Tending Towards Stability 📈: Convergence Challenges in Small Language Models

Richard Diehl Martinez **Pietro Lesci** **Paula Buttery**
University of Cambridge
{rd654, pl487, pjb48}@cam.ac.uk

Abstract

Increasing the number of parameters in language models is a common strategy to enhance their performance. However, smaller language models remain valuable due to their lower operational costs. Despite their advantages, smaller models frequently underperform compared to their larger counterparts, even when provided with equivalent data and computational resources. Specifically, their performance tends to degrade in the late pretraining phase. This is anecdotally attributed to their reduced representational capacity. Yet, the exact causes of this performance degradation remain unclear. We use the Pythia model suite to analyse the training dynamics that underlie this phenomenon. Across different model sizes, we investigate the convergence of the Attention and MLP activations to their final state and examine how the effective rank of their parameters influences this process. We find that nearly all layers in larger models stabilise early in training—within the first 20%—whereas layers in smaller models exhibit slower and less stable convergence, especially when their parameters have lower effective rank. By linking the convergence of layers' activations to their parameters' effective rank, our analyses can guide future work to address inefficiencies in the learning dynamics of small models.

rdiehlmartinez/pretraining-playground

1 Introduction

Scaling the number of parameters in language models (LMs) has provided impressive performance gains on a variety of tasks (Hendrycks et al., 2021) and has become the *de facto* standard to make progress in model design (e.g., Chowdhery et al., 2023). Small LMs, however, remain essential as they are more practical: lower training and inference costs result in a smaller environmental impact (Schwartz et al., 2020). Small LMs empower individuals to train on proprietary data by requiring fewer resources, enhancing data privacy (Huang et al., 2022) and democratising access to language modelling technology (Bender et al., 2021). However, for the same data and computational budget, small LMs (unsurprisingly) underperform larger ones (Biderman et al., 2023) and (importantly) their performance tends to degrade in the late pretraining phase, a phenomenon termed *saturation* by Godey et al. (2024).[1] Saturation is typically attributed to the "limited representational capacity" of small LMs; besides this anectodal justification, our understanding of its causes is still limited.

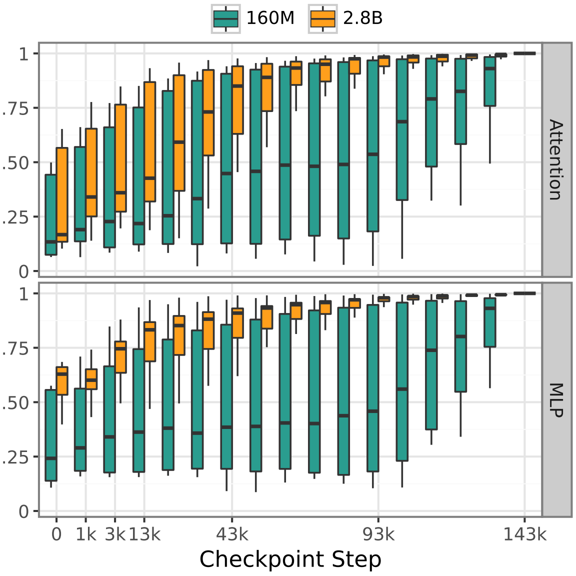

Figure 1: CKA similarity (current vs. last checkpoint) of Attention and MLP activations for Pythia 160M and 2.8B. Distribution across layers: 10, 25, 50, 75, and 90-th percentiles per checkpoint.

Recently, Godey et al. (2024) linked saturation to the reduced variability of the output embeddings of LMs caused by the mismatch between the hidden model dimension and the vocabulary size (Yang et al., 2018). Specifically, the last layer of LMs

[1] Subsequent references in this paper to *saturation* align with the concept introduced by Godey et al. (2024).

Findings of the Association for Computational Linguistics: EACL 2024, pages 3275–3286
November 12-16, 2024 ©2024 Association for Computational Linguistics

maps the hidden representation of random tokens to output embeddings with high cosine similarity.[2]

In this paper, we use the Pythia model suite (Biderman et al., 2023) to provide orthogonal analyses that consider models' training dynamics. First, we study how the activations of the $\mathrm{Attention}$ and MLP layers converge to their final state across LMs of different sizes. Then, we relate the difference in convergence behaviour across sizes to the effective rank of their parameters: layers whose activations converge later in training span a smaller fraction of their dimensions.

Specifically, we first use the **Centered Kernel Alignment** (CKA; Kornblith et al., 2019) metric to measure the similarity of layers' activations across checkpoints. We observe that larger LMs converge faster and more smoothly to their final state. As shown in Fig. 1, within the first 20% of training nearly all layers in the larger LM (2.8B) resemble their final state, while most layers in the smaller LM (160M) remain different for most of training.

We then find a strong correlation between the convergence pattern of a layer's activations and the rank of its parameters and gradients. We introduce the concept of **proportional effective rank** (§3) to consistently compare these effective ranks across model sizes. Our analyses highlight training inefficiencies in small-scale LMs, paving the way for targeted improvements in future work.

