Establishing a Scale for Kullback–Leibler Divergence in Language Models Across Various Settings

Ryo Kishino[1] **Yusuke Takase**[1] **Momose Oyama**[1,2]
Hiroaki Yamagiwa[3*] **Hidetoshi Shimodaira**[1,2]
[1]Kyoto University [2]RIKEN [3]SB Intuitions
kishino.ryo.32s@st.kyoto-u.ac.jp,
{y.takase, oyama.momose}@sys.i.kyoto-u.ac.jp,
hiroaki.yamagiwa@sbintuitions.co.jp, shimo@i.kyoto-u.ac.jp

Abstract

Log-likelihood vectors define a common space for comparing language models as probability distributions, enabling unified comparisons across heterogeneous settings. We extend this framework to training checkpoints and intermediate layers, and establish a consistent scale for KL divergence across pretraining, model size, random seeds, quantization, fine-tuning, and layers. Analysis of Pythia pretraining trajectories further shows that changes in log-likelihood space, as measured by the scaling behavior of KL divergence, are much smaller than in weight space, resulting in subdiffusive learning trajectories and early stabilization of language-model behavior despite weight drift.

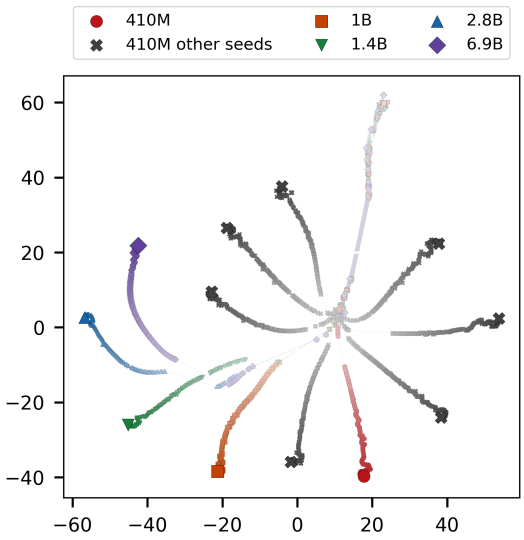


Figure 1: Visualization of the pretraining trajectories of Pythia models across five model sizes and seven random seeds using t-SNE (perplexity=30) applied to double-centered log-likelihood vectors. Each point represents a model checkpoint, with lighter colors indicating earlier steps. The thickness of the connecting lines reflects the magnitude of KL divergence between successive checkpoints. See Appendix D for additional visual analyses.

1 Introduction

To understand learning dynamics and intermediate-layer representations in language models, it is essential to quantify changes in model behavior and compare them across models. While traditional analyses rely on weight parameters (Chen et al., 2022; Kunin et al., 2024; Singh et al., 2025; Gigante et al., 2019), weight permutation symmetries (Hecht-Nielsen, 1990) and architectural dependencies hinder direct comparisons between models with different learning methods or designs.

Oyama et al. (2025) introduced the log-likelihood vector for large-scale comparison of language models, representing each model by its probabilities over a set of texts. They showed that models with different architectures can be embedded in the same space. Moreover, since the squared Euclidean distance in this space estimates the Kullback–Leibler (KL) divergence, model comparison can be formulated as a geometric problem.

In this study, we demonstrate that log-likelihood vectors provide a consistent representation not only for fully trained models but also for training checkpoints, quantized models, and intermediate layers. We therefore handle diverse models within a single coordinate system (see Figs. 1 and 2). We establish KL divergence scales across various settings, including checkpoints, model sizes, random seeds, quantization, fine-tuning, and layers as practical metrics for model comparison.

Moreover, analysis of Pythia pretraining trajectories via KL-scaling exponents shows that, despite large changes in weight space, the corresponding changes in log-likelihood space remain remarkably small. This discrepancy appears as subdiffusive behavior in later training, suggesting early stabi-

*Work done while at Kyoto University.

Our code is available at https://github.com/shimo-lab/modelmap.

Findings of the Association for Computational Linguistics: ACL 2026, pages 23223–23248

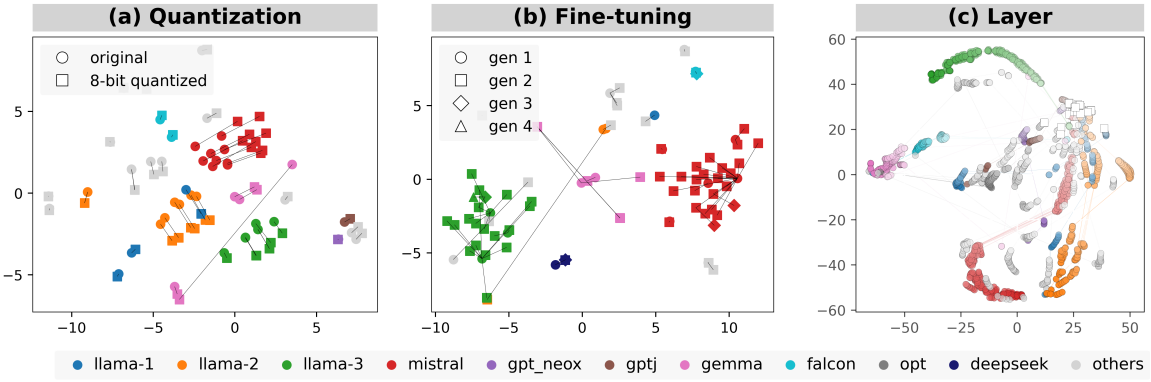


Figure 2: t-SNE visualizations in the space of double-centered log-likelihood vectors. (a) Original models and their 8-bit quantized models (perplexity=30). (b) Base models and their fine-tuned models (perplexity=20). Lines represent parent-child relationships, and generation refers to the depth from the root model within each lineage. (c) Trajectories across layers for 50 models (perplexity=30), obtained by applying the logit lens to each layer. Deeper layers are represented with lighter colors, and the final layer is indicated by a white square.

lization of the model's output distribution despite continued weight drift.

2 Preliminaries: Model Map and KL

Log-likelihood vector. The probability that a language model p generates a text (i.e., a sequence of tokens) $x = (y_1, \ldots, y_M)$ is given by $p(x) = \prod_{m=1}^{M} p(y_m \mid y_0, \ldots, y_{m-1})$. Oyama et al. (2025) defined the log-likelihood vector of a model p over a predefined text set[1] $\{x_1, \ldots, x_N\}$ as

$$\boldsymbol{\ell} = (\log p(x_1), \ldots, \log p(x_N))^\top \in \mathbb{R}^N,$$

and showed that it is a useful feature representation for model comparison and downstream performance prediction. This method has also been used in subsequent work such as Harada et al. (2025).

KL divergence. Let $\{p_i\}_{i=1}^{K}$ be a set of language models, and $\boldsymbol{L} = (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_K)^\top \in \mathbb{R}^{K \times N}$ be the matrix formed by stacking the log-likelihood vectors $\boldsymbol{\ell}_i$ of each model. Define $\boldsymbol{Q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_K)^\top$ as the matrix obtained by double-centering[2] $\boldsymbol{L}$. Then, the KL divergence between two models p_i and p_j can be approximated as

$$2\mathrm{KL}(p_i, p_j) \approx \|\boldsymbol{q}_i - \boldsymbol{q}_j\|^2 / N.$$

In other words, by associating each model p_i with a coordinate $\boldsymbol{q}_i \in \mathbb{R}^N$, KL divergence is approximated by squared Euclidean distance in this space. We refer to the $\boldsymbol{q}$-coordinate system as the model map throughout this paper. For the interpretation of the KL divergence, the standard error, and the dependence on the text set, see Appendix C.

Model map for various settings. Log-likelihood vectors can be computed not only for fully trained language models but also for models during training (Figs. 1 and 2b) and for quantized models (Fig. 2a). Moreover, by treating the network up to each layer as a single model via the logit lens (nostalgebraist, 2020), intermediate layers can also be analyzed (Fig. 2c). These analyses are difficult to carry out with weight-based approaches (Gigante et al., 2019; Chen et al., 2022; Kunin et al., 2024; Singh et al., 2025).

General experimental settings. For the text set, we use the same 10,000 texts extracted from the Pile (Gao et al., 2020) as in Oyama et al. (2025). The values of KL divergence are normalized by the average text length, $\bar{B} = 972.3188$ bytes, and are reported in units of bits per byte. Accordingly, the matrix $\boldsymbol{Q}$ is divided in advance by $\sqrt{2N\bar{B}\log 2}$, so that the squared Euclidean distance between rows of $\boldsymbol{Q}$ directly approximates the KL divergence in bits per byte.

3 KL Divergence Across Various Settings

In this section, we measure and summarize KL divergence under a variety of settings, and present the results in Fig. 3. KL divergence is an interpretable quantity on an absolute scale; for example, a value of 0.1 bits/byte corresponds to only 0.1 bit per 8-bit text unit, whereas 10 bits/byte indicates a very large difference, exceeding the 8-bit scale of the text itself[3]. As a reference point, the effective per-byte entropy of the text set in our experiments

[1] Olsson et al. (2022) used token-level log-likelihoods.

[2] Row-wise centering (over texts) and column-wise centering (over models), respectively, of the log-likelihood matrix.

[3] Since KL divergence is unbounded, it can exceed 8 bits/byte despite the 8-bit representation of text.

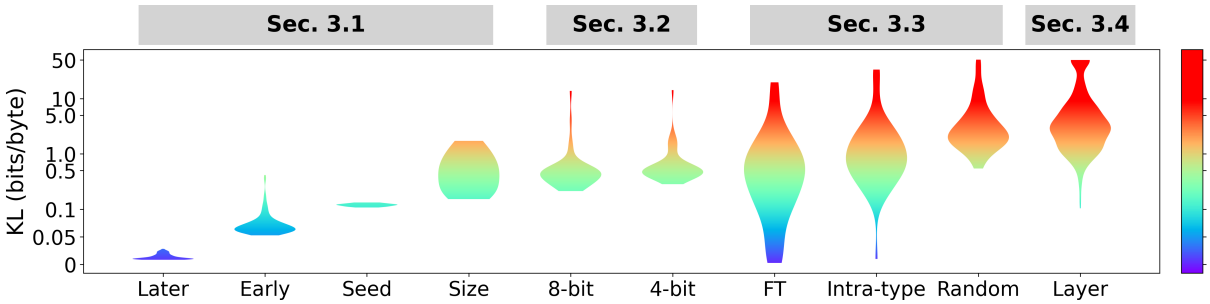


Figure 3: Scale of KL divergence (bits per byte) across various settings. The first four entries from the left correspond to pretraining-related comparisons, showing the KL divergence between consecutive saved checkpoints at 1k-step intervals during the last 50k steps of training, at 1k-step intervals during the first 50k steps, across different random seeds at the final checkpoint, and across different model sizes. The subsequent entries show the KL divergence before and after quantization (8-bit and 4-bit), before and after fine-tuning, between randomly selected pairs within the same model type, between randomly selected pairs across all model types, and between adjacent layers. The vertical axis is shown on a `symlog` scale. See Appendix A for more details.

is approximately 0.71 bits/byte (see Appendix C). Our experiments reveal that each setting is associated with a characteristic scale of KL divergence. Even the value of 0.1 bits/byte can be interpreted as a substantial change between consecutive 1k-step checkpoints during training, but as a minor difference when comparing models of different types.

3.1 Pretraining

Settings. We use publicly available pretraining checkpoints from the Pythia suite (Biderman et al., 2023) for model sizes 410M, 1B, 1.4B, 2.8B, and 6.9B. For the 410M model, checkpoints are available for nine random seeds (1–9) released by van der Wal et al. (2025). From these, we exclude seeds 3 and 4 due to training loss spikes detected via KL divergence, leaving seven seeds for our experiments. Checkpoints are saved at training steps 0, 1, 2, 4, . . . , 512, 1k, 2k, 3k, . . . , and 143k. We exclude the 1B checkpoint at step 116k due to anomalous weights and remove texts exhibiting irregular log-likelihood variations across checkpoints. Details of the preprocessing, data filtering, and anomaly detection procedures for pretraining are provided in Appendix E.

KL divergence. For all model sizes, the KL divergence between consecutive saved checkpoints at 1k-step intervals after the warmup phase is typically 0.05–0.1 bits/byte during the early stage of training, and becomes extremely small in the later stage, ranging from about 0.01 to 0.05 bits/byte. At the final checkpoint, the KL divergence across different random seeds for the 410M model is around 0.1 bits/byte, while that across different model sizes ranges from 0.15 to 1.7 bits/byte, indicating substantially larger differences.

3.2 Quantization

Settings. We analyze the impact of post-training quantization (Dettmers et al., 2022, 2023). We use a subset of the 1,018 language models analyzed in Oyama et al. (2025), selecting the 50 most downloaded models (Fig. 2a); see Appendix F for details on the models used.

KL divergence. The median KL divergence before and after 8-bit quantization is 0.44 bits/byte, while it is slightly larger for 4-bit quantization at 0.49 bits/byte. As shown in Fig. 3, the distributions exhibit low variance in both cases, indicating that the degree of degradation induced by quantization is relatively consistent across models. Furthermore, in Fig. 2a, the shifts induced by quantization appear to be aligned in both direction and magnitude within each model type, suggesting that quantization acts as a structured perturbation in the log-likelihood space rather than as random noise.

To quantitatively validate this observation, we computed all pairwise cosine similarities among the log-likelihood difference vectors induced by 8-bit quantization within each model type. The resulting mean similarities were 0.98 for llama-2 (8 models), 0.91 for llama-3 (7 models), and 0.96 for Mistral (9 models). In contrast, when nine models were randomly sampled irrespective of family, the mean cosine similarity was 0.67, averaged over 100 trials.

3.3 Fine-tuning Effects

Settings. Out of the 1,018 language models analyzed in Oyama et al. (2025), we use 66 pairs of

fine-tuned and base models (87 models in total; Fig. 2b). The parent-child relationships are identified using Hugging Face Hub API. See Appendix G for details.

KL divergence. Fig. 3 shows that fine-tuning induces a relatively small but non-negligible change within a model type, smaller than the variation among randomly paired models of the same type and much smaller than the variation across model types; the corresponding median KL divergence values are 0.40, 0.95, and 2.2 bits/byte.

3.4 Across Layers

Settings. We investigate model trajectories across layers by using the logit lens (nostalgebraist, 2020) to treat each subnetwork, from the input up to a given layer, as an individual model. This allows us to trace how model behavior changes across layers. We use the same set of 50 models as in Section 3.2 (Fig. 2c).

KL divergence. The KL divergence between adjacent layers spans a wide range, with a median of 3.0 bits/byte and a standard deviation of 13 bits/byte. Moreover, within each model type, models follow similar trajectories across layers, as shown in Fig. 2c, and exhibit similar layerwise profiles of KL divergence between adjacent layers, as shown in Fig. 17 in Appendix H.

