Loss landscape geometry reveals stagewise development of transformers

George Wang[=] GEORGE@TIMAEUS.CO
Timaeus

Matthew Farrugia-Roberts[=] MATTHEW@FAR.IN.NET
Independent & Timaeus

Jesse Hoogland JESSE@TIMAEUS.CO
Timaeus

Liam Carroll LEMMYKC@GMAIL.COM
Independent

Susan Wei SUSAN.WEI@UNIMELB.EDU.AU
School of Mathematics and Statistics, the University of Melbourne

Daniel Murfet D.MURFET@UNIMELB.EDU.AU
School of Mathematics and Statistics, the University of Melbourne

Abstract

The development of the internal structure of neural networks throughout training occurs in tandem with changes in the local geometry of the population loss. By quantifying the degeneracy of this geometry using the recently proposed Local Learning Coefficient, we show that the training process for a transformer language model can be decomposed into discrete developmental stages. We connect these stages to interpretable shifts in input–output behavior and developments in internal structure. These findings offer new insights into transformer development and underscore the crucial role of loss landscape geometry in understanding the dynamics of deep learning.

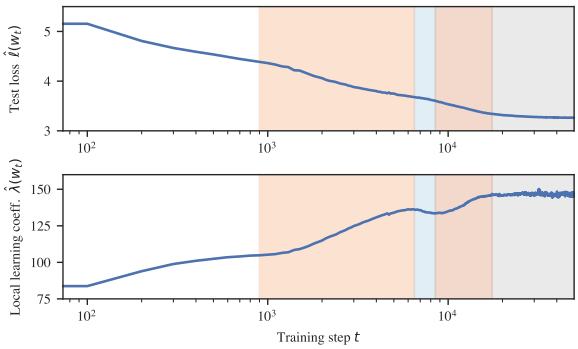
1. Introduction

A striking phenomenon in modern deep learning is the event of sudden shifts in a model's internal computational structure and associated changes in input–output behavior [22, 28, 41]. Understanding this phenomenon is a priority for the science of deep learning. **We propose that the local geometry of the population loss holds the key to understanding this phenomenon.** This is motivated by the perspective of singular learning theory [39], wherein the local geometry of the model likelihood governs stagewise development in the Bayesian posterior distribution with increasing samples [6, 39]. In this paper we study the connection between loss landscape geometry and stagewise development. We propose and investigate the following geometry-based methodology for stage identification.

- **Transformer training.** We train a transformer language model with around 3 million parameters on a subset of the Pile [15, 43].
- **Geometry tracing.** We track the evolution of degeneracy in the local geometry of the loss landscape by estimating the Local Learning Coefficient [LLC; 18] at frequent checkpoints.
- **Stage division.** Motivated by the singular learning process in Bayesian statistics [6, 39], we identify critical points of the LLC curve and use them to divide training into stages.

 [=]Equal contribution.

This methodology reveals our transformer's development can be divided into five clear stages (Figure 1). We show that this division is meaningful in that the stages coincide with key changes in the model's internal structure and input–output behavior, revealing that the transformer implements a progression of language modeling heuristics over training. In particular, we associate the first four stages with learning to predict according to bigram statistics (Stage LM1), learning to predict frequent n-grams and use the positional embedding (Stage LM2), and forming "previous-token heads" and "induction heads" as studied by Olsson et al. [28] (Stages LM3 and LM4).



Key	Stage	End t	$\Delta\hat{\ell}$	$\Delta\hat{\lambda}$
	LM1	900	-2.33	$+26.4$
	LM2	6.5k	-1.22	$+22.5$
	LM3	8.5k	-0.18	-1.57
	LM4	17k	-0.40	$+8.62$
	LM5	50k	-0.34	$+1.77$

Figure 1: We train a two-layer attention-only transformer language model, tracking test loss (top left) and estimated Local Learning Coefficient (LLC; bottom left; see Section 4). Critical points in the LLC curve divide training into distinct *developmental stages* (right). Orange hue indicates LLC increase, blue indicates decrease. Additional seeds in Appendix G.

2. Related work

The study of stagewise development in artificial neural networks has a long history [3, 30, 32, 34] with renewed relevance due to emergent structure and behavior in large models. This section briefly reviews recent related work. We discuss additional related work in Appendix A.