2 Related Work

Prior work has studied various learning dynamics of the Pythia suite, including memorisation (Biderman et al., 2023; Lesci et al., 2024), training data influence (Liu et al., 2024), and statistics of learned embeddings (Belrose et al., 2024). Related to our work, Godey et al. (2024) examine the differences in the rank of the unembedding matrix (mapping from hidden representations to tokens) across model sizes, known as the softmax bottleneck (Yang et al., 2018). Unlike their findings, we focus on the convergence dynamics of all layers.

Similarity metrics like CKA and Singular Vector Canonical Correlation Analysis (SVCCA) are widely used to analyse language model properties. Nguyen et al. (2021) find that architectural decisions, such as model width and depth, affect hidden representation similarity. Wu et al. (2020) show that models within the same architectural family share similar hidden structures, a similarity that persists even in fine-tuned models (Phang et al., 2021). Additionally, SVCCA has been used to study token representation distribution in multilingual models (Singh et al., 2019) and syntactic element learning in monolingual models (Saphra and Lopez, 2019). Most similar to our work, Brown et al. (2023) use representation similarity metrics, including CKA, to study Pythia generalisation capabilities. However, our study is the first to use the CKA metric to examine the convergence dynamics of layers' activations across model sizes.

3 Methodology

We first describe the residual stream view of transformer-based models and define layers' activations. Then, we introduce the CKA and proportional effective rank metrics.

The *Residual Stream* view. The residual stream view of the transformer architecture (Vaswani et al., 2017) is an analytical framework to study how information flows through its layers (Elhage et al., 2021). This conceptualisation focuses on the residual connections as they provide a direct reference to the inputs. Specifically, the set of residual connections across layers is termed the **residual stream**. Each layer can be seen as providing modifications to the residual stream via addition operations. Layers have two main components, $\mathrm{Attention}$ and MLP, that sequentially update the residual stream. Formally, a sequence of T tokens $\mathbf{t} = \langle t_1, ..., t_T \rangle$ is first converted into a matrix $\boldsymbol{x}_0 \in \mathbb{R}^{T \times D}$ by the embedding layer: each column is a token representation of size D. Then, each layer $l \in \{1, ..., L\}$ updates these representations as follows:

$$\boldsymbol{x}' = \boldsymbol{x}_{l-1} + \underline{\mathrm{Attention}(\boldsymbol{x}_{l-1})} \tag{1}$$

$$\boldsymbol{x}_l = \boldsymbol{x}' + \underline{\mathrm{MLP}(\boldsymbol{x}')} \tag{2}$$

Finally, the T-th column of $\boldsymbol{x}_L$ is used to predict the $(T+1)$-th token. More details in App. A.

***Activations* and *Parameters*.** The updates to the residual stream—underlined in eq. (1)—are the layer's **activations** and have the same dimensions as the residual stream, i.e., $\mathbb{R}^{T \times D}$. Both $\mathrm{Attention}$ and MLP first project, or "read", the residual stream into lower-dimensional intermediate representations; then project these representations back, or "write", into the residual stream. Here, we study the behaviour of the **parameters** that write to the residual stream. We use $\boldsymbol{a}^{\mathtt{ATT}}$ and $\boldsymbol{a}^{\mathtt{MLP}}$ to denote

[2] This issue is termed *anisotropy* (Ethayarajh, 2019).

the activations and $\boldsymbol{\theta}^{\texttt{ATT}}$ and $\boldsymbol{\theta}^{\texttt{MLP}}$ to denote the parameters of, respectively, Attention and MLP.

Activations' Similarity. Given a set of activations, either $\boldsymbol{a}^{\texttt{ATT}}$ or $\boldsymbol{a}^{\texttt{MLP}}$, of a layer l at a particular checkpoint c, $\boldsymbol{a}_{l,c}$, we measure how similar they are to those at the last checkpoint C, $\boldsymbol{a}_{l,C}$, using the linear variant of the Centred Kernel Alignment metric (CKA; Kornblith et al., 2019):

$$\mathrm{CKA}(\overline{\boldsymbol{a}}_c, \overline{\boldsymbol{a}}_C) = \frac{\left\|\overline{\boldsymbol{a}}_c{}^\top \overline{\boldsymbol{a}}_C\right\|_F^2}{\left\|\overline{\boldsymbol{a}}_c{}^\top \overline{\boldsymbol{a}}_c\right\|_F \ \left\|\overline{\boldsymbol{a}}_C{}^\top \overline{\boldsymbol{a}}_C\right\|_F} \quad (3)$$

where $\overline{\boldsymbol{a}}$ denotes the centred activations, and $\|\cdot\|_F$ is the Frobenius norm; we omit the layer subscript l for clarity. We compute eq. (3) for both $\boldsymbol{a}^{\texttt{ATT}}$ and $\boldsymbol{a}^{\texttt{MLP}}$ across all layers and checkpoints throughout training, allowing us to examine the convergence dynamics of each layer's activations.