4 Scaling Analysis of KL Divergence Along Pretraining Trajectories

In this section, we examine the scaling behavior of KL divergence over training time along pretraining trajectories of Pythia models, using the same checkpoints as in Section 3.1.

4.1 Trajectory on the Log-Likelihood Space

Fig. 1 visualizes pretraining trajectories of Pythia models across five model sizes and seven random seeds in the log-likelihood space using t-SNE, enabling joint comparisons that are difficult in weight space[4]. Fig. 4 shows the KL divergence between consecutive saved checkpoints during pretraining. After the warmup phase, the KL divergence between checkpoints saved every 1k steps decreases and, after step 10k, remains within a narrow range regardless of model size, whereas substantially larger values are observed during warmup.

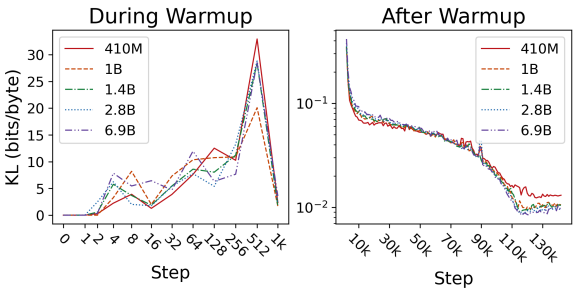


Figure 4: KL divergence between consecutive saved checkpoints of Pythia during pretraining. (Left) Warmup phase with non-uniform checkpoint spacing. (Right) Post-warmup phase with checkpoints saved every 1k training steps.

4.2 Comparison with Weight Space

Anomalous diffusion. We compare the stability of training trajectories by estimating diffusion exponents that characterize the power-law scaling of KL divergence. According to Kunin et al. (2024), after model performance has converged, the weights diffuse following a power law with exponent c_{w}. Specifically, with respect to an initial step t_0, the squared Euclidean distance satisfies $\|\boldsymbol{W}_t - \boldsymbol{W}_{t_0}\|^2 \propto |t - t_0|^{c_{\mathrm{w}}}$. We show that an analogous power-law relationship also holds for the log-likelihood vectors $\boldsymbol{q}_t$: $\|\boldsymbol{q}_t - \boldsymbol{q}_{t_0}\|^2 \propto |t - t_0|^{c_{\mathrm{q}}}$, where the left-hand side is proportional to the KL divergence.

For diffusion exponent c, Brownian motion corresponds to $c = 1$, while $c \neq 1$ indicates anomalous diffusion. In Fig. 5, the weights $\boldsymbol{W}$ exhibit Brownian-like diffusion with $c_{\mathrm{w}} \approx 1$, whereas the diffusion of $\boldsymbol{q}$ is substantially suppressed with $c_{\mathrm{q}} \approx 0.2$. The right panel of Fig. 5 shows how the exponents c_{w} and c_{q} vary with the starting step t_0. Compared to weight trajectories, the smaller exponent c_{q} indicates that trajectories in log-likelihood space remain confined to a relatively narrow region, suggesting early stabilization of language-model behavior despite continued drift in weight space.

Under a fractional Brownian motion interpretation, these exponents correspond to effective fractal dimensions of $D_{\mathrm{w}} \approx 2$ and $D_{\mathrm{q}} \approx 10$, respectively[5]. Thus, the larger effective dimension D_{q} suggests greater geometric complexity of the trajectories in

[4]Two-dimensional t-SNE visualizations may exhibit apparent jumps due to dimensionality constraints, whereas three-dimensional t-SNE visualizations alleviate this issue and support continuity of the sampled trajectories at the displayed resolution; see Appendix D.

[5]If the process is fractional Brownian motion (Mandelbrot and Van Ness, 1968) with diffusion exponent c, the Hurst exponent is $H = c/2$, and the Hausdorff dimension of the trajectory is $D = 1/H$ (Adler, 1981, Theorem 8.4.1).

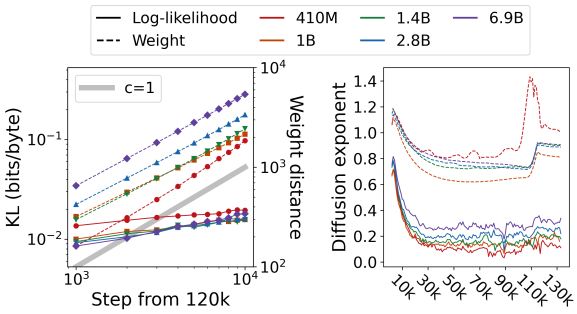


Figure 5: (Left) Temporal evolution of the squared Euclidean distance in the weights and log-likelihood vectors from steps 120k to 130k for Pythia. See Table 1 for the diffusion exponents in the weight and log-likelihood space for each model size estimated using the least squares method. (Right) Diffusion exponent as a function of the starting point t_0. The exponent is estimated using the least squares method.

Size	c_{w}	c_{q}	α	$c_{\exp(\mathrm{q})}$	Diff
410M	1.1	0.15	0.14	0.15	+0.00
1B	0.83	0.20	0.24	0.20	+0.00
1.4B	0.91	0.21	0.23	0.21	+0.00
2.8B	0.90	0.26	0.29	0.26	+0.00
6.9B	0.92	0.33	0.36	0.34	+0.01

Table 1: Diffusion exponents and Hölder exponents for each model size. The checkpoints of Pythia from steps 120k to 130k are used. $\alpha = c_{\mathrm{q}}/c_{\mathrm{w}}$ denotes the Hölder exponent. "Diff" denotes $c_{\exp(\mathrm{q})} - c_{\mathrm{q}}$, where $c_{\exp(\mathrm{q})}$ represents the diffusion exponent in the likelihood space.

log-likelihood space. See Appendix B for details.

Geometric interpretation via folding. The suppressed diffusion exponent c_{q} may be understood through the many-to-one structure of the mapping from weight space to log-likelihood space. Due to permutation symmetries of hidden units, many distinct weight configurations correspond to the same function or output distribution (Fukumizu and Amari, 2000). From a statistical viewpoint, such permutation-induced redundancies are associated with degenerate directions of the Fisher information matrix, giving rise to flat directions in the loss landscape. As a result, trajectories that are well separated in weight space can be mapped to nearby or identical points in the log-likelihood space. Viewed through this many-to-one mapping, training trajectories in the log-likelihood space are strongly folded, leading to complex geometric structures that remain confined to a relatively narrow region of the space.

Quantifying folding via Hölder regularity. The geometric interpretation based on folding can be further quantified by considering the regularity of the mapping $f : \boldsymbol{W} \mapsto \boldsymbol{q}(\boldsymbol{W})$ from weight space to log-likelihood space. Specifically, a mapping is α-Hölder continuous if distances in the output space are bounded by a constant times the α-th power of distances in the input space, with $\alpha = 1$ corresponding to Lipschitz continuity and smaller values of $\alpha > 0$ indicating increasing non-smoothness. If f is α-Hölder continuous and satisfies the non-degeneracy condition required to saturate the theoretical bound, then the fractal dimensions of the corresponding trajectories obey $D_{\mathrm{q}} = D_{\mathrm{w}}/\alpha$ (Falconer, 2014), which implies

$$\alpha = \frac{D_{\mathrm{w}}}{D_{\mathrm{q}}} = \frac{c_{\mathrm{q}}}{c_{\mathrm{w}}}.$$

This relation shows that smaller α corresponds to stronger folding: the weight-space trajectory is mapped to a geometrically more complex trajectory in log-likelihood space while its large-scale displacement is strongly suppressed. Related discussions of α-Hölder regularity and learning dynamics also appear in Ly and Gong (2025), although the setting there differs from ours. In our empirical setting, the observed diffusion exponents imply effective dimensions $D_{\mathrm{w}} \approx 2$ and $D_{\mathrm{q}} \approx 10$, yielding an effective Hölder exponent of $\alpha \approx 0.2$. Table 1 also suggests that α tends to increase with model size, raising the possibility that it may reflect underlying properties of trained language models as manifested in the geometry of training trajectories.

Comparison on the likelihood scale. A possible concern is that the anomalous diffusion reflected in c_{q} might be an artifact of the logarithmic scaling used to define the log-likelihood coordinates. To address this, we repeated the same analysis on the likelihood scale using exponentiated coordinates, i.e., $\exp(\boldsymbol{q})$. As shown in Table 1, the resulting diffusion exponents $c_{\exp(\mathrm{q})}$ are quantitatively consistent with those obtained from $\boldsymbol{q}$, with differences within ± 0.01 across model sizes.

5 Conclusion

We measured KL divergence between language models across a wide range of conditions and established a unified and interpretable scale. We further examined the scaling behavior of KL divergence along pretraining trajectories.

Limitations

- We conduct our experiments using the same set of 10,000 text chunks as those used in Oyama et al. (2025), and the measured KL divergence therefore depends on the choice of the text set. However, we verified that different random subsets of 10,000 text chunks from the same Pile corpus yield highly consistent KL estimates (Appendix C), while the impact of domain shifts across text sets has not been examined.

- The set of models analyzed in this study is limited and does not fully cover the diversity of existing language models. In particular, pretraining trajectory analyses are conducted only on the Pythia family, and experiments involving fine-tuning, quantization, and layer-wise analysis rely on selected subsets of models due to data availability and metadata constraints, such as incomplete base-model information in model cards.

- In the layer-wise analysis, the logits obtained from shallow layers via the logit lens contain substantial noise. This issue could be addressed by using the tuned lens (Belrose et al., 2023), an improved version of the logit lens. However, the number of publicly available pretrained tuned lenses is currently limited. We believe this limitation can be overcome in future work by training tuned lenses.

- The checkpoints for Pythia are available only at 1k-step intervals except for the early stages of training. Accordingly, our analysis focuses on trajectory dynamics at scales of 1k steps or larger, while finer-scale behavior below this resolution is not examined. This limitation does not affect our analysis of dynamics at coarser temporal scales.

- The effective Hölder exponent α in our analysis is only a trajectory-based reference value. Hölder regularity of a mapping f is, in principle, defined through local variation in all directions around each point, whereas we observe only the variation of f along training trajectories. Moreover, these trajectories are shaped by optimization of log-likelihood rather than being truly random. Therefore, the estimated α should not be interpreted as a full characterization of the regularity of f, but only as an indicator of its behavior along training trajectories.

- While we observe structured behaviors such as subdiffusion in the log-likelihood space, this work represents an initial attempt at trajectory analysis in this space, and our analysis of their underlying causes and implications for learning dynamics remains preliminary and has not yet been quantitatively established.

Acknowledgments

We thank Luis Iván Hernández Ruíz for helpful discussions. This work was partially supported by JSPS KAKENHI JP22H05106 and JP23H03355 (to HS), JST CREST JPMJCR21N3 (to HS), JSPS KAKENHI JP25K24366 (to HY), and JST BOOST JPMJBS2407 (to YT and MO).

References

01. AI, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Wen Xie, and 13 others. 2025. Yi: Open foundation models by 01.ai. *Preprint*, arXiv:2403.04652.

Robert J. Adler. 1981. *The Geometry of Random Fields*. John Wiley & Sons, Chichester.

AI@Meta. 2024. Llama 3 model card.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *Preprint*, arXiv:2305.13245.

AI@Waktaverse. 2024. Waktaverse llama 3 model card.

Pieter C Allaart and Kiko Kawamura. 2011/2012. The takagi function: a survey. *Real Anal. Exchange*, 37(1):1–54.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Shivanshu Purohit, Tri Songz, Wang Phil, and Samuel Weinbach. 2021. GPT-NeoX: Large scale autoregressive language modeling in PyTorch.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

Pierpaolo Basile, Elio Musacchio, Marco Polignano, Lucia Siciliani, Giuseppe Fiameni, and Giovanni Semeraro. 2023. Llamantino: Llama 2 models for effective text generation in italian language. *Preprint*, arXiv:2312.09993.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. 2022. Efficient training of language models to fill in the middle. *Preprint*, arXiv:2207.14255.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *Preprint*, arXiv:2303.08112.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Preprint*, arXiv:2004.05150.

Stella Biderman, Kieran Bicheno, and Leo Gao. 2022. Datasheet for the pile. *Preprint*, arXiv:2201.07311.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. *Preprint*, arXiv:2304.01373.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Guozhang Chen, Cheng Kevin Qu, and Pulin Gong. 2022. Anomalous diffusion dynamics of learning in deep neural networks. *Neural Networks*, 149:18–28.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

François Chollet. 2019. On the measure of intelligence. *Preprint*, arXiv:1911.01547.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *Preprint*, arXiv:1905.10044.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Ultrafeedback: Boosting language models with scaled ai feedback. *Preprint*, arXiv:2310.01377.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Preprint*, arXiv:2205.14135.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Preprint*, arXiv:2305.14314.

Peter Devine. 2024. Tagengo: A multilingual chat dataset. *Preprint*, arXiv:2405.12612.

Jwala Dhamala, Tony Sun, Varun Kumar, Satyapriya Krishna, Yada Pruksachatkun, Kai-Wei Chang, and Rahul Gupta. 2021. Bold: Dataset and metrics for measuring biases in open-ended language generation. *Preprint*, arXiv:2101.11718.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *Preprint*, arXiv:2305.14233.

Kenneth Falconer. 2014. *Fractal Geometry: Mathematical Foundations and Applications, 3rd Edition*. John Wiley & Sons.

Kenji Fukumizu and Shun-ichi Amari. 2000. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural networks*, 13(3):317–327.

Victor Gallego. 2024. Configurable safety tuning of language models with synthetic preference data. *Preprint*, arXiv:2404.00495.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling. *Preprint*, arXiv:2101.00027.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *Preprint*, arXiv:2009.11462.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1331 others. 2024. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Xinyang Geng and Hao Liu. 2023. Openllama: An open reproduction of llama.

Scott Gigante, Adam S Charles, Smita Krishnaswamy, and Gal Mishne. 2019. Visualizing the phate of neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Tilmann Gneiting, Hana Ševčíková, and Donald B. Percival. 2012. Estimators of fractal dimension: Assessing the roughness of time series and spatial data. *Statistical Science*, 27(2):247–277.

Yuto Harada, Yusuke Yamauchi, Yusuke Oda, Yohei Oseki, Yusuke Miyao, and Yu Takagi. 2025. Massive supervised fine-tuning experiments reveal how data, layer, and training factors shape LLM alignment quality. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 22371–22392, Suzhou, China. Association for Computational Linguistics.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *Preprint*, arXiv:2203.09509.