Elhage et al. [9], Olsson et al. [28] studied fully-trained language models, showing "zero-layer" transformers (embedding, then zero attention layers, then unembedding) learn bigram statistics; one-layer transformers learn skip-trigrams; and two-layer transformers form an *in-context learning circuit* with previous-token heads (layer 1) and induction heads (layer 2). We show a similar progression occurring *in a single* two-layer transformer, *throughout training* rather than across architectures.

Chen et al. [5] showed that the emergence of syntactic attention structure within language models coincides with a sudden change in the loss and the intrinsic dimension [another geometric complexity measure; 11]. Contemporaneously, Edelman et al. [7] study transformers trained on Markov chain sequence modeling, showing stagewise development of bigrams, previous-token heads and induction heads revealed by the loss curve. In contrast, we propose a principled method for stage identification using critical points in the LLC curve, capable of revealing stages not visible from the loss alone.

Barak et al. [4] and Nanda et al. [25] give examples of *progress measures,* metrics that reflect hidden model changes not visible in the loss and may precede phase transitions. Unlike a progress measure, the LLC can detect hidden model changes without requiring prior knowledge of what is changing. Our structural metrics resemble mechanistic progress measures [25], demonstrating that mechanistic analysis can complement geometric analysis by helping to discover stage content.

3. Transformer training

Following Elhage et al. [9] and Olsson et al. [28], we train an autoregressive attention-only transformer with two layers. We use token sequences taken from a subset of the Pile [15, 43]. Full architecture, tokenization, and training details are in Appendix B.

Let f_w denote the transformer language model with parameter w. Our data comprises n length-K *contexts*, $\{S_K^i\}_{i=1}^n$, where each context is a sequence of *tokens* $S_K^i = (t_1^i, \ldots, t_K^i)$. Let $S_{\leq k}^i$ denote the prefix $(t_1^i, \ldots, t_k^i)$ of S_K^i. For $k \leq K-1$ the *per-token empirical loss* is

$$\ell_{n,k}(w) = -\frac{1}{n}\sum_{i=1}^{n} \log f_w(t_{k+1}^i | S_{\leq k}^i). \tag{1}$$

The associated *empirical loss* is

$$\ell_n(w) = \frac{1}{K-1}\sum_{k=1}^{K-1} \ell_{n,k}(w), \tag{2}$$

with the *test loss* $\hat{\ell}$ defined analogously on a held-out set of examples. The corresponding *population loss* $\ell(w)$ is the expectation over some true distribution of contexts.

4. Geometry tracing

We track the evolution of degeneracy in the local geometry of the loss landscape throughout training by estimating the Local Learning Coefficient [LLC; 18, 39] at model checkpoints.

The LLC at a neural network parameter w^*, denoted $\lambda(w^*)$, is a positive scalar measuring the *degeneracy* of the geometry of the population loss ℓ near w^*. The geometry is more degenerate (lower LLC) if there are more ways in which w can be varied near w^* such that $\ell(w)$ remains equal to $\ell(w^*)$. Watanabe [39] studied the *global* learning coefficient and proved that this quantity of the population loss ℓ can be estimated using the empirical loss ℓ_n. Lau et al. [18, see also 14] introduced the *local* learning coefficient along with a scalable LLC estimator based on stochastic-gradient Langevin dynamics [SGLD; 42]. For some examples of the theoretical LLC, see Appendix C.

We generalize Lau et al. [18]'s LLC estimator from *likelihood-based* to *loss-based*. Let w^* be a local minimum of the population loss ℓ. The generalized LLC estimate $\hat{\lambda}(w^*)$ is

$$\hat{\lambda}(w^*) = n\beta \left[\mathbb{E}^{\beta}_{w|w^*,\gamma}[\ell_n(w)] - \ell_n(w^*) \right], \tag{3}$$

where $\mathbb{E}^{\beta}_{w|w^*,\gamma}$ denotes the expectation with respect to the Gibbs posterior

$$p(w; w^*, \beta, \gamma) \propto \exp\left\{ -n\beta\ell_n(w) - \frac{\gamma}{2}||w - w^*||_2^2 \right\},$$

β is an inverse temperature controlling the contribution of the loss, and γ is the localization strength controlling proximity to w^*. Intuitively, the more degenerate the geometry, the more ways there are to vary w near w^* without changing the loss, the easier it is for a sampler exploring the posterior to find points of low loss, the lower $\hat{\lambda}(w^*)$. For details on LLC estimation, see Appendix D.