Parameters' *Proportional Effective Rank*. Let H be the dimension of the intermediate representation of either Attention or MLP. For a layer l, let $\boldsymbol{\theta}_l \in \mathbb{R}^{D\times H}$ be the subset of parameters of either $\boldsymbol{\theta}^{\texttt{ATT}}$ or $\boldsymbol{\theta}^{\texttt{MLP}}$ that comprise the matrix that projects from the hidden space into the residual stream. We measure the effective number of dimensions onto which $\boldsymbol{\theta}_l$ projects the intermediate representations using the definition of **effective rank** introduced in Roy and Vetterli (2007). The effective rank is computed as the entropy over the normalised singular values of the parameter matrix $\boldsymbol{\theta}_l$, that is:

$$\mathrm{ER}(\boldsymbol{\theta}_l) = \exp\left(-\sum_{k=1}^{K} \frac{\sigma_k}{\|\sigma\|_1} \log \frac{\sigma_k}{\|\sigma\|_1}\right) \quad (4)$$

where $\sigma = \langle\sigma_1, ..., \sigma_K\rangle$ is the vector of singular values and $\|\cdot\|_1$ is the ℓ_1 norm. In this paper, we introduce the notion of a **proportional effective rank** (PER) computed as the effective rank normalised by the number of hidden dimensions:

$$\mathrm{PER}(\boldsymbol{\theta}_l) = \mathrm{ER}(\boldsymbol{\theta}_l)\,/\,H \quad (5)$$

The PER allows us to compare the effective rank of layers with different sizes consistently. We compute the PER of both $\boldsymbol{\theta}^{\texttt{ATT}}$ and $\boldsymbol{\theta}^{\texttt{MLP}}$, as well as the gradients of these parameters, across all layers and checkpoints throughout training.

4 Experimental Setup

We use the Pythia model suite (Biderman et al., 2023), composed of 8 transformers of different sizes trained for 143k steps on the deduplicated[3] version of the Pile dataset (Gao et al., 2020; Biderman et al., 2022). Intermediate checkpoints are available every 1k steps and at log-spaced intervals early in training. To comply with our computational budget, we consider models up to 2.8B parameters—i.e., 70M, 160M, 410M, 1.4B, and 2.8B—evaluated at the following steps: 0, all log-spaced steps $\{1, 2, 4, ..., 512\}$, 1k, 3k, and then every 10k steps up to 143k. We evaluate each checkpoint on the last batch of the training set and collect its activations. More details in App. B.

5 Results

Our analyses reveal quantitative differences in the learning dynamics of layers across model sizes.

Result 1. Activations of larger models converge faster and more monotonically to their final state than those of smaller models. As observed in Fig. 2 (first column), larger models show, on average, earlier convergence of Attention and MLP activations. For example, by 20% of training, the CKA score in 2.8B is 0.8 for MLP and 0.7 for Attention, where in 70M and 160M it is around 0.5. This fast convergence pattern holds across layers, as shown by the distributions in Fig. 1.

Result 2. Activations of earlier layers converge faster, regardless of the model size. Across model sizes, earlier layers' activations converge faster to their final state than those of later layers. As shown in Fig. 3 (App. C), the faster average convergence in larger models is due to more of their later layers converging earlier, whereas smaller models' layers only reach their final state towards the end of training.

Based on recent work that identifies parameter rank differences across model sizes (Godey et al., 2024), in the next paragraphs, we study whether the different convergence behaviours are related to the effective rank of layers' parameters and gradients.

Result 3. Parameters of layers in larger models proportionally span more dimensions. Parameters in layers of larger models span a slightly larger fraction of their available dimensions compared to smaller models, as shown in Fig. 2 (second column). Moreover, the PER of larger models stabilises early, while it keeps decreasing throughout training for smaller ones. This finding is further

[3] There exists a non-deduplicated (or standard) version of the Pile dataset used to train a first version of the Pythia suite.