Robert Hecht-Nielsen. 1990. On the algebraic structure of feedforward network weight spaces. In Rolf ECKMILLER, editor, *Advanced Neural Computers*, pages 129–135. North-Holland, Amsterdam.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. 2020. Query-key normalization for transformers. *Preprint*, arXiv:2010.04245.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. *Preprint*, arXiv:2403.07691.

"interstellarninja", "Teknium", "theemozilla", "karan4d", and "huemin_art". 2024. Hermes-2-pro-mistral-7b.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *Preprint*, arXiv:1705.03551.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*.

Dahyun Kim, Yungi Kim, Wonho Song, Hyeonwoo Kim, Yunsu Kim, Sanghoon Kim, and Chanjun Park. 2024a. sdpo: Don't use your data all at once. *Preprint*, arXiv:2403.19270.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, Changbae Ahn, Seonghoon Yang, Sukyung Lee, Hyunbyung Park, Gyoungjin Gim, Mikyoung Cha, Hwalsuk Lee, and Sunghun Kim. 2024b. Solar 10.7b: Scaling large language models with simple yet effective depth upscaling. *Preprint*, arXiv:2312.15166.

Daniel Kunin, Javier Sagastuy-Breña, Lauren E. Gillespie, Eshed Margalit, Hidenori Tanaka, Surya Ganguli, and Daniel L. K. Yamins. 2024. The limiting dynamics of SGD: modified loss, phase-space oscillations, and anomalous diffusion. *Neural Comput.*, 36(1):151–174.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning. *Preprint*, arXiv:1910.09700.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report. *Preprint*, arXiv:2309.05463.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. *Preprint*, arXiv:2109.07958.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, and 47 others. 2024. Starcoder 2 and the stack v2: The next generation. *Preprint*, arXiv:2402.19173.

Andrew Ly and Pulin Gong. 2025. Optimization on multifractal loss landscapes explains a diverse range of geometrical and dynamical properties of deep learning. *Nature Communications*, 16(1):3252.

Benoit B Mandelbrot and John W Van Ness. 1968. Fractional brownian motions, fractional noises and applications. *SIAM review*, 10(4):422–437.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *Preprint*, arXiv:1809.02789.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. 2023. Orca 2: Teaching small language models how to reason. *Preprint*, arXiv:2311.11045.

Koh Mitsuda, Xinqi Chen, Toshiaki Wakatsuki, and Kei Sawada. 2024. rinna/llama-3-youko-8b.

MosaicML NLP Team. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2023-03-28.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2025. Generative representational instruction tuning. *Preprint*, arXiv:2402.09906.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *Preprint*, arXiv:2306.02707.

Nexusflow.ai team. 2023. Nexusraven-v2: Surpassing gpt-4 for zero-shot function calling.

Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. 2023. Capabilities of gpt-4 on medical challenge problems. *Preprint*, arXiv:2303.13375.

nostalgebraist. 2020. interpreting gpt: the logit lens. *LessWrong*.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, and 7 others. 2022. In-context learning and induction heads. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Momose Oyama, Hiroaki Yamagiwa, Yusuke Takase, and Hidetoshi Shimodaira. 2025. Mapping 1, 000+ language models via the log-likelihood vector. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 32983–33038. Association for Computational Linguistics.

Ankit Pal and Malaikannan Sankarasubbu. 2024a. Gemini goes to med school: Exploring the capabilities of multimodal large language models on medical challenge problems & hallucinations. *Preprint*, arXiv:2402.07023.

Ankit Pal and Malaikannan Sankarasubbu. 2024b. Openbiollms: Advancing open-source large language models for healthcare and life sciences.

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. 2024. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *Preprint*, arXiv:2402.13228.

Alicia Parrish, Angelica Chen, Nikita Nangia, Vishakh Padmakumar, Jason Phang, Jana Thompson, Phu Mon Htut, and Samuel R. Bowman. 2022. Bbq: A hand-built bias benchmark for question answering. *Preprint*, arXiv:2110.08193.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only. *Preprint*, arXiv:2306.01116.

Marco Polignano, Pierpaolo Basile, and Giovanni Semeraro. 2024. Advanced natural-based interaction for the italian language: Llamantino-3-anita. *Preprint*, arXiv:2405.07101.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. *Preprint*, arXiv:2108.12409.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, and 7 others. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. *Preprint*, arXiv:1804.09301.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *Preprint*, arXiv:1907.10641.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *Preprint*, arXiv:1904.09728.

Akira Sasaki, Masato Hirakawa, Shintaro Horie, and Tomoaki Nakamura. 2023. Elyza-japanese-llama-2-7b.

Tim Sauer, James A Yorke, and Martin Casdagli. 1991. Embedology. *Journal of Statistical Physics*, 65(3):579–616.

Kei Sawada, Tianyu Zhao, Makoto Shing, Kentaro Mitsui, Akio Kaga, Yukiya Hono, Toshiaki Wakatsuki, and Koh Mitsuda. 2024. Release of pre-trained models for the japanese language. *Preprint*, arXiv:2404.01657.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *Preprint*, arXiv:1911.02150.

Abdullah Shidfar and Kazem Sabetfakhri. 1986. On the continuity of van der waerden's function in the hölder sense. *The American Mathematical Monthly*, 93(5):375–376.

Sidak Pal Singh, Bobby He, Thomas Hofmann, and Bernhard Schölkopf. 2025. The directionality of optimization trajectories in neural networks. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Scharli, Aakanksha Chowdhery, Philip Mansfield, Blaise Aguera y Arcas, Dale Webster, and 11 others. 2022. Large language models encode clinical knowledge. *Preprint*, arXiv:2212.13138.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, Mike Schaekermann, Amy Wang, Mohamed Amin, Sami Lachgar, Philip Mansfield, Sushant Prakash, Bradley Green, Ewa Dominowska, Blaise Aguera y Arcas, and 12 others. 2023. Towards expert-level medical question answering with large language models. *Preprint*, arXiv:2305.09617.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, and 432 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Preprint*, arXiv:2206.04615.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding. *Preprint*, arXiv:2104.09864.

Shivchander Sudalairaj, Abhishek Bhandwaldar, Aldo Pareja, Kai Xu, David D. Cox, and Akash Srivastava. 2024. Lab: Large-scale alignment for chatbots. *Preprint*, arXiv:2403.01081.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *Preprint*, arXiv:1811.00937.

"Teknium", Charles Goddard, "interstellarninja", "theemozilla", "karan4d", and "huemin_art". 2024a. Hermes-2-theta-llama-3-8b.

"Teknium", "interstellarninja", "theemozilla", "karan4d", and "huemin_art". 2024b. Hermes-2-pro-llama-3-8b.

"Teknium", "theemozilla", "karan4d", and "huemin_art". 2024c. Nous hermes 2 mistral 7b dpo.

Matus Telgarsky. 2016. Benefits of depth in neural networks. In *Proceedings of the 29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1517–1539. PMLR.

Matus Telgarsky. 2017. Neural networks and rational functions. In *Proceedings of the 34th International*

Conference on Machine Learning, volume 70 of *Proceedings of Machine Learning Research*, pages 3387–3393. PMLR.

Together Computer. 2023. Redpajama-data: An open source recipe to reproduce llama training dataset.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

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, and 49 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment. *Preprint*, arXiv:2310.16944.

Oskar van der Wal, Pietro Lesci, Max Müller-Eberstein, Naomi Saphra, Hailey Schoelkopf, Willem Zuidema, and Stella Biderman. 2025. Polypythias: Stability and outliers across fifty language model pre-training runs. In *The Thirteenth International Conference on Learning Representations*.

Ben Wang. 2021. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. 2024a. Openchat: Advancing open-source language models with mixed-quality data. *Preprint*, arXiv:2309.11235.

Shenzhi Wang, Yaowei Zheng, Guoyin Wang, Shiji Song, and Gao Huang. 2024b. Llama3-8b-chinese-chat (revision 6622a23).

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. 2024. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *Preprint*, arXiv:2408.08152.

Haoran Xu, Young Jin Kim, Amr Sharaf, and Hany Hassan Awadalla. 2024a. A paradigm shift in machine translation: Boosting translation performance of large language models. *Preprint*, arXiv:2309.11674.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024b. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *Preprint*, arXiv:2401.08417.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. *Preprint*, arXiv:2309.12284.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *Preprint*, arXiv:1905.07830.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. *Preprint*, arXiv:1804.06876.

Tianyu Zhao, Akio Kaga, and Kei Sawada. 2023. rinna/youri-7b.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *Preprint*, arXiv:2304.06364.

A Details of Fig. 3

In Fig. 3, the vertical axis, i.e., the KL divergence values, is plotted using a symmetric logarithmic scale (symlog). The scale is linear up to 0.1 bits/byte and logarithmically scaled beyond that point. For the color bar, values are clipped at 10 bits/byte.

Detailed statistics corresponding to Fig. 3 are presented in Table 2a. In addition, for Pythia-410M, we report the KL divergence between checkpoints separated by 1k steps during pretraining, as well as the KL divergence across different seeds and model sizes at the final checkpoint, in Table 2b. Furthermore, for the three concrete models, the KL divergence between selected layers is shown in Table 2c.

B Related Work and Further Discussions on Diffusion Exponents

In this appendix, we discuss related work and possible interpretations of the marked discrepancy between the diffusion behavior in weight space and that in log-likelihood space observed in Section 4.2.

Anomalous diffusion in weight space. It has been reported that, during the training of neural networks, trajectories in weight space can exhibit anomalous diffusion, and this phenomenon has recently attracted considerable research interest (Chen et al., 2022; Kunin et al., 2024; Ly and Gong, 2025). These studies analyze training dynamics through the temporal scaling of distances in weight space and report deviations from standard Brownian motion.

Diffusion exponents in weight and log-likelihood spaces. In Section 4.2, we estimated the diffusion exponents c_{w} and c_{q} for trajectories in weight space and log-likelihood space, respectively. More generally, estimating such scaling exponents from discretely observed data via power-law scaling over a finite range of scales is a classical issue in fractal analysis; see Gneiting et al. (2012) for related methodology and for asymptotic and finite-sample assessments under infill asymptotics[6]. Our results show that during the early phase of training, both exponents take relatively large values ($c_{\mathrm{w}} = 1.2$ and $c_{\mathrm{q}} = 0.8$), but decrease as training progresses. In the later stages, the exponent in weight space approaches $c_{\mathrm{w}} \approx 1$, corresponding to Brownian-like diffusion, whereas the exponent in log-likelihood space becomes much smaller ($c_{\mathrm{q}} \approx 0.2 \ll 1$), which is characteristic of subdiffusion.

In our discussion, fractional Brownian motion is used purely as an interpretive model to relate the observed diffusion exponents to effective fractal dimensions, rather than as a generative model of the training dynamics. Under this interpretation, a trajectory with diffusion exponent c has effective fractal dimension given by

$$D = \frac{2}{c},$$

following the standard Hausdorff-dimension relation for fractional Brownian motion discussed by Adler (1981); see Mandelbrot and Van Ness (1968) for the underlying process model. Accordingly, the representative later-stage values $c_{\mathrm{w}} \approx 1$ and $c_{\mathrm{q}} \approx 0.2$ correspond to $D_{\mathrm{w}} \approx 2$ and $D_{\mathrm{q}} \approx 10$, respectively.

As shown in the right panel of Fig. 5 and Table 1 in Section 4, the estimated diffusion exponent c_{q} exhibits systematic variation across model sizes. In particular, smaller models tend to show lower values ($c_{\mathrm{q}} \approx 0.1$–$0.2$), while larger models exhibit higher values ($c_{\mathrm{q}} \approx 0.2$–$0.3$). While we report $c_{\mathrm{q}} \approx 0.2$ as a representative value capturing the typical subdiffusive behavior in log-likelihood space, a more detailed analysis of model-size dependence is beyond the scope of this paper.

Interpretation via loss landscape and flat minima. In our experimental setting, texts are sampled from a corpus that closely approximates the training data. Consequently, each component of the log-likelihood vector corresponds to a term contributing to the cross-entropy loss, and the negative average of these components serves as a reasonable empirical estimate of the expected loss. The marked contrast between the near-Brownian diffusion in weight space and the strongly suppressed diffusion in log-likelihood space indicates that, even as model weights continue to change, the output probability distributions remain relatively stable. This observation is consistent with the hypothesis that stochastic gradient descent tends to favor flat regions of the loss landscape (Keskar et al., 2017).

[6] Gneiting et al. (2012) consider a real-valued stochastic process $X(t)$, $t \in \mathbb{R}^d$ and estimate c from finite-scale increment statistics, where the diffusion exponent c is defined via the mean squared displacement scaling $\mathbb{E}(|X(t)-X(t_0)|^2) \propto |t-t_0|^c$. They then relate it to the fractal dimension of the graph $\{(t, X(t))\}$ via $D_{\mathrm{graph}} = d + 1 - c/2$. This differs from the effective dimension interpretation used for our one-parameter training trajectories, for which $d = 1$ and hence $D = 2/c$.

	Median	Mean	SD
Later	0.011	0.014	0.0050
Early	0.067	0.078	0.047
Seed	0.12	0.12	0.0071
Size	0.48	0.61	0.47
8-bit	0.44	0.80	1.9
4-bit	0.49	0.91	2.0
FT	0.40	1.5	3.6
Intra-type	0.95	2.8	6.4
Random	2.2	5.3	8.8
Layer	3.0	8.1	13

(a) Summary statistics (median, mean, SD) of KL divergence across various settings. This table corresponds to Fig. 3.

	between checkpoints			between seeds	between model sizes			
	10k→11k	50k→51k	100k→101k	avg.	1B	1.4B	2.8B	6.9B
410M	0.069 (±0.0011)	0.053 (±0.0010)	0.023 (±0.00042)	0.11 (± 0.0024)	0.28 (±0.010)	0.52 (±0.018)	1.0 (±0.036)	1.7 (±0.060)

(b) KL divergence for Pythia 410M, measured between checkpoints saved during pretraining, across different random seeds, and across different model sizes. For the inter-seed and inter-size comparisons, the final checkpoints are used.

Model	1→2	16→17	31→32
Llama-2-7b-hf	0.61 (±0.013)	1.6 (±0.063)	1.3 (±0.10)
Meta-Llama-3-8B	4.0 (±0.075)	22 (±0.78)	1.7 (±0.037)
Mistral-7B-v0.3	2.4 (±0.10)	5.7 (±0.12)	8.6 (±0.54)

(c) KL divergence between adjacent layers within specific models, measured using the logit lens.

Table 2: KL divergence in bits per byte between language models under various conditions. Values in parentheses represent standard errors. See Appendix C for the formula used to compute the standard errors.