5. Stage division

We use critical points (that is, plateaus, where the first derivative vanishes) in the LLC curve to define *stage boundaries* that divide training into *developmental stages* (see Appendix E for details).

This approach is motivated by Watanabe's free energy formula [40, Theorem 11] generalized to a local setting by Chen et al. [6]. This gives an asymptotic expansion in the number of samples n of the expected Bayesian free energy of some neighborhood W^* surrounding a local minimum w^* of ℓ,

$$\mathbb{E}_{D_n}[F_n(W^*)] = n\ell(w^*) + \lambda(w^*)\log n + O(\log\log n). \tag{4}$$

The coefficients of the linear and logarithmic terms are the population loss (a negative log likelihood) and the LLC, respectively. This creates a tradeoff between accuracy (ℓ) and degeneracy (λ), and suggests that at certain *critical dataset sizes* the Bayesian posterior will suddenly "jump" from concentrating around one local minima to another. This sequence of discrete jumps is referred to as the *singular learning process* [39, §7.6].

In small models such jumps behave like *phase transitions* in statistical physics [2, 6] and are reflected as sudden changes in the estimated LLC. In larger models the change in estimated LLC is more gradual, and we speak instead of *stages* separated by *stage boundaries* at which the posterior is stably concentrated around a given local minima (see also Appendix A).

The connection between the singular learning process in Bayesian statistics and stagewise development in deep learning is not understood in general, but has been studied in small autoencoders by Chen et al. [6]. This perspective suggests that changes in local population loss geometry, as measured by the LLC, reflect qualitative changes in the network parameter.

6. Stage validation

The LLC estimates (Figure 1; see also Appendix E) reveal five developmental stages over the training of our language transformer. In order to validate that this stage division is meaningful, we search for concomitant changes in the model's input–output behavior and its weights or activations.

We found that stages LM1–LM4 coincide with interesting developments, as documented below and in Figures 2 and 3. We do not claim this list is exhaustive; there may be additional developments we did not detect. For example, we do not rule out the use of skip-trigrams as studied by Olsson et al. [28]. We did not discover significant changes in Stage LM5.

Stage LM1 (0–900 steps). The model learns bigram statistics, which give the optimal next-token prediction given the current token. Figure 2 (top) shows that the average cross entropy between model logits and empirical bigram frequencies (see Appendix F.1) is minimized at the LM1–LM2 boundary, with a value only 0.3 nats above the irreducible entropy of the empirical bigram distribution.

Stage LM2 (900–6.5k steps). The model memorizes common n-*grams* for $n > 2$. We define an n-*gram score* as the ratio of final-position token loss on (1) a baseline set of samples from a validation set truncated to n tokens and (2) a fixed set of common n-grams (see Appendix F.2). Figure 2 (bottom) shows a large improvement in n-gram score for $n = 3, 4$ during LM2.

During this stage the positional embedding becomes structurally important. Figure 2 (middle) shows that during LM2 the test loss for the model with the positional embedding zero-ablated diverges from the test loss of the unablated model (see Appendix F.3). Note that memorizing n-grams for $n > 2$ requires the positional embedding. Previous-token attention among second-layer attention

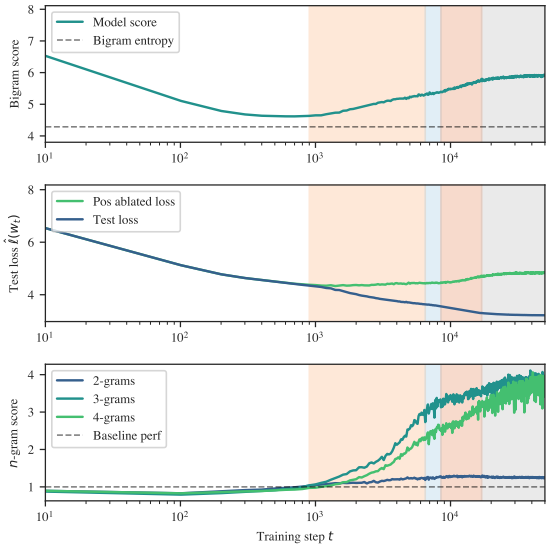


Figure 2: The transformer first