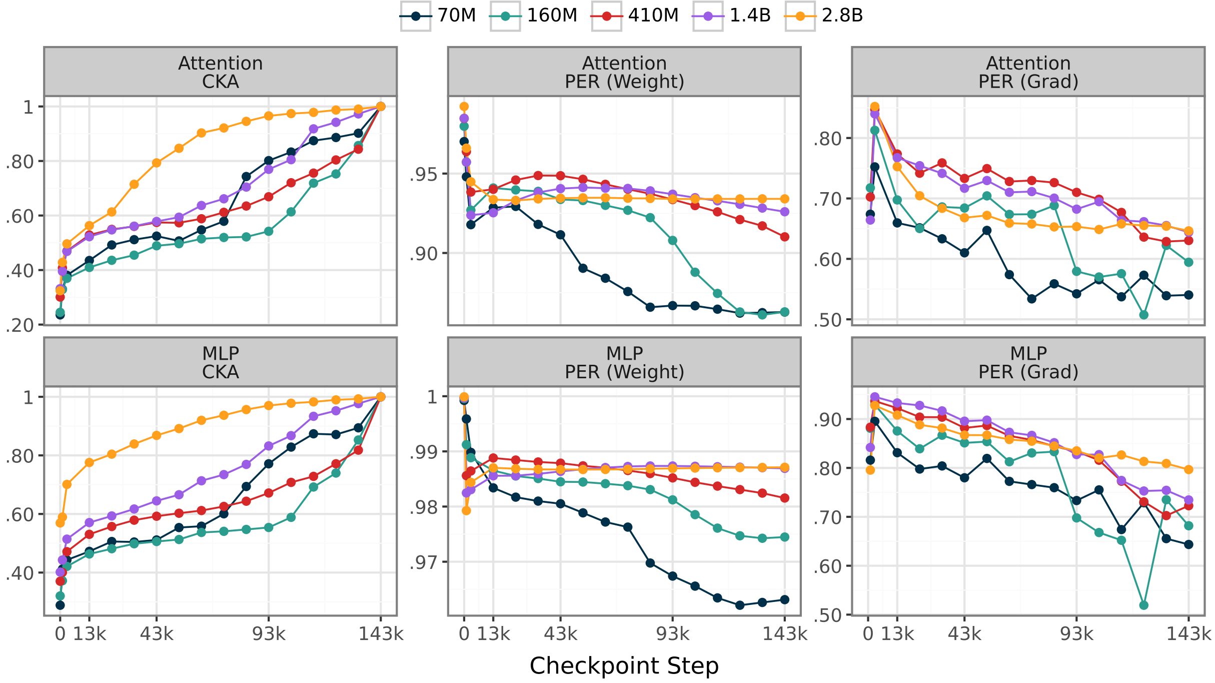

Figure 2: CKA similarity (current vs. last checkpoint) of layers' activations (first column), PER of layers' parameters (second column) and gradients (third column) for Attention (top row) and MLP (bottom row) in Pythia 70M, 160M, 410M, 1.4B, and 2.8B averaged (mean) across layers per each checkpoint.

underscored when visualising the PER for each layer, as shown in Fig. 4 (App. D); we observe that in smaller models the PER of later layers tends to decrease over the course of training, while in larger models the PER of all layers stabilises early in training. This difference is even more pronounced in the PER of these layers' gradients, as shown in Fig. 2 (third column).

Result 4. Parameters of layers in larger models receive gradient updates along proportionally more dimensions. The PER of gradients reflects the proportion of the learning signal transmitted by the gradients relative to the available parameter dimensions. In Fig. 2 (third column), we observe that throughout training gradients in larger models consistently span a larger fraction of the available dimensions, with this fraction gradually decreasing over time. In contrast, smaller models display more variability. At first glance, the averaged PER of gradients in the Attention layer of the 2.8B model might appear to contradict the observed trend. However, this discrepancy is clarified when examining the PER of gradients across individual layers, as shown in Fig. 5 (App. E). Once again, we observe that the PER of gradients in later layers of smaller models are less stable compared to larger models. The reason the average PER of gradients in the Attention layer of the 2.8B model is smaller than in smaller models is that, early in training, all layers of the larger model stabilise at their final values. At this stage, the stabilised layers of the larger model have lower gradient PER values compared to those of smaller models, which have not yet converged. Overall, our findings suggest that layers in larger models converge both more quickly and tend to receive proportionally larger rank updates during training.

Result 5. The dynamics of the parameters' effective rank and the activations' convergence patterns are correlated. We investigate the correlation between a layer's activations convergence rate and the rank of its parameters and gradients. Broadly, we find that layers with higher effective rank in both weights and gradients converge faster. To measure this correlation, we first create two binary variables for each layer indicating whether (i) it converges early in training and (ii) maintains a stable PER throughout training. Then, we calculate the Matthew's Correlation Coefficient between these two statistics across layers and report them in Table 1. Specifically, for each layer of a given model, we determine whether that layer exhibits early activations' convergence and large and stable parameters' and gradients' PERs (relative to other

Size	θ^{ATT}	$\nabla\theta^{\text{ATT}}$	θ^{MLP}	$\nabla\theta^{\text{MLP}}$
70M	1.00	1.00	0.63	1.00
160M	1.00	0.85	0.36	0.71
410M	0.84	0.92	0.19	0.78
1.4B	0.78	0.85	0.21	0.64
2.8B	0.73	0.52	0.11	0.18

Table 1: Matthew's Correlation Coefficient between binary variables indicating whether a given layer converges early in training and whether it maintains a stable PER of the parameters ($\boldsymbol{\theta}$) and gradients ($\nabla\boldsymbol{\theta}$) throughout training for both Attention and MLP.

model layers) using the following heuristics:

- **Early activations' convergence.** Activations' CKA ≥ 0.45 by the first 10% of training (applies to both the Attention and MLP layers).
- **Large parameters'** PER**.** Parameters' PER ≥ 0.95 by the end of training (applies to both the Attention and MLP layers).
- **Large gradients'** PER**.** We note that gradients' PER slightly decreases throughout training for each model size. Rather than choosing a fixed value to determine large and stable gradients' PERs, we dynamically set the threshold at 90% of the largest PER attained by any layer at the end of training.

We observe a strong correlation for the Attention layers across model sizes. For the MLP layers, the correlation with the gradients' PER is strong for models up to 1.4B, while the correlation with the parameters' PER is strong only for the 70M model. We hypothesise that this discrepancy can be explained by the fact that MLP layers have a large PER throughout training across all model sizes, apart from those of the 70M model.

While these results are correlational, they provide a foundation for future work to test whether methods that specifically increase the PER of layers' parameters and gradients induce faster convergence of the layers' activations in small models.