Geometric interpretation via folding. This optimization-based view is closely related to a more structural interpretation arising from the intrinsic redundancies of multilayer neural networks. Due to permutation symmetries of hidden units, many distinct weight configurations correspond to the same function or output distribution (Fukumizu and Amari, 2000). From a statistical viewpoint, such permutation-induced redundancies are associated with degenerate directions of the Fisher information matrix, giving rise to flat directions in the loss landscape. As a result, trajectories that are well separated in weight space can be mapped to nearby or identical points in the log-likelihood space. Viewed through this many-to-one mapping, training trajectories in the log-likelihood space are strongly folded, leading to complex geometric structures that remain confined to a relatively narrow region of the space.

Quantifying folding via Hölder regularity. The geometric interpretation based on folding can be further quantified by considering the regularity of the mapping $f : \boldsymbol{W} \mapsto \boldsymbol{q}(\boldsymbol{W})$ from weight space to log-likelihood space. Specifically, recall that a mapping is α-Hölder continuous if distances in the output space are bounded by a constant times the α-th power of distances in the input space, with $\alpha = 1$ corresponding to Lipschitz continuity and smaller values of $\alpha > 0$ indicating increasing non-smoothness. If f is α-Hölder continuous, then the fractal dimensions of the corresponding trajectories obey

$$D_{\mathrm{q}} \leq \frac{D_{\mathrm{w}}}{\alpha}$$

(Falconer, 2014). If f additionally satisfies the non-degeneracy condition required to saturate this bound, then the corresponding effective Hölder exponent along the training trajectory is

$$\alpha = \frac{D_{\mathrm{w}}}{D_{\mathrm{q}}} = \frac{c_{\mathrm{q}}}{c_{\mathrm{w}}}.$$

This relation shows that as α decreases, the mapping can strongly fold the weight-space trajectory into the log-likelihood space. As a result, the resulting trajectory becomes geometrically more complex and has a higher effective fractal dimension, while its large-scale displacement remains strongly suppressed. In our empirical setting, the observed diffusion exponents imply effective dimensions $D_{\mathrm{w}} \approx 2$ and $D_{\mathrm{q}} \approx 10$, which in turn yield an effective Hölder exponent of $\alpha \approx 0.2$.

Interpretation of the effective Hölder exponent. The effective Hölder exponent $\alpha = c_{\mathrm{q}}/c_{\mathrm{w}}$, computed from the trajectory starting at t_0, is used as a trajectory-based proxy for the pointwise Hölder exponent $\alpha(\boldsymbol{W}_{t_0})$ at $\boldsymbol{W}_{t_0}$, in the sense that it reflects the effective roughness of f along the observed training trajectory rather than the actual pointwise regularity defined using all nearby perturbation directions around $\boldsymbol{W}_{t_0}$[7]. Estimating the local regularity of f at $\boldsymbol{W}_{t_0}$ along a fixed direction $\boldsymbol{\delta}$ from the local scaling $\|f(\boldsymbol{W}_{t_0} + \epsilon\boldsymbol{\delta}) - f(\boldsymbol{W}_{t_0})\| \propto |\epsilon|^{\alpha(\boldsymbol{W}_{t_0}, \boldsymbol{\delta})}$, where $\epsilon \in \mathbb{R}$, can in principle yield an even larger directional Hölder exponent $\alpha(\boldsymbol{W}_{t_0}, \boldsymbol{\delta})$ than the trajectory-based estimate reported here. This does not contradict the present interpretation, because Hölder regularity need not be isotropic, and the pointwise Hölder exponent captures the roughest local behavior, in the sense that $\alpha(\boldsymbol{W}_{t_0}) \leq \alpha(\boldsymbol{W}_{t_0}, \boldsymbol{\delta})$ for any nonzero $\boldsymbol{\delta}$. Even when $\boldsymbol{\delta} = \boldsymbol{W}_{t_1} - \boldsymbol{W}_{t_0}$, $\alpha(\boldsymbol{W}_{t_0}, \boldsymbol{\delta})$ can still be larger than the trajectory-based estimate.

A perspective from embedding theory. In our analysis, we visualize trajectories in the N-dimensional log-likelihood space $\mathbb{R}^N$ in $n = 2$ or $n = 3$ dimensions via t-SNE. This approach naturally raises the question of when such low-dimensional representations remain one-to-one and when self-intersections become unavoidable. According to the theory of *embedology* established by Sauer et al. (1991), for a compact set $A \subset \mathbb{R}^N$ with box-counting dimension[8] d, almost every C^1 map $F : \mathbb{R}^N \to \mathbb{R}^n$ is one-to-one on A provided $n > 2d$. This property is established in the sense of *prevalence* (i.e., a probability-one notion under finite-dimensional perturbations of the map). Conversely, for $n \leq 2d$ and for each fixed $\delta > 0$, the δ-distant self-intersection set has a lower box-counting dimension of at most $2d - n$.

For the practically important case $n = 2$, we have $2d - n = 2d - 2 \geq 0$ for any $d \geq 1$. Thus, generic injectivity is no longer guaranteed, and self-intersections cannot in general be ruled out. In particular, when $d = 1$, the self-intersection set has dimension at most 0, corresponding heuristically to isolated intersection points. For $n = 3$, we have $2d - n = 2d - 3$. Hence, if $d \geq 1.5$, self-intersections may still remain, whereas if $d < 1.5$, the condition $n > 2d$ holds, and the set can generically be represented in three dimensions by a one-to-one map without self-intersections.

We return to this viewpoint in Appendix D.2, where we discuss its heuristic implications for the two- and three-dimensional t-SNE visualizations of the Pythia pretraining trajectories.

A related perspective on Hölder regularity. Ly and Gong (2025) also relate anomalous diffusion during training to Hölder regularity. In their framework, the Hölder exponent is introduced through the loss landscape, which in our setting corresponds not to the full log-likelihood vector but to its one-dimensional component along the $\mathbf{1}$-direction, i.e., to the mean log-likelihood up to a constant scaling. From this perspective, their analysis may be viewed as emphasizing a one-dimensional aspect of the geometry induced by the log-likelihood vector. Our analysis extends this viewpoint by considering training trajectories in the full log-likelihood space, rather than only the averaged direction. This makes it possible to compare the scaling behavior in weight space and in log-likelihood space, and thereby to quantify how the mapping from weights to model behavior contracts and folds trajectories. Another difference lies in the notion of dimension being considered. Ly and Gong (2025) discuss the fractal dimension of the graph of the loss landscape[9], which in our setup corresponds to $\{(\boldsymbol{W}, \bar{\ell}(\boldsymbol{W}))\}$. By contrast, our D_{w} and D_{q} are the effective fractal dimensions of the trajectories themselves, namely $\{\boldsymbol{W}_t\}$ in weight space and $\{\boldsymbol{q}_t\}$ in log-likelihood space, respectively. Thus, while both studies relate training dynamics to Hölder-type regularity and anomalous diffusion, our formulation adds a geometric comparison between parameter space and behavior space, providing a complementary perspective on how training trajectories are transformed when viewed through model behavior.

An illustrative example of folding. Although the mapping $f : \boldsymbol{W} \mapsto \boldsymbol{q}(\boldsymbol{W})$ is explicitly defined once the neural network architecture and the eval-

[7] Along the training trajectory starting at t_0, we observe the power-law relations $\|\boldsymbol{W}_t - \boldsymbol{W}_{t_0}\|^2 \propto |t - t_0|^{c_{\mathrm{w}}}$ and $\|f(\boldsymbol{W}_t) - f(\boldsymbol{W}_{t_0})\|^2 \propto |t - t_0|^{c_{\mathrm{q}}}$. Eliminating $|t - t_0|$ from these two relations yields $\|f(\boldsymbol{W}_t) - f(\boldsymbol{W}_{t_0})\| \propto \|\boldsymbol{W}_t - \boldsymbol{W}_{t_0}\|^{c_{\mathrm{q}}/c_{\mathrm{w}}}$. This heuristically motivates interpreting $c_{\mathrm{q}}/c_{\mathrm{w}}$ as a trajectory-based effective Hölder exponent of f at $\boldsymbol{W}_{t_0}$.

[8] Although the Hausdorff dimension and box-counting dimension can differ in a strict mathematical sense, they are expected to be equal in the context of the dynamical systems analyzed here. Thus, we use them interchangeably throughout this heuristic analysis.

[9] In their fractional Brownian surface model, the fractal dimension is $D_{\bar{\ell}} = \dim \boldsymbol{W} + 1 - \alpha_{\bar{\ell}}$ with $\alpha_{\bar{\ell}}$ being the Hölder exponent of the loss landscape.

uation text set are fixed, its geometric regularity and folding structure are not directly accessible in closed form.

To illustrate this folding mechanism more concretely, we consider an example based on generalized Takagi–Landsberg functions. These functions are defined as multiscale summations of sawtooth functions $S(x) = \mathrm{dist}(x, \mathbb{Z})$:

$$f(\boldsymbol{W}) = \sum_{k=1}^{\infty} \mathbf{a}_k \lambda^{-k\alpha} S\left(\lambda^k \mathbf{b}_k^\top \boldsymbol{W}\right),$$

where $\lambda > 1$ is a constant and $\mathbf{a}_k$ and $\mathbf{b}_k$ are vectors with the same dimensions as $\boldsymbol{q}$ and $\boldsymbol{W}$, respectively. This class of functions is known to be α-Hölder continuous for any $0 < \alpha < 1$ under mild boundedness and non-degeneracy conditions on $\{\mathbf{a}_k, \mathbf{b}_k\}$ (Shidfar and Sabetfakhri, 1986; Allaart and Kawamura, 2011/2012). Such constructions provide a simple geometric illustration of how the many-to-one and redundant parameterization of deep neural networks can give rise to highly folded trajectories in the log-likelihood space. In contrast to this construction, previous studies have focused on folding phenomena in input–output mappings of deep networks, where depth induces increasingly fine sawtooth partitions of the input space (Telgarsky, 2016, 2017).

C On the Estimation of the KL Divergence

Interpretation of KL divergence via text entropy. As a reference for interpreting KL divergence values, we estimate the entropy of the text from log-likelihood statistics, as described below. In our setting, this estimated text entropy is approximately 0.71 bits/byte; for example, a KL divergence of 0.1 bits/byte corresponds to a value slightly over 10% of the text entropy.

In this work, we use the minimum value of the negative mean log-likelihood across models as an estimate of the text entropy. This is because, given a model p_i, a text set $\{x_1, \ldots, x_N\}$, the log-likelihood $\ell_i(x_s)$, and the true data-generating distribution p_0, the following approximation holds:

$$\begin{aligned} -\frac{1}{N}\sum_{s=1}^{N} \ell_i(x_s) &\approx \mathbb{E}_{x\sim p_0}[-\log p_i(x)] \\ &= H(p_0) + \mathrm{KL}(p_0, p_i) \\ &\geq H(p_0), \end{aligned}$$

where $H(p_0) = \mathbb{E}_{x\sim p_0}[-\log p_0(x)]$ is the text entropy. Therefore, the minimum of the negative mean log-likelihood across models provides an empirical upper bound on the text entropy $H(p_0)$ and may serve as a rough approximation when at least one model is close to p_0.

Standard error of the estimated KL divergence. Let $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ denote the $\boldsymbol{q}$ coordinates of models p_i and p_j, respectively. While in the main text of this paper we use $\boldsymbol{q}$ coordinates rescaled by $1/\sqrt{2N\bar{B}\log 2}$ so that squared Euclidean distances approximate KL divergence in bits per byte, we do not apply this rescaling in this appendix. Accordingly, the KL divergence between the models is estimated as $\mathrm{KL}(p_i, p_j) \approx \frac{1}{2N}\|\boldsymbol{q}_i - \boldsymbol{q}_j\|^2 = \frac{1}{2N}\sum_{k=1}^{N}(Q_{ik} - Q_{jk})^2$, which corresponds to the sample mean of $\{(Q_{ik} - Q_{jk})^2/2\}_{k=1}^{N}$. Assuming that the effects of double-centering in $\boldsymbol{Q}$ are negligible, these terms can be treated as approximately independent. When N is sufficiently large, the variance of the estimated KL divergence is given by

$$\begin{aligned} &\mathrm{Var}(\mathrm{KL}(p_i, p_j)) \\ &\approx \frac{1}{N}\left\{\frac{1}{4N}\sum_{k=1}^{N}(Q_{ik} - Q_{jk})^4 \right. \\ &\qquad \left. -\frac{1}{4N^2}\|\boldsymbol{q}_i - \boldsymbol{q}_j\|^4\right\} \\ &\approx \frac{1}{4N^2}\sum_{k=1}^{N}(Q_{ik} - Q_{jk})^4 - \frac{1}{N}\mathrm{KL}(p_i, p_j)^2. \end{aligned}$$

Therefore, the standard error of the estimator $\mathrm{KL}(p_i, p_j)$ can be approximated as

$$\begin{aligned} &\mathrm{SE}(\mathrm{KL}(p_i, p_j)) \\ &\approx \sqrt{\frac{1}{4N^2}\sum_{k=1}^{N}(Q_{ik} - Q_{jk})^4 - \frac{1}{N}\mathrm{KL}(p_i, p_j)^2}. \end{aligned}$$

The standard error estimated in nats above is converted to bits per byte by dividing by $\bar{B}\log 2$. Equivalently, when using $\boldsymbol{q}$ coordinates rescaled by $1/\sqrt{2N\bar{B}\log 2}$ as in the main text, the standard error of KL divergence in bits per byte is given by

$$\begin{aligned} &\mathrm{SE}(\widetilde{\mathrm{KL}}(p_i, p_j)) \\ &\approx \sqrt{\sum_{k=1}^{N}(\tilde{Q}_{ik} - \tilde{Q}_{jk})^4 - \frac{1}{N}\widetilde{\mathrm{KL}}(p_i, p_j)^2}, \end{aligned}$$

where $\tilde{Q} = Q/\sqrt{2N\bar{B}\log 2}$ and

$$\widetilde{\mathrm{KL}}(p_i, p_j) = \frac{\mathrm{KL}(p_i, p_j)}{\bar{B}\log 2} = \sum_{k=1}^{N} (\tilde{Q}_{ik} - \tilde{Q}_{jk})^2.$$

On the robustness to the sampling of texts. As an additional experiment, we computed the log-likelihood vectors using a new set of 10,000 texts randomly sampled from the 5,703,791 texts in a subset of the Pile[10], using the Pythia-410M model. We then computed the pairwise KL divergence between all checkpoints after warmup, and found that the Pearson correlation coefficient between the KL values obtained using the original text set and those obtained using the new text set was 0.99. This indicates that the impact of text sampling on the KL values is limited.

One advantage of using log-likelihood vectors is that the resolution for each domain can be adjusted by the amount of text from that domain included in the dataset. If a specific domain is fixed, it effectively corresponds to projecting the trajectory onto a certain aspect, and thus the shape of the trajectory is expected to change.

