.

Table 9: 10^{th} and 90^{th} percentiles estimated via bootstrapping data for $E = 16$.

N	D	G
16 x 100M	(10.29B, 17.73B)	(8 , 16)
16 x 1B	(53.74B, 103.54B)	(16, 32)
16 x 3B	(106.22B, 261.04B)	(16, 32)
16 x 7B	(177.65B, 511.43B)	(16, 32)
16 x 70B	(721.60B, 3.22T)	(32, 64)
16 x 300B	(1.73T, 10.69T)	(32, 64)
16 x 1T	(3.60T, 28.22T)	(32, 128)

J FLOPs Constants

The number of FLOPs F used in Transformer training, considering the routing operation overhead in MoE, can be described by the following formula:

$$F = (12{d_{\text{model}}}^2 c_f + d_{\text{model}} E G c_r) \cdot n_{\text{tokens}}. \tag{4}$$

Following Hoffmann et al. (2022), we assume c_f to be 6. This is interpreted as 6 FLOPs for each pair of an active parameter (in linear projection) and a processed token. The breakdown of operations is as follows:

- During the forward pass, 2 operations (single multiplication and single addition) are used to compute the matrix multiplication of an input and linear projection.
- During the backward pass, 2 operations are used to compute gradients wrt. the input.
- During the backward pass, 2 operations are used to compute gradients wrt. the weights of linear projection.

In our work, we have assumed the routing constant, c_r, to be 14, with the breakdown presented below. The exact number of operations may depend on the implementation of routing, but it will be between 6 and 20. However, our main conclusions of the paper are resistant to different assumptions of this constant.

- During the forward pass, 2 operations are used to compute the expert logits based on an input and "routing linear projection".
- During the backward pass, 2 operations are used to compute gradients for "routing linear projection" wrt. the input.
- During the backward pass, 2 operations are used to compute gradients for "routing linear projection" wrt. the weights of linear projection.
- During the forward pass, 2 operations are used to route input tokens to chosen experts.
- During the forward pass, 2 operations are used to route expert outputs to chosen tokens and multiply those outputs by the routing score.
- During the backward pass, 2 operations are used to route gradients from output tokens to experts.
- During the backward pass, 2 operations are used to route gradients from experts to input tokens.

Similarly to the calculation of FLOPs for c_f, FLOPs come in pairs as each multiplication is followed by an addition (used to accumulate outputs or gradients).