6 Conclusion

Our study highlights disparities in the learning dynamics of small and large LMs. Using the Pythia model suite, we demonstrate that layers' activations in larger models converge faster and more monotonically to their final state. We correlate this phenomenon with the larger PER in the parameters and gradients of larger models. Our analyses expand our understanding of training inefficiencies in small models and provide insights for future work to address them, e.g., by developing methods that increase the PER of layers' parameters.

Ethical Impact

Our work is part of a greater effort in Green AI (Schwartz et al., 2020) to lower the environmental footprint of training and using language models. We acknowledge, however, that small language models are prone to the same types of biases as large language models that are encoded through the data the models are trained on; the Pile is known to contain gender and racial biases (Gao et al., 2020).

Limitations

We experiment only with the Pythia model suite and the Pile dataset. It is unclear to what extent our findings translate to other models and datasets (including datasets in languages other than English). Moreover, because of our restricted computational budget, we are limited in our ability to thoroughly study larger language models. The largest models we experiment with are still relatively small given the scale of currently available open-source large language models (in the hundreds of billions). Finally, the relationship we find between the CKA similarity scores and the proportional effective rank is purely correlational: in future work, we aim to use our results to guide targeted interventions to assess whether the relationship we found is causal, i.e. whether increasing the effective rank of a layer can increase its convergence speed.

Acknowledgements

We thank the anonymous reviewers for their helpful comments, which helped us improve the paper.

The experiments reported in this paper were performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service, provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council. Richard Diehl Martinez is supported by the Gates Cambridge Trust (grant OPP1144 from the Bill & Melinda Gates Foundation).

Pietro received funding from the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation programme grant AVeriTeC (Grant agreement No. 865958).

References

Nora Belrose, Quintin Pope, Lucia Quirke, Alex Mallen, and Xiaoli Fern. 2024. Neural networks learn statistics of increasing complexity. *arXiv preprint 2402.04362*.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Stella Biderman, Kieran Bicheno, and Leo Gao. 2022. Datasheet for the Pile. *arXiv preprint 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. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23.

Davis Brown, Charles Godfrey, Nicholas Konz, Jonathan Tu, and Henry Kvinge. 2023. Understanding the inner-workings of language models through representation dissimilarity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6543–6558, Singapore. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

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. *arXiv preprint 2101.00027*.

Nathan Godey, Éric de la Clergerie, and Benoît Sagot. 2024. Why do small language models underperform? Studying language model saturation via the softmax bottleneck. *arXiv preprint 2404.07647*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022. Are large pre-trained language models leaking your personal information? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2038–2047, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR.

Pietro Lesci, Clara Meister, Thomas Hofmann, Andreas Vlachos, and Tiago Pimentel. 2024. Causal estimation of memorisation profiles. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15616–15635, Bangkok, Thailand. Association for Computational Linguistics.

Qingyi Liu, Yekun Chai, Shuohuan Wang, Yu Sun, Keze Wang, and Hua Wu. 2024. On training data influence of GPT models. *arXiv preprint 2404.07840*.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. 2021. Do wide and deep networks learn the same things? Uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jason Phang, Haokun Liu, and Samuel R. Bowman. 2021. Fine-tuned transformers show clusters of similar representations across layers. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 529–538, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Olivier Roy and Martin Vetterli. 2007. The effective rank: A measure of effective dimensionality. In *15th European Signal Processing Conference*, pages 606–610, Poznan, Poland.

Naomi Saphra and Adam Lopez. 2019. Understanding learning dynamics of language models with SVCCA. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3257–3267, Minneapolis, Minnesota. Association for Computational Linguistics.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green AI. *Commun. ACM*, 63(12):54–63.

Jasdeep Singh, Bryan McCann, Richard Socher, and Caiming Xiong. 2019. BERT is not an interlingua and the bias of tokenization. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 47–55, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

John Wu, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2020. Similarity analysis of contextual word representation models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4638–4655, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

A The Residual Stream View

The residual stream is a mathematical formalization through which to study how transformer models process inputs (Elhage et al., 2021). Under this framework, each of the L layers of a transformer model processes a series of input tokens $\mathbf{t} = \langle t_1, ..., t_T \rangle$ consecutively and communicate the result of their computation for each token to subsequent layers via a residual stream of dimension D. The reading, processing, and writing of the residual stream occur independently in each $\mathrm{Attention}$ head via combinations of the query, key, value and output matrices, W_Q, W_K, W_V, W_O: The **query-key circuit**, $W_Q^\top W_K$, of the $\mathrm{Attention}$ mechanism controls how the residual stream should be recomposed, and the **output circuit**, $W_O W_V$, writes to the residual stream an update that is mediated by the query-key circuit. The write operation of each $\mathrm{Attention}$ head is of low rank relative to D. After each $\mathrm{Attention}$ head has written to the residual stream, a bottleneck MLP projection performs a full-rank transformation on the residual stream. Due to their pivotal role in updating the state of the residual stream, our work analyses the learning dynamics of the two operations that write to the residual stream: the output circuit of each head of the $\mathrm{Attention}$ layer—that we refer to as $\mathrm{Attention}$—and the MLP projection layer—that we denote MLP for conciseness.

B Implementation Details

We implement all experiments using the PyTorch framework (Paszke et al., 2019). We access the Pythia models through the transformers library (Wolf et al., 2020).