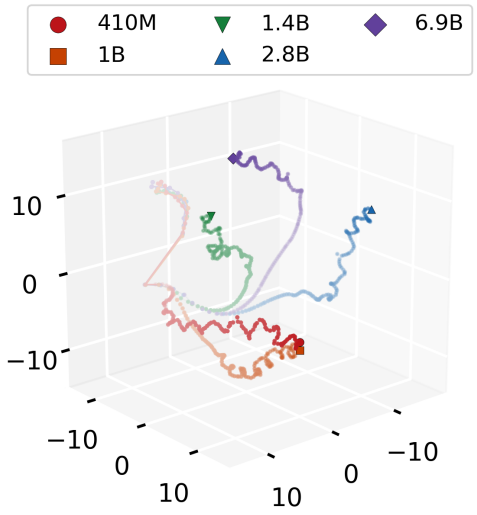
D Additional Visual Analyses of Pretraining Trajectories

This appendix provides supplementary visual analyses and interpretation of pretraining trajectories. Appendix D.1 presents three-dimensional t-SNE visualizations in the log-likelihood space, and Appendix D.2 discusses their embedding-theoretic interpretation, including possible sources of apparent discontinuities and oscillatory artifacts. Appendix D.3 then uses PCA to examine the global structure of the trajectories in the log-likelihood space. In contrast, Appendix D.4 presents trajectory visualizations in the weight space, highlighting fundamental limitations of joint visualization across models.

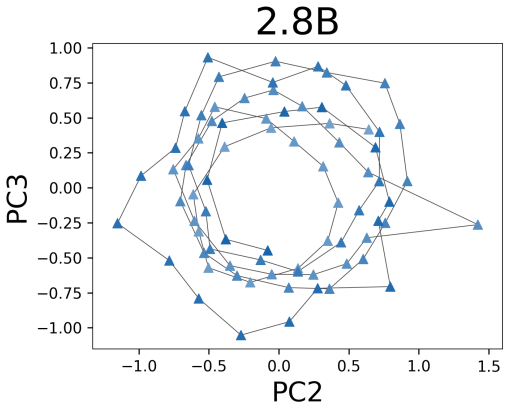
D.1 Three-Dimensional t-SNE Visualizations

This subsection presents three-dimensional t-SNE visualizations of the pretraining trajectories discussed in Section 4. In the main text, Fig. 1 shows two-dimensional t-SNE visualizations of pretraining trajectories for Pythia models across multiple model sizes and random seeds. In two dimensions, apparent discontinuities may arise near regions

[10]The full Pile corpus, when segmented into 1,024-byte text chunks, yields a total of 5,703,791 texts (Oyama et al., 2025).



(a) Visualization of pretraining trajectories for five Pythia model sizes using 3D t-SNE (perplexity = 30). This figure corresponds to a three-dimensional version of Fig. 1, excluding additional random seeds of the 410M model.



(b) PC2 versus PC3 for the later portion of the Pythia-2.8B training trajectory after applying 3D t-SNE.

Figure 6: Pythia training trajectories visualized using 3D t-SNE.

where the images of different trajectories overlap. This phenomenon should be understood as a limitation of low-dimensional visualization rather than as evidence of discontinuity of the underlying trajectories. A heuristic interpretation of this phenomenon from embedding theory is given in Appendix D.2.

As shown in Fig. 6, this issue is alleviated in the three-dimensional visualization. Since the visualization is based on finitely many checkpoints sampled along training, the curves shown here should be understood as coarse approximations to the underlying continuous trajectories. In the 3D t-SNE representation, the sampled training trajectories appear as continuous curves without obvious jumps or self-intersections. For clarity, we exclude additional seeds of the Pythia-410M model and visual-

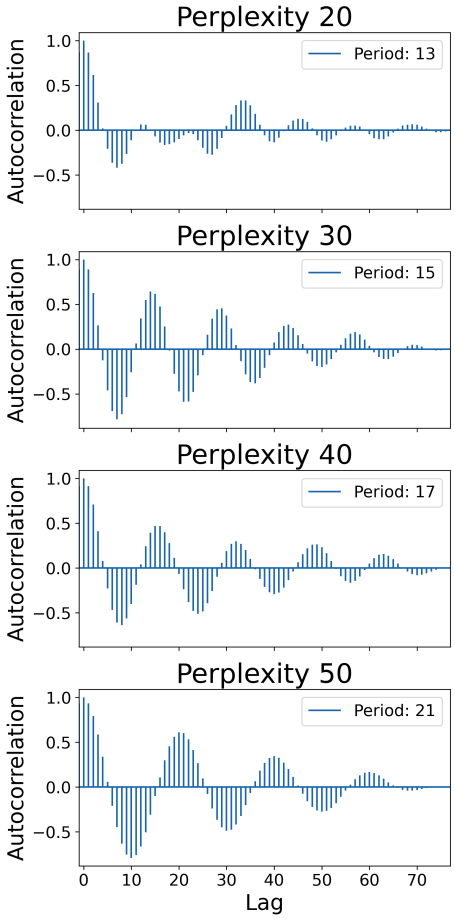


Figure 7: Autocorrelation functions of PC3 for the later portion of the Pythia-2.8B training trajectory (from step 67k onward) after applying 3D t-SNE, shown for different perplexity values.

ize the trajectories of the five model sizes simultaneously in three dimensions, as shown in Fig. 6a.

Figure 6b further visualizes the later portion of the Pythia-2.8B trajectory by plotting PC2 versus PC3 after applying 3D t-SNE. In this representation, the trajectory appears to exhibit a spiral-like structure. To analyze this phenomenon, Fig. 7 shows the autocorrelation function of PC3 for the later portion of the trajectory, computed under different t-SNE perplexity settings (20, 30, 40, and 50). The spiral period is estimated as the training step at which the autocorrelation function attains its maximum between the second and third zero-crossings. As the perplexity increases, the estimated period also increases, indicating that the apparent spiral structure depends on the neighborhood size used in the embedding.

This spiral-like appearance should therefore be interpreted primarily as a visualization artifact rather than as an intrinsic property of the training dynamics. At the same time, its emergence may reflect the difficulty of representing a highly folded trajectory in only three dimensions, which is consistent with the subdiffusive scaling $\|q_t - q_{t_0}\|^2 \propto |t - t_0|^{c_q}$ and the large effective fractal dimension inferred from it.

D.2 Embedding-Theoretic Interpretation of t-SNE Trajectories

We now apply the embedding-theoretic viewpoint discussed in Appendix B to the pretraining trajectories of Pythia. The actual two- and three-dimensional t-SNE visualizations are given in Section 4.1 and Appendix D.1, respectively. If the continuous trajectory in log-likelihood space is interpreted heuristically as a fractal-like compact set with effective dimension $d \approx D_q$, then our estimate $D_q \approx 10$ places it well within the regime $d \geq 1.5$. From this perspective, even a three-dimensional representation is generally not expected to remain one-to-one, and residual folding or self-overlap is therefore not surprising. The oscillatory artifacts observed in Fig. 6 may reflect this geometric difficulty of representing a highly folded trajectory in only three dimensions.

At the same time, the actual visualization is constructed from finitely many checkpoints sampled along training. More generally, fractal or roughness-related quantities are defined through limiting behavior at small scales, whereas actual observations are available only over a finite range of scales (Gneiting et al., 2012). When these sampled points are connected in temporal order, they form a polygonal chain, which is a one-dimensional object in a coarse geometric sense. Under this approximation, three dimensions become the first case in which a generic one-to-one representation is possible, whereas two dimensions remain a critical case in which apparent overlaps may persist. Although this argument remains heuristic, since t-SNE is not itself a generic C^1 map of the type assumed in the embedding theorem and the sampled trajectories are only finite approximations to the underlying continuous path, it is nevertheless consistent with our empirical observations: in the three-dimensional t-SNE visualization, no obvious self-intersections are observed among the sampled trajectories, whereas in the two-dimensional visualization of Fig. 1, apparent overlaps can manifest themselves as trajectory jumps.

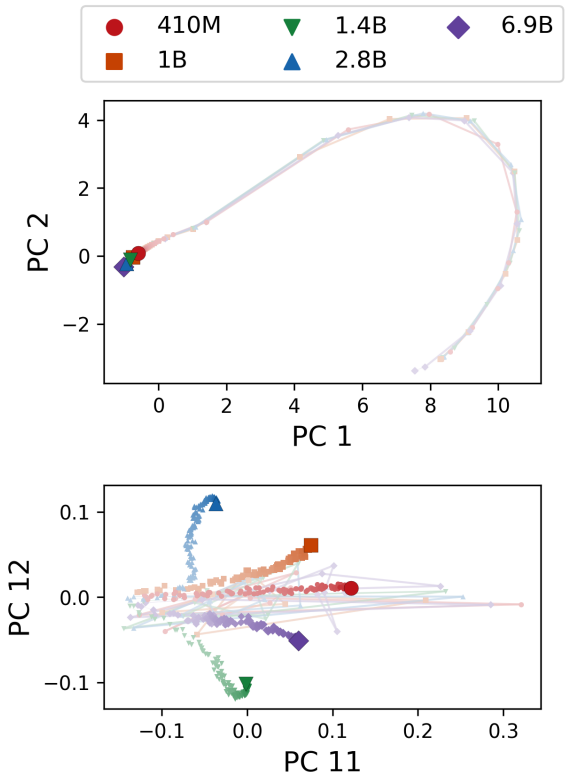


Figure 8: PCA visualization of the training trajectories of Pythia models. (Top) PC1 vs. PC2. (Bottom) PC11 vs. PC12.

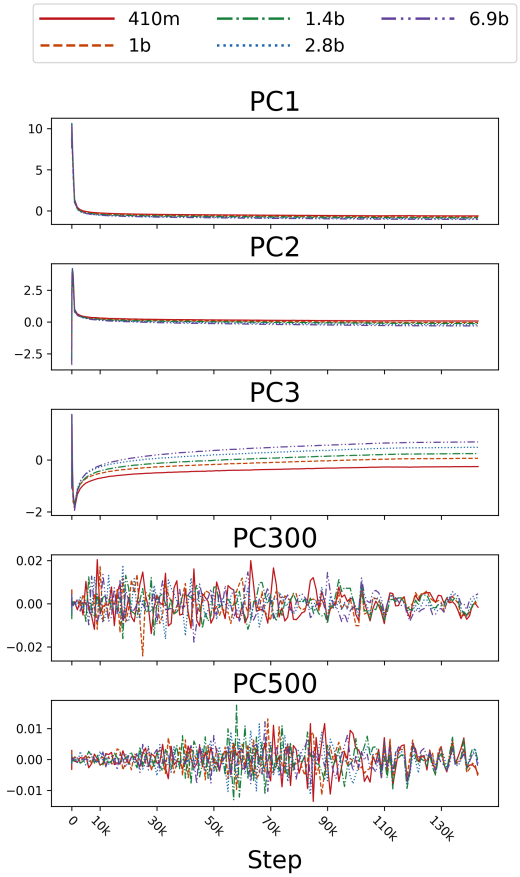


Figure 9: Variation of principal component values from PCA applied to the trajectories of all Pythia model sizes.

D.3 PCA-Based Visualizations of Pretraining Trajectories

We further analyze the same pretraining trajectories using principal component analysis (PCA) to highlight their global geometric structure. In contrast to t-SNE, which emphasizes local neighborhood relationships, PCA captures large-scale variations shared across different model sizes.

Fig. 8 shows the result of applying PCA simultaneously to all model sizes and plotting their trajectories. Fig. 9 further illustrates the evolution of multiple principal component values along the training steps. Unlike t-SNE, the top principal components exhibit similar values across model sizes, resulting in overlapping trajectories. This difference likely arises from the fact that the top principal components capture global variation in the trajectories, whereas t-SNE emphasizes local structure. In contrast, the lower principal components exhibit more noise-like behavior. As shown in Table 3, the KL divergence between different model sizes at the final checkpoint is larger than that between consecutive saved checkpoints. This is consistent with the observation that trajectories are separated in the lower components.

D.4 Limitations of Trajectory Visualization in Weight Space

Since pretraining trajectories in the weight space do not admit a joint visualization across different model sizes and random seeds, we visualize them separately for each model. For each model size, we computed a distance matrix containing the pairwise Euclidean distances between the weight vectors of all checkpoints. Figure 10 shows the trajectories from step 100k onward, visualized using t-SNE[11]. The trajectories were centered after t-SNE to align the plot ranges. In addition, for the 1B model, the trajectory was rotated by 180 degrees to roughly align its direction of progression with those of the other model sizes.

Despite these adjustments, the resulting visualizations remain difficult to interpret. Because each model requires a separate embedding, the relative positions and orientations of trajectories across models are arbitrary and cannot be meaningfully

[11] t-SNE can take a distance matrix as input.

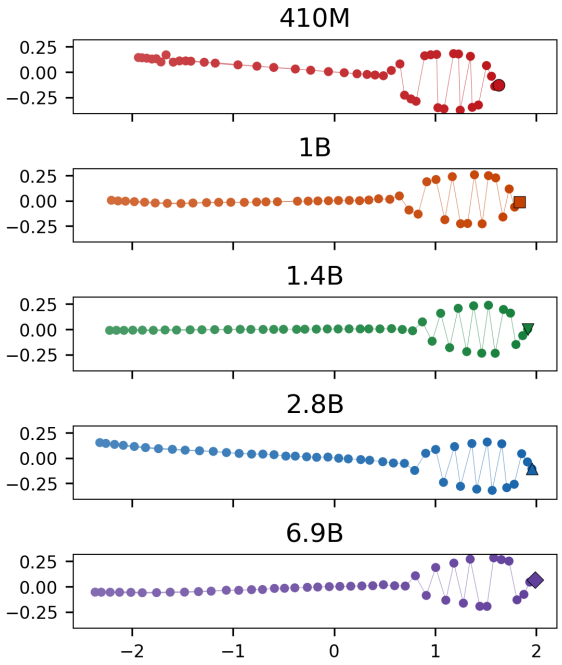


Figure 10: t-SNE visualization (perplexity=30) of the trajectories based on pairwise distances in the weight space for Pythia models after step 100k. To align plot ranges across model sizes, the trajectories were centered after t-SNE. They were also rotated to roughly align the directions of progression. Final checkpoints are indicated using distinct markers.

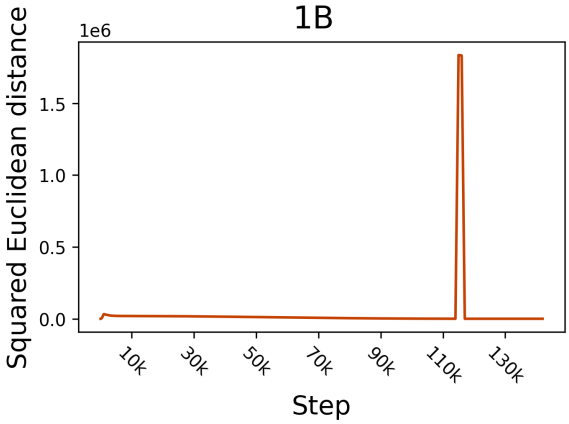


Figure 11: Squared Euclidean distance between the weights of consecutive saved checkpoints for Pythia 1B. The distances between 115k and 116k, and between 116k and 117k are abnormally large.

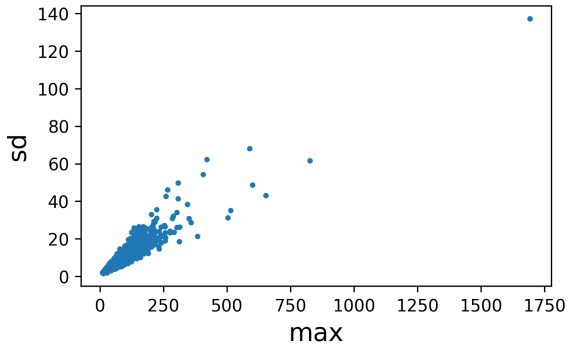


Figure 12: The relationship between the maximum and the standard deviation of the absolute differences in log-likelihoods between consecutive saved checkpoints of the same model size, computed for each text.

compared. This highlights a fundamental limitation of weight-space representations for trajectory visualization, in contrast to the log-likelihood space, which provides a shared coordinate system for joint analysis.

E Details of Sections 3.1 and 4

To analyze the trajectories of language models during pretraining, we used Pythia models[12] of various sizes, as listed in Table 5 in Appendix I. For each checkpoint, we computed log-likelihood vectors using the same set of 10,000 texts as in Oyama et al. (2025). The computations were performed on an NVIDIA RTX 6000 Ada and took approximately 10 minutes for a 7B model loaded in float16 precision.

E.1 Preprocessing

Assuming that the KL divergence between consecutive saved checkpoints changes continuously, we regarded certain checkpoints and texts as outliers. All experiments were conducted after removing these outliers. This appendix provides a detailed description of the preprocessing procedure.

Removing outlier models. As shown in Fig. 11, the Euclidean distance between weights near step 116k of Pythia 1B is abnormally large. We suspect that this checkpoint may not have been saved properly and therefore excluded it from our analysis. A question about this checkpoint was also raised on the Discussions page of the Pythia 1B model on Hugging Face[13].

Removing outlier texts. Following Oyama et al. (2025), we clip the log-likelihood matrix $\boldsymbol{L}$ at the bottom 2%, after excluding the outlier model. Next, we consider the log-likelihood difference for each text between consecutive saved check-

[12] Released under the Apache 2.0 License.

[13] https://huggingface.co/EleutherAI/pythia-1b/discussions/4

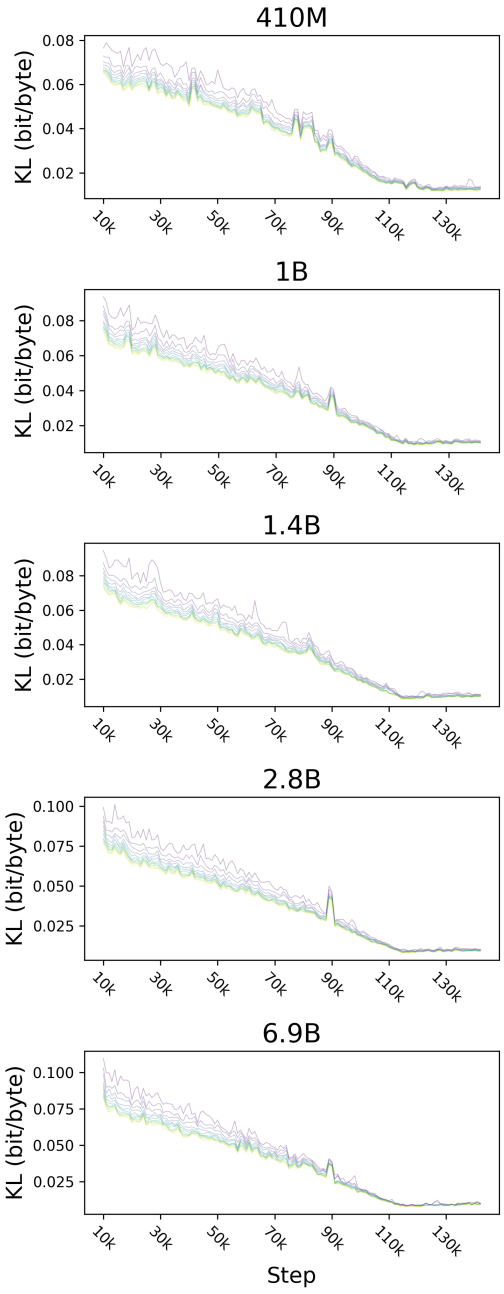


Figure 13: Change in KL divergence after step 10k for different numbers of top outlier texts removed (10, 100, 200, ..., 1000). Brighter colors indicate larger numbers of removed texts.

points after the warmup phase. For each text, we define its outlier score as the maximum absolute difference across all model sizes and training steps. Specifically, letting $p_{i,t}$ denote the model of size i at step t, the outlier score for a text x is given by $\max_{i,t} \{|\log p_{i,t+1}(x) - \log p_{i,t}(x)|\}$. It is also possible to assign scores to texts using the standard deviation of these differences. However, Fig. 12 shows that the maximum and the standard deviation are highly correlated. Therefore, we adopt the maximum value for scoring. The figure further confirms the presence of texts that clearly behave as outliers. For example, the text with the highest outlier score was a meaningless and repetitive sequence of symbols, such as `028a28a0028a28a0028a28a...`.

Fig. 13 shows the KL divergence between consecutive saved checkpoints after step 10k[14], shown for different numbers of top outlier-scored texts removed (top 10, 100, 200, ..., 1000). When the top 300 texts (3%) are removed, the variation in KL divergence stabilizes with respect to the number of removed texts. Therefore, we excluded these 300 texts from our experiments.

	410M	1B	1.4B	2.8B	6.9B
410M	0.00	0.28	0.52	1.04	1.7
1B	0.28	0.00	0.15	0.44	0.92
1.4B	0.52	0.15	0.00	0.21	0.57
2.8B	1.0	0.44	0.21	0.00	0.21
6.9B	1.7	0.92	0.57	0.21	0.00

Table 3: KL divergence (bits/byte) between different model sizes at the final checkpoint of Pythia.

E.2 KL Divergence between Checkpoints

During pretraining, the learning rate changes dynamically with the training step. In particular, the first 1,430 steps correspond to the warmup phase, during which the learning rate increases linearly. As a result, as shown in the left panel of Fig. 4, the KL divergence between the checkpoints takes on relatively large values, mostly exceeding 10 bits/byte during this interval. The right panel of Fig. 4 shows the variation in KL divergence between consecutive saved checkpoints for all model sizes. In this figure, the value at step 10k, for example, represents the KL divergence between the checkpoints at steps 10k and 11k. Additionally, Fig. 14 plots the KL divergence between consecutive saved checkpoints after step 10k for Pythia, along with the corresponding standard error. The derivation of the standard error is described in Appendix C. Finally, Table 3 presents the pairwise KL divergence between model sizes at the final checkpoint. As discussed in Section 4, the divergence caused by differences in model size is larger than that resulting from 1k steps of training.

E.3 Estimation of Diffusion Exponents

In this subsection, we describe the procedure for estimating diffusion exponents in both the log-

[14]The maximum is computed from step 2k onward, after the warmup phase, but we visualize from step 10k to better capture the variation in KL divergence.

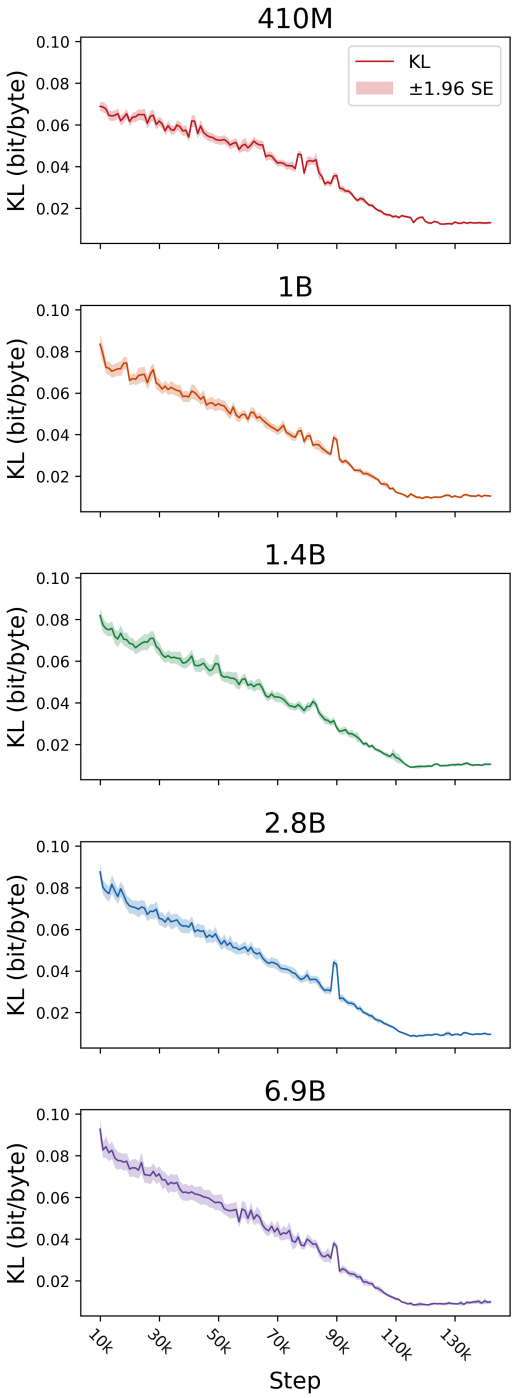


Figure 14: KL divergence and its standard error between consecutive saved checkpoints of Pythia after step 10k.

likelihood and weight spaces. The left panel of Fig. 5 shows the evolution of squared Euclidean distances in weights and log-likelihood vectors over 10k steps starting from step 120k. The right panel of Fig. 5 illustrates how the estimated diffusion exponent changes when varying the starting step for the calculation. To ensure stability, we vary the starting point only after the warmup phase. The exponent is estimated using linear regression (least

Size	Log-likelihood	Weight
410M	0.849 (±0.158)	0.993 (±0.00622)
1B	0.935 (± 0.0866)	0.999 (±0.000922)
1.4B	0.955 (± 0.0336)	0.999 (±0.000687)
2.8B	0.959 (± 0.088)	0.999 (±0.000704)
6.9B	0.977 (± 0.0305)	0.999 (±0.000975)

Table 4: The mean and standard deviation of the coefficient of determination (R^2) when varying the starting step used to compute the diffusion exponent.

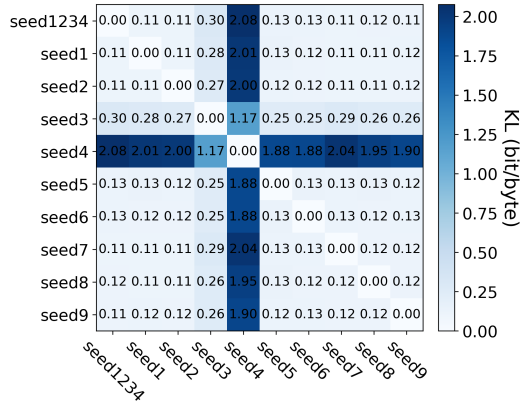


Figure 15: KL divergence between the final checkpoints of Pythia 410M models with different seeds.

squares) over a 10k-step window beginning at each initial step. Summary statistics of the coefficient of determination (R^2) for each regression are provided in Table 4.

E.4 Different Seeds

van der Wal et al. (2025) trained Pythia models with nine different seeds and released checkpoints at the same steps as in the original Pythia suite. The random seed affects the initialization of weights and the order of the training data. We used all available checkpoints for the 410M models with seeds 1 through 9. However, at the time of writing this paper, the weights for (seed, step) = (2, 111k), (6, 88k), and (9, 40k) were not available on Hugging Face and were therefore excluded from our analysis[15]. The original 410M model was trained with seed 1234.

Fig. 15 shows the KL divergence between the final checkpoints of models trained with different seeds. Despite sharing the same architecture and data, the KL divergence between seed 3 and the

[15]For example, see https://huggingface.co/EleutherAI/pythia-410m-seed2/tree/step111000.

other seeds, as well as between seed 4 and the others, are significantly larger. According to van der Wal et al. (2025), models trained with seed 3 and seed 4 experienced loss spikes during training and are considered outliers in terms of performance. This suggests that KL divergence is also effective in identifying such outlier models. Therefore, seed 3 and seed 4 are excluded when generating Fig. 1. We also report in Table 2b the average KL divergence between the final checkpoint of the original 410M model and those of seeds 1, 2, 5, 6, 7, 8, and 9.

F Details of Section 3.2

Here, we describe the language models used in Section 3.2. We selected the 50 most downloaded models from those analyzed in Oyama et al. (2025). A complete list of the models is provided in Table 7 in Appendix I.

For the quantization methods, we applied 8-bit (Dettmers et al., 2022) and 4-bit (Dettmers et al., 2023) quantization using `bitsandbytes`[16].

G Details of Section 3.3

We describe the language models used in Section 3.3. From the 1,018 models analyzed in Oyama et al. (2025), we identified fine-tuned models as those whose base model is specified in the metadata available via the Hugging Face Hub API[17]. Our analysis was conducted on pairs of fine-tuned models and their base models. Models associated with multiple base models (i.e., merged models) were excluded. A list of the language models used in this analysis is provided in Table 6 in Appendix I. In Fig. 2b, there are a few pairs connected by line segments that span across different colors, namely different model types. This occurs because, under the model type classification method used in Oyama et al. (2025), some fine-tuned models were categorized as belonging to the "others" type.

Summary statistics for these distributions are reported in Table 2a. Fig. 3 shows that KL divergence after fine-tuning tends to be slightly smaller, as indicated by the left-skewed distribution. As noted in Oyama et al. (2025), models of the same type tend to have smaller KL divergence. To more accurately assess the effect of fine-tuning, we also analyze random model pairs sampled within the same model type.