B.1 Hardware Details

We use a server with one NVIDIA A100 80GB PCIe, 32 CPUs, and 32 GB of RAM for all experiments. Collecting model activations for all analyses required in total about 24 GPU hours. Below, we report a subset of the output of the lscpu command:

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Address sizes:         46 bits physical,
                       48 bits virtual
Byte Order:            Little Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Vendor ID:             GenuineIntel
Model name:            Intel(R) Xeon(R)
                       Silver 4210R CPU
                       @ 2.40GHz
CPU family:            6
Model:                 85
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             8
Stepping:              7
BogoMIPS:              4800.11
```

B.2 The Pythia Suite

We use the publicly available Pythia model suite (Biderman et al., 2023), which was trained on the Pile (Gao et al., 2020; Biderman et al., 2022). Both the preprocessed training data and intermediate checkpoints are publicly available.[4]

Data. The Pile is a 300B-token curated open-source collection of English documents, spanning a wide range of domains (e.g. books, academic publications, Wikipedia).[5] The deduplicated version of the dataset is obtained by applying a near-deduplication method based on MinHashLSH and has 207B tokens. Thus, models trained on this version of the dataset are trained for circa 1.5 epochs to keep an equal token

[4] github.com/EleutherAI/pythia (Apache License 2.0).

[5] github.com/EleutherAI/the-pile (MIT License).

Size	L	D	# Heads	Head Dim.	Batch Size	Learning Rate	Checkpoints
70M	6	512	8	64	2M	1×10^{-3}	Standard, Deduped
160M	12	768	12	64	2M	6×10^{-4}	Standard, Deduped
410M	24	1,024	16	64	2M	3×10^{-4}	Standard, Deduped
1.4B	24	2,048	16	128	2M	2×10^{-4}	Standard, Deduped
2.8B	32	2,560	32	80	2M	1.6×10^{-4}	Standard, Deduped

Table 2: Details on the architecture and training hyper-parameters for models in the Pythia suite used in this paper.

count relative to the non-deduplicated versions. The dataset is shuffled, tokenised, and "packed" into sequences of $2{,}049$ tokens with no end-of-document token.[6] Noticeably, the packing process implies that the second half-epoch of deduplicated data contains the same documents but not necessarily the same sequences. By design, each sequence can pack multiple documents and tokens can attend across document boundaries.

Models. The Pythia model suite is composed of 16 models: transformers of 8 different sizes trained on the Pile as-is and deduplicated. All model sizes were trained using a cosine learning rate schedule with warm-up, the same data order, and a batch size of $1{,}024$ sequences, resulting in exactly 143k optimization steps. Checkpoints are available at initialization (step 0), and after every 1k iterations (steps 1k-143k) resulting in 144 checkpoints evenly spaced throughout training. Additionally, log-spaced checkpoints are available early in training (steps $\{2^i\}_{i=0}^{9}$). In Table 2 we report more details about the architecture and training hyper-parameters of the models in the suite.

[6] github.com/EleutherAI/pythia/issues/123.

C Layer-wise CKA Convergence Dynamics

In Fig. 3, we visualise the activations' CKA convergence dynamics of layers in different models as a colour-coded line plot.

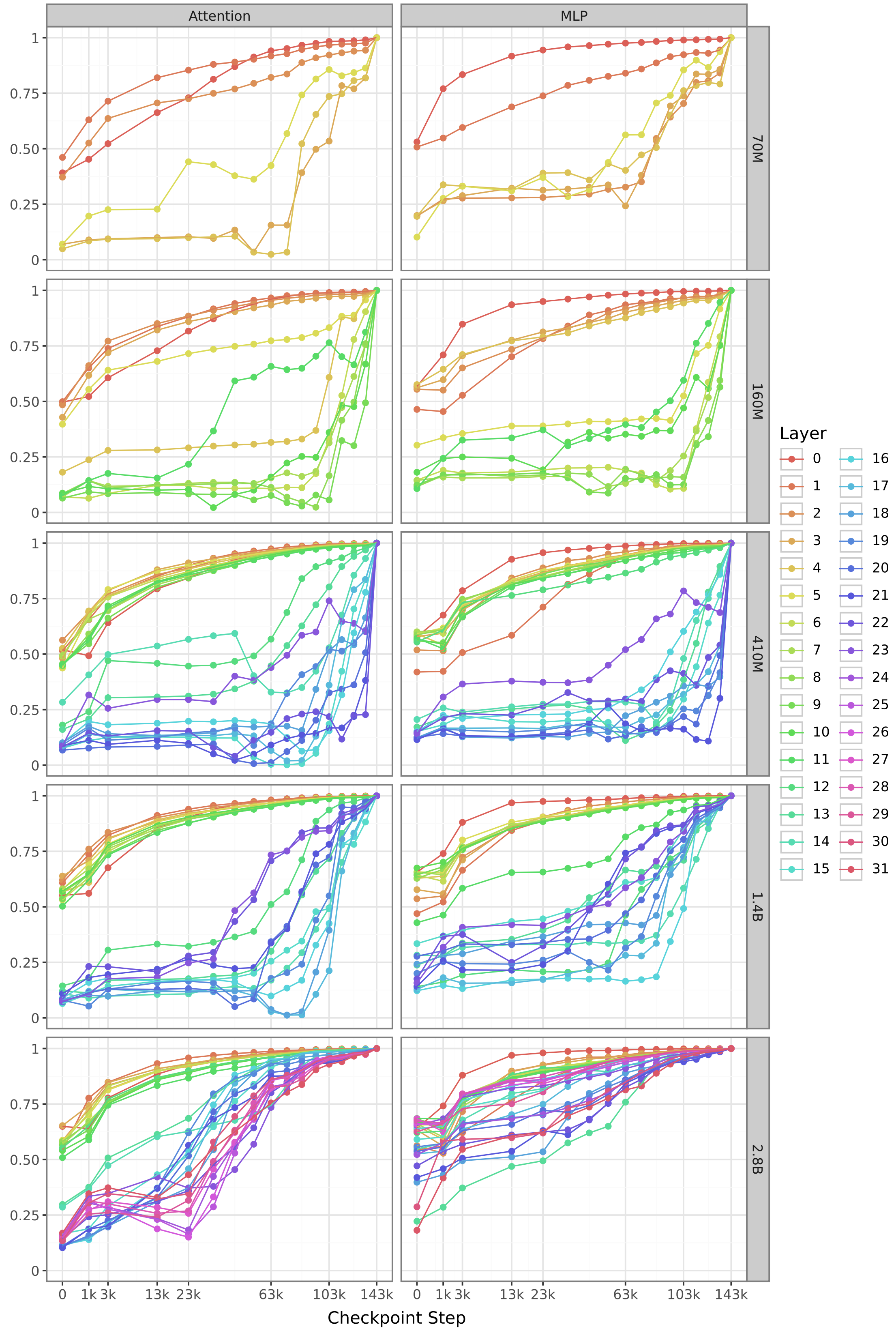

Figure 3: CKA similarity (current vs last checkpoint) of the activations of Attention and MLP in each layer of Pythia 70M, 160M, 410M, 1.4B and 2.8B throughout training.

D Layer-wise PER Weight Dynamics

In Fig. 4, we visualise the learning dynamics of the PER of weight matrices of layers in different models as a colour-coded line plot.

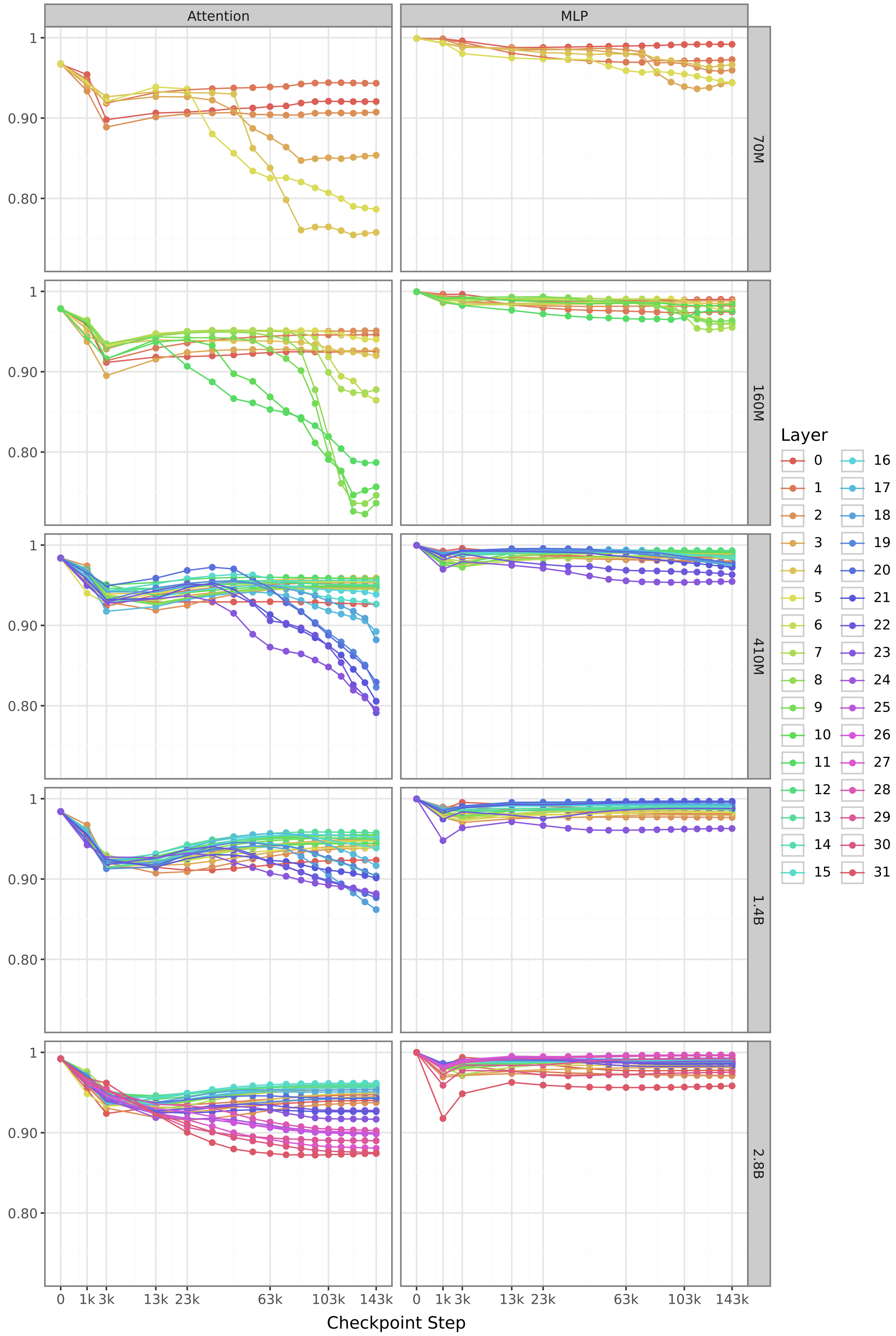

Figure 4: PER of the weight matrices of Attention and MLP in each layer of Pythia 70M, 160M, 410M, 1.4B and 2.8B throughout training.

E Layer-wise PER Gradient Dynamics

In Fig. 5, we visualise the learning dynamics of the PER of gradients of layers in different models as a colour-coded line plot.

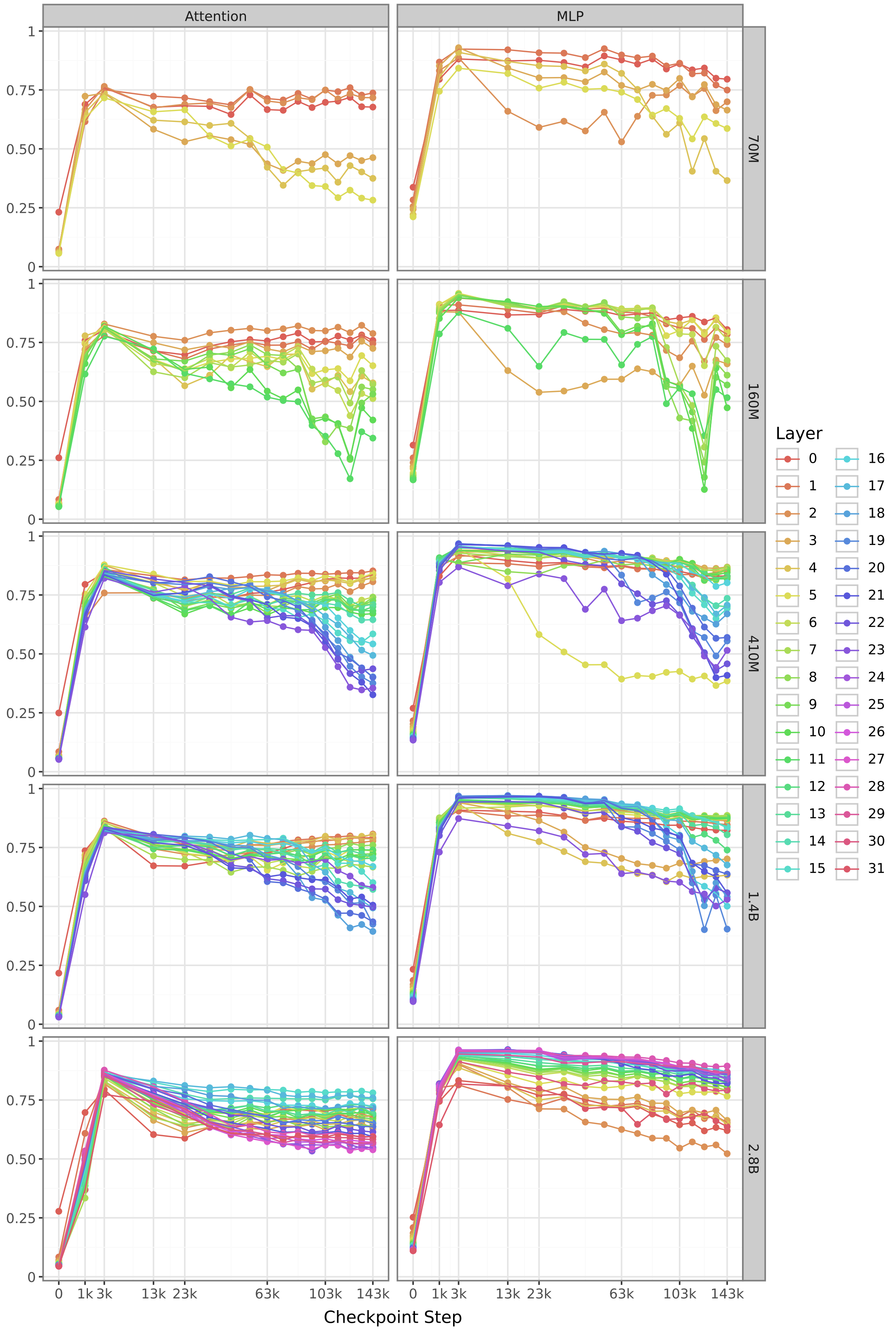

Figure 5: PER of the gradients of the weight matrices of Attention and MLP in each layer of Pythia 70M, 160M, 410M, 1.4B and 2.8B throughout training.