[16] https://github.com/bitsandbytes-foundation/bitsandbytes

[17] https://github.com/huggingface/huggingface_hub

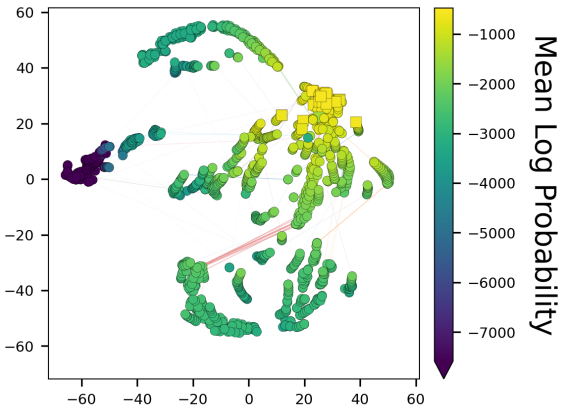


Figure 16: Re-rendering of the top panel of Fig. 2c. Colors indicate mean log-likelihood, clipped at the bottom 5% across all values to improve visibility.

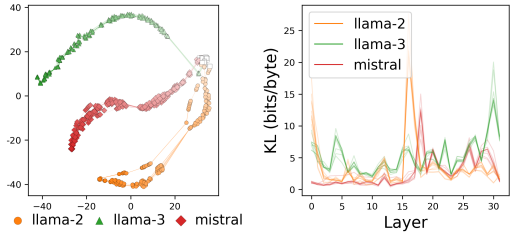


Figure 17: For 10 models per model type: (left) t-SNE visualization of trajectories across layers, where the final layer is indicated by a white square; (right) KL divergence between adjacent layers (corresponding to Table 2c).

H Details of Section 3.4

Here, we describe the language models used in Section 3.4. For the trajectories shown in Fig. 2c and Fig. 16, we selected the 50 most downloaded models from those analyzed in Oyama et al. (2025). A complete list of the models is provided in Table 7 in Appendix I.

Fig. 16 shows the trajectories of models across layers, color-coded by mean log-likelihood, depending on layer depth. The mean log-likelihood increases as the layer depth increases. This indicates that the direction of trajectory progression corresponds to deeper layers, that is, to increasing mean log-likelihood.

Additionally, for detailed analysis, we selected models with 32 layers from the llama-2, llama-3, and mistral types. For each of these types, we used the 10 most downloaded models. That is, the top 50 do not include 10 models for each of

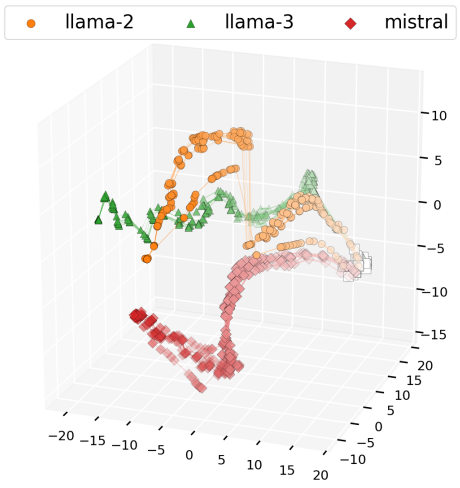


Figure 18: 3D t-SNE visualization (perplexity=30) of the left panel of Fig. 17. Deeper layers are represented with lighter colors, and the final layer is indicated by a white square.

llama-2, llama-3, and mistral types. Therefore, additional models were included to ensure 10 models per category, resulting in a total of 9 added models. Fig. 17 shows the trajectories of these models and the KL divergence between adjacent layers, while Fig. 18 illustrates the trajectories in 3d space. For all model types, the trajectories are already clustered by model type from the shallow layers onward. In addition, the KL divergence values exhibit similar trends within each type.

I Model List

The models used in Sections 3.1, 3.3, and 3.4 are listed in Tables 5, 6, and 7, respectively. Table 5 is sorted by model size; Table 6 is sorted alphabetically by model name, considering the model generation, model type, and corresponding base model; and Table 7 is sorted in descending order of download counts.

BibTeX entries for each model were selected according to the procedure described by Oyama et al. (2025). Note that, for the Pythia models trained with different random seeds (van der Wal et al., 2025) and used in Section 4, the corresponding BibTeX entries in Table 5 were prepared manually.

Table 5: List of 14 models in the experiments for pretraining models. "ID" denotes the index sorted by model size; "Model Name" denotes the name of the model. Seeds 3 and 4 of Pythia-410M are excluded from the experiments in Sections 3.1 and 4.

ID	Model Name
1	EleutherAI/pythia-410m (Gao et al., 2020; Biderman et al., 2022, 2023)
2	EleutherAI/pythia-410m-seed1 (van der Wal et al., 2025)
3	EleutherAI/pythia-410m-seed2 (van der Wal et al., 2025)
4	EleutherAI/pythia-410m-seed3 (van der Wal et al., 2025)
5	EleutherAI/pythia-410m-seed4 (van der Wal et al., 2025)
6	EleutherAI/pythia-410m-seed5 (van der Wal et al., 2025)
7	EleutherAI/pythia-410m-seed6 (van der Wal et al., 2025)
8	EleutherAI/pythia-410m-seed7 (van der Wal et al., 2025)
9	EleutherAI/pythia-410m-seed8 (van der Wal et al., 2025)
10	EleutherAI/pythia-410m-seed9 (van der Wal et al., 2025)
11	EleutherAI/pythia-1b (Gao et al., 2020; Biderman et al., 2022, 2023)
12	EleutherAI/pythia-1.4b (Gao et al., 2020; Biderman et al., 2022, 2023)
13	EleutherAI/pythia-2.8b (Gao et al., 2020; Biderman et al., 2022, 2023)
14	EleutherAI/pythia-6.9b (Gao et al., 2020; Biderman et al., 2022, 2023)

Table 6: List of 87 models in the experiments for fine-tuning models. "ID" denotes the alphabetical index considering the generation and the base model; "Model Name" denotes the name of the model; "Base Model Name" denotes the name of the base model before training; "Model Type" denotes the model classification defined in Oyama et al. (2025); "Generation" denotes the model generation in the training process; "DLs" denotes the total number of downloads.

ID	Model Name	Base Model Name	Model Type	Generation	DLs
1	deepseek-ai/DeepSeek-Prover-V1.5-Base (Xin et al., 2024)	–	deepseek	1	230
2	tiiuae/falcon-rw-1b (Brown et al., 2020; Press et al., 2022; Dao et al., 2022; Penedo et al., 2023)	–	falcon	1	22921
3	google/gemma-1.1-7b-it (Joshi et al., 2017; Mihaylov et al., 2018; Zhao et al., 2018; Clark et al., 2019; Talmor et al., 2019; Sap et al., 2019; Sakaguchi et al., 2019; Bisk et al., 2019; Chollet, 2019; Zellers et al., 2019; Hendrycks et al., 2021; Chen et al., 2021; Austin et al., 2021; Cobbe et al., 2021; Parrish et al., 2022; Srivastava et al., 2023; Zhong et al., 2023; Gemini Team et al., 2024)	–	gemma	1	19835
4	google/gemma-7b (Joshi et al., 2017; Mihaylov et al., 2018; Zhao et al., 2018; Rudinger et al., 2018; Bisk et al., 2019; Chollet, 2019; Clark et al., 2019; Talmor et al., 2019; Sap et al., 2019; Zellers et al., 2019; Sakaguchi et al., 2019; Gehman et al., 2020; Chen et al., 2021; Austin et al., 2021; Hendrycks et al., 2021; Dhamala et al., 2021; Cobbe et al., 2021; Lin et al., 2022; Hartvigsen et al., 2022; Parrish et al., 2022; Srivastava et al., 2023; Zhong et al., 2023; Dettmers et al., 2023; Gemini Team et al., 2024)	–	gemma	1	72047
5	openchat/openchat-3.5-0106-gemma (Wang et al., 2024a)	–	gemma	1	7817
6	openlm-research/open_llama_3b (Together Computer, 2023; Touvron et al., 2023a; Geng and Liu, 2023)	–	llama-1	1	157405
7	codellama/CodeLlama-13b-Instruct-hf (Rozière et al., 2024)	–	llama-2	1	25248
8	meta-llama/Llama-2-13b-hf (Touvron et al., 2023b)	–	llama-2	1	120871
9	meta-llama/Llama-2-7b-hf (Touvron et al., 2023b)	–	llama-2	1	1294737
10	meta-llama/Meta-Llama-3-8B-Instruct (AI@Meta, 2024)	–	llama-3	1	2058011
11	meta-llama/Meta-Llama-3-8B (AI@Meta, 2024)	–	llama-3	1	675850
12	NousResearch/Hermes-2-Pro-Llama-3-8B ("Teknium" et al., 2024b)	–	llama-3	1	26797
13	mistralai/Mistral-7B-Instruct-v0.2 (Jiang et al., 2023)	–	mistral	1	3565248
14	mistralai/Mistral-7B-v0.1 (Jiang et al., 2023)	–	mistral	1	1750089
15	mlabonne/Marcoro14-7B-slerp	–	mistral	1	3792
16	mlabonne/NeuralMonarch-7B	–	mistral	1	13486
17	stabilityai/japanese-stablelm-base-gamma-7b (Jiang et al., 2023)	–	mistral	1	2072
18	beomi/KoRWKV-6B	–	others	1	2127
19	microsoft/Orca-2-13b (Mitra et al., 2023)	–	others	1	13616
20	upstage/SOLAR-10.7B-v1.0 (Kim et al., 2024b)	–	others	1	24478
21	01-ai/Yi-1.5-9B (01. AI et al., 2025)	–	others	1	20252
22	deepseek-ai/DeepSeek-Prover-V1.5-SFT (Xin et al., 2024)	deepseek-ai/DeepSeek-Prover-V1.5-Base	deepseek	2	6459
23	euclaise/falcon_1b_stage1	tiiuae/falcon-rw-1b	falcon	2	2119
24	lemon-mint/gemma-ko-7b-instruct-v0.71	google/gemma-1.1-7b-it	gemma	2	2232
25	lemon-mint/gemma-7b-openhermes-v0.80	google/gemma-1.1-7b-it	gemma	2	4867
26	google/gemma-7b-it (Joshi et al., 2017; Mihaylov et al., 2018; Zhao et al., 2018; Rudinger et al., 2018; Bisk et al., 2019; Chollet, 2019; Clark et al., 2019; Talmor et al., 2019; Sap et al., 2019; Zellers et al., 2019; Sakaguchi et al., 2019; Gehman et al., 2020; Chen et al., 2021; Austin et al., 2021; Hendrycks et al., 2021; Dhamala et al., 2021; Cobbe et al., 2021; Lin et al., 2022; Hartvigsen et al., 2022; Parrish et al., 2022; Srivastava et al., 2023; Zhong et al., 2023; Gemini Team et al., 2024)	google/gemma-7b	gemma	2	66776
27	Telugu-LLM-Labs/Indic-gemma-7b-finetuned-sft-Navarasa-2.0	google/gemma-7b	gemma	2	2025

ID	Model Name	Base Model Name	Model Type	Generation	DLs
28	lemon-mint/gemma-ko-7b-instruct-v0.62	openchat/openchat-3.5-0106-gemma	gemma	2	7543
29	rinna/youri-7b (Andonian et al., 2021; Touvron et al., 2023b; Zhao et al., 2023; Sawada et al., 2024)	meta-llama/Llama-2-7b-hf	llama-2	2	2512
30	aaditya/Llama3-OpenBioLLM-8B (Singhal et al., 2022; Nori et al., 2023; Singhal et al., 2023; Pal and Sankarasubbu, 2024b,a; Rafailov et al., 2024)	meta-llama/Meta-Llama-3-8B	llama-3	2	9308
31	cognitivecomputations/dolphin-2.9-llama3-8b	meta-llama/Meta-Llama-3-8B	llama-3	2	130977
32	cognitivecomputations/dolphin-2.9.1-llama-3-8b	meta-llama/Meta-Llama-3-8B	llama-3	2	7575
33	DeepMount00/Llama-3-8b-Ita	meta-llama/Meta-Llama-3-8B	llama-3	2	179401
34	dfurman/Llama-3-8B-Orpo-v0.1	meta-llama/Meta-Llama-3-8B	llama-3	2	5146
35	johnsnowlabs/JSL-MedLlama-3-8B-v2.0	meta-llama/Meta-Llama-3-8B	llama-3	2	11813
36	jondurbin/bagel-8b-v1.0	meta-llama/Meta-Llama-3-8B	llama-3	2	7960
37	openchat/openchat-3.6-8b-20240522 (Wang et al., 2024a)	meta-llama/Meta-Llama-3-8B	llama-3	2	10464
38	rinna/llama-3-youko-8b (Andonian et al., 2021; Mitsuda et al., 2024; AI@Meta, 2024; Sawada et al., 2024)	meta-llama/Meta-Llama-3-8B	llama-3	2	1468
39	ruslanmv/Medical-Llama3-8B	meta-llama/Meta-Llama-3-8B	llama-3	2	5457
40	lightblue/suzume-llama-3-8B-multilingual (Devine, 2024)	meta-llama/Meta-Llama-3-8B-Instruct	llama-3	2	16442
41	MaziyarPanahi/Llama-3-8B-Instruct-v0.4	meta-llama/Meta-Llama-3-8B-Instruct	llama-3	2	1407
42	PathFinderKR/Waktaverse-Llama-3-KO-8B-Instruct (AI@Waktaverse, 2024; AI@Meta, 2024)	meta-llama/Meta-Llama-3-8B-Instruct	llama-3	2	2305
43	shenzhi-wang/Llama3-8B-Chinese-Chat (Wang et al., 2024b)	meta-llama/Meta-Llama-3-8B-Instruct	llama-3	2	55718
44	SJ-Donald/llama3-passthrough-chat	meta-llama/Meta-Llama-3-8B-Instruct	llama-3	2	2241
45	swap-uniba/LLaMAntino-3-ANITA-8B-Inst-DPO-ITA (Basile et al., 2023; AI@Meta, 2024; Polignano et al., 2024)	meta-llama/Meta-Llama-3-8B-Instruct	llama-3	2	5995
46	NousResearch/Hermes-2-Theta-Llama-3-8B ("Teknium" et al., 2024a)	NousResearch/Hermes-2-Pro-Llama-3-8B	llama-3	2	12392
47	vicgalle/Configurable-Hermes-2-Pro-Llama-3-8B (Gallego, 2024)	NousResearch/Hermes-2-Pro-Llama-3-8B	llama-3	2	10273
48	abacusai/bigstral-12b-32k	mistralai/Mistral-7B-Instruct-v0.2	mistral	2	5216
49	Mihaiii/Metis-0.3	mistralai/Mistral-7B-Instruct-v0.2	mistral	2	1279
50	alignment-handbook/zephyr-7b-sft-full	mistralai/Mistral-7B-v0.1	mistral	2	11605
51	cognitivecomputations/dolphin-2.2.1-mistral-7b	mistralai/Mistral-7B-v0.1	mistral	2	7292
52	cypienai/cymist-2-v02-SFT (Lacoste et al., 2019)	mistralai/Mistral-7B-v0.1	mistral	2	2717
53	HuggingFaceH4/mistral-7b-sft-beta	mistralai/Mistral-7B-v0.1	mistral	2	7408
54	HuggingFaceH4/zephyr-7b-alpha (Ding et al., 2023; Tunstall et al., 2023; Cui et al., 2024; Rafailov et al., 2024)	mistralai/Mistral-7B-v0.1	mistral	2	12911
55	HuggingFaceH4/zephyr-7b-beta (Ding et al., 2023; Tunstall et al., 2023; Cui et al., 2024; Rafailov et al., 2024)	mistralai/Mistral-7B-v0.1	mistral	2	294019
56	ibm/merlinite-7b (Sudalairaj et al., 2024)	mistralai/Mistral-7B-v0.1	mistral	2	11156
57	INSAIT-Institute/BgGPT-7B-Instruct-v0.2	mistralai/Mistral-7B-v0.1	mistral	2	3265
58	Intel/neural-chat-7b-v3-1 (Mukherjee et al., 2023)	mistralai/Mistral-7B-v0.1	mistral	2	3976
59	kaist-ai/mistral-orpo-capybara-7k (Hong et al., 2024)	mistralai/Mistral-7B-v0.1	mistral	2	4821
60	Locutusque/Hercules-2.5-Mistral-7B	mistralai/Mistral-7B-v0.1	mistral	2	1953
61	mistralai/Mistral-7B-Instruct-v0.1 (Jiang et al., 2023)	mistralai/Mistral-7B-v0.1	mistral	2	1370245
62	NousResearch/Hermes-2-Pro-Mistral-7B ("interstellarninja" et al., 2024)	mistralai/Mistral-7B-v0.1	mistral	2	14586
63	NousResearch/Nous-Hermes-2-Mistral-7B-DPO ("Teknium" et al., 2024c)	mistralai/Mistral-7B-v0.1	mistral	2	8725
64	openchat/openchat-3.5-0106 (Wang et al., 2024a; OpenAI et al., 2024)	mistralai/Mistral-7B-v0.1	mistral	2	26858
65	openchat/openchat-3.5-1210 (Wang et al., 2024a; OpenAI et al., 2024)	mistralai/Mistral-7B-v0.1	mistral	2	2123
66	statking/zephyr-7b-sft-full-orpo	mistralai/Mistral-7B-v0.1	mistral	2	2278
67	teknium/OpenHermes-2.5-Mistral-7B	mistralai/Mistral-7B-v0.1	mistral	2	100997
68	mlabonne/NeuralMarcoro14-7B	mlabonne/Marcoro14-7B-slerp	mistral	2	2015
69	mlabonne/AlphaMonarch-7B	mlabonne/NeuralMonarch-7B	mistral	2	12804
70	augmxnt/shisa-gamma-7b-v1	stabilityai/japanese-stablelm-base-gamma-7b	mistral	2	151426
71	beomi/KoAlpaca-KoRWKV-6B	beomi/KoRWKV-6B	others	2	2288
72	Nexusflow/NexusRaven-V2-13B (Nexusflow.ai team, 2023; Rozière et al., 2024)	codellama/CodeLlama-13b-Instruct-hf	others	2	3966
73	haoranxu/ALMA-13B-Pretrain (Xu et al., 2024a,b)	meta-llama/Llama-2-13b-hf	others	2	5510
74	ruslanmv/ai-medical-model-32bit	meta-llama/Meta-Llama-3-8B-Instruct	others	2	2810
75	Locutusque/Orca-2-13b-SFT-v4	microsoft/Orca-2-13b	others	2	2345
76	Locutusque/Orca-2-13b-SFT-v6	microsoft/Orca-2-13b	others	2	2319
77	Locutusque/Orca-2-13b-SFT_v5	microsoft/Orca-2-13b	others	2	2191
78	mwitiderrick/open_llama_3b_code_instruct_0.1	openlm-research/open_llama_3b	others	2	1252
79	NousResearch/Nous-Hermes-2-SOLAR-10.7B	upstage/SOLAR-10.7B-v1.0	others	2	9918
80	upstage/SOLAR-10.7B-Instruct-v1.0 (Kim et al., 2024b,a)	upstage/SOLAR-10.7B-v1.0	others	2	67725
81	cognitivecomputations/dolphin-2.9.1-yi-1.5-9b	01-ai/Yi-1.5-9B	others	2	4880
82	deepseek-ai/DeepSeek-Prover-V1.5-RL (Xin et al., 2024)	deepseek-ai/DeepSeek-Prover-V1.5-SFT	deepseek	3	12223
83	euclaise/falcon_1b_stage2	euclaise/falcon_1b_stage1	falcon	3	4016
84	MaziyarPanahi/Llama-3-8B-Instruct-v0.8	MaziyarPanahi/Llama-3-8B-Instruct-v0.4	llama-3	3	7281
85	argilla/notus-7b-v1	alignment-handbook/zephyr-7b-sft-full	mistral	3	7660
86	Intel/neural-chat-7b-v3-3 (Yu et al., 2024)	Intel/neural-chat-7b-v3-1	mistral	3	166226
87	MaziyarPanahi/Llama-3-8B-Instruct-v0.9	MaziyarPanahi/Llama-3-8B-Instruct-v0.8	llama-3	4	6241

Table 7: List of 50 (+9) models in the experiments for quantization and layer-wise models. "ID" denotes the index sorted in descending order of downloads; "Model Name" denotes the name of the model; "Model Type" denotes the model classification defined in Oyama et al. (2025); "Layers" denotes the number of layers in the model; "DLs" denotes the total number of downloads.

ID	Model Name	Model Type	Layers	DLs
1	mistralai/Mistral-7B-Instruct-v0.2 (Jiang et al., 2023)	mistral	32	3565248
2	meta-llama/Meta-Llama-3-8B-Instruct (AI@Meta, 2024)	llama-3	32	2058011
3	mistralai/Mistral-7B-v0.1 (Jiang et al., 2023)	mistral	32	1750089
4	mistralai/Mistral-7B-v0.3	mistral	32	1443065
5	meta-llama/Llama-2-7b-chat-hf (Touvron et al., 2023b)	llama-2	32	1402244
6	mistralai/Mistral-7B-Instruct-v0.1 (Jiang et al., 2023)	mistral	32	1370245
7	meta-llama/Llama-2-7b-hf (Touvron et al., 2023b)	llama-2	32	1294737
8	TinyLlama/TinyLlama-1.1B-Chat-v1.0	others	22	1078500
9	TinyLlama/TinyLlama-1.1B-intermediate-step-1431k-3T	others	22	798206
10	meta-llama/Meta-Llama-3-8B (AI@Meta, 2024)	llama-3	32	675850
11	OpenAssistant/oasst-sft-4-pythia-12b-epoch-3.5	gpt_neox	36	458718
12	bigcode/starcoder2-3b (Beltagy et al., 2020; Dao et al., 2022; Bavarian et al., 2022; Ainslie et al., 2023; Lozhkov et al., 2024)	others	30	445316
13	lmsys/vicuna-7b-v1.5 (Touvron et al., 2023b; Zheng et al., 2023)	llama-2	32	368410
14	HuggingFaceH4/zephyr-7b-beta (Ding et al., 2023; Tunstall et al., 2023; Cui et al., 2024; Rafailov et al., 2024)	mistral	32	294019
15	meta-llama/Llama-2-13b-chat-hf (Touvron et al., 2023b)	llama-2	40	266090
16	google/gemma-2b (Joshi et al., 2017; Mihaylov et al., 2018; Zhao et al., 2018; Rudinger et al., 2018; Bisk et al., 2019; Chollet, 2019; Clark et al., 2019; Talmor et al., 2019; Sap et al., 2019; Zellers et al., 2019; Sakaguchi et al., 2019; Gehman et al., 2020; Chen et al., 2021; Austin et al., 2021; Hendrycks et al., 2021; Dhamala et al., 2021; Cobbe et al., 2021; Lin et al., 2022; Hartvigsen et al., 2022; Parrish et al., 2022; Srivastava et al., 2023; Zhong et al., 2023; Gemini Team et al., 2024)	gemma	18	256798
17	EleutherAI/gpt-neo-1.3B (Gao et al., 2020; Black et al., 2021)	others	24	243332
18	EleutherAI/gpt-j-6b (Gao et al., 2020; Wang and Komatsuzaki, 2021; Wang, 2021; Su et al., 2023)	gptj	28	241435
19	microsoft/phi-2	others	32	237268
20	EleutherAI/gpt-neo-2.7B (Gao et al., 2020; Black et al., 2021)	others	32	205503
21	huggyllama/llama-7b	llama-1	32	192383
22	tiiuae/falcon-7b-instruct (Shazeer, 2019; Brown et al., 2020; Dao et al., 2022; Su et al., 2023; Almazrouei et al., 2023; Penedo et al., 2023)	falcon	32	179952
23	DeepMount00/Llama-3-8b-Ita	llama-3	32	179401
24	Intel/neural-chat-7b-v3-3 (Yu et al., 2024)	mistral	32	166226
25	openlm-research/open_llama_3b (Together Computer, 2023; Touvron et al., 2023a; Geng and Liu, 2023)	llama-1	26	157405
26	augmxnt/shisa-gamma-7b-v1	mistral	32	151426
27	facebook/opt-1.3b (Brown et al., 2020; Zhang et al., 2022)	opt	24	139758
28	Qwen/Qwen1.5-1.8B (Bai et al., 2023)	others	24	137256
29	codellama/CodeLlama-7b-Instruct-hf (Rozière et al., 2024)	llama-2	32	133523
30	cognitivecomputations/dolphin-2.9-llama3-8b	llama-3	32	130977
31	Qwen/Qwen1.5-7B (Bai et al., 2023)	others	32	122768
32	meta-llama/Llama-2-13b-hf (Touvron et al., 2023b)	llama-2	40	120871
33	microsoft/phi-1_5 (Li et al., 2023)	others	24	112476
34	NousResearch/Meta-Llama-3-8B-Instruct (AI@Meta, 2024)	llama-3	32	104388
35	tiiuae/falcon-7b (Shazeer, 2019; Brown et al., 2020; Gao et al., 2020; Dao et al., 2022; Su et al., 2023; Almazrouei et al., 2023; Penedo et al., 2023)	falcon	32	104010
36	google/gemma-2b-it (Joshi et al., 2017; Mihaylov et al., 2018; Zhao et al., 2018; Rudinger et al., 2018; Bisk et al., 2019; Chollet, 2019; Clark et al., 2019; Talmor et al., 2019; Sap et al., 2019; Zellers et al., 2019; Sakaguchi et al., 2019; Gehman et al., 2020; Chen et al., 2021; Austin et al., 2021; Hendrycks et al., 2021; Dhamala et al., 2021; Cobbe et al., 2021; Lin et al., 2022; Hartvigsen et al., 2022; Parrish et al., 2022; Srivastava et al., 2023; Zhong et al., 2023; Gemini Team et al., 2024)	gemma	18	103851
37	teknium/OpenHermes-2.5-Mistral-7B	mistral	32	100997
38	mosaicml/mpt-7b-chat (Henry et al., 2020; Press et al., 2022; Dao et al., 2022; MosaicML NLP Team, 2023)	others	32	88069
39	Qwen/CodeQwen1.5-7B-Chat (Bai et al., 2023)	others	32	77169
40	GritLM/GritLM-7B (Muennighoff et al., 2025)	mistral	32	72991
41	google/gemma-7b (Joshi et al., 2017; Mihaylov et al., 2018; Zhao et al., 2018; Rudinger et al., 2018; Bisk et al., 2019; Chollet, 2019; Clark et al., 2019; Talmor et al., 2019; Sap et al., 2019; Zellers et al., 2019; Sakaguchi et al., 2019; Gehman et al., 2020; Chen et al., 2021; Austin et al., 2021; Hendrycks et al., 2021; Dhamala et al., 2021; Cobbe et al., 2021; Lin et al., 2022; Hartvigsen et al., 2022; Parrish et al., 2022; Srivastava et al., 2023; Zhong et al., 2023; Dettmers et al., 2023; Gemini Team et al., 2024)	gemma	28	72047
42	upstage/SOLAR-10.7B-Instruct-v1.0 (Kim et al., 2024b,a)	others	48	67725
43	google/gemma-7b-it (Joshi et al., 2017; Mihaylov et al., 2018; Zhao et al., 2018; Rudinger et al., 2018; Bisk et al., 2019; Chollet, 2019; Clark et al., 2019; Talmor et al., 2019; Sap et al., 2019; Zellers et al., 2019; Sakaguchi et al., 2019; Gehman et al., 2020; Chen et al., 2021; Austin et al., 2021; Hendrycks et al., 2021; Dhamala et al., 2021; Cobbe et al., 2021; Lin et al., 2022; Hartvigsen et al., 2022; Parrish et al., 2022; Srivastava et al., 2023; Zhong et al., 2023; Gemini Team et al., 2024)	gemma	28	66776
44	codellama/CodeLlama-7b-hf (Rozière et al., 2024)	llama-2	32	66040
45	facebook/opt-2.7b (Brown et al., 2020; Zhang et al., 2022)	opt	32	61450
46	shenzhi-wang/Llama3-8B-Chinese-Chat (Wang et al., 2024b)	llama-3	32	55718
47	VAGOsolutions/Llama-3-SauerkrautLM-8b-Instruct	llama-3	32	55400
48	lmsys/vicuna-13b-v1.5 (Touvron et al., 2023b; Zheng et al., 2023)	llama-2	40	48653
49	Qwen/Qwen2-1.5B (Yang et al., 2024)	others	28	46510
50	openlm-research/open_llama_7b (Together Computer, 2023; Touvron et al., 2023a; Geng and Liu, 2023)	llama-1	32	44968
51	LinkSoul/Chinese-Llama-2-7b	llama-2	32	40799
52	mistral-community/Mistral-7B-v0.2	mistral	32	30074
53	NousResearch/Hermes-2-Pro-Llama-3-8B ("Teknium" et al., 2024b)	llama-3	32	26797
54	lightblue/suzume-llama-3-8B-multilingual (Devine, 2024)	llama-3	32	16442
55	abacusai/Llama-3-Smaug-8B (Pal et al., 2024)	llama-3	32	12873
56	NousResearch/Nous-Hermes-llama-2-7b	llama-2	32	12019
57	togethercomputer/LLaMA-2-7B-32K	llama-2	32	8884
58	elyza/ELYZA-japanese-Llama-2-7b-instruct (Touvron et al., 2023b; Sasaki et al., 2023)	llama-2	32	6296
59	togethercomputer/Llama-2-7B-32K-Instruct (Liu et al., 2023)	llama-2	32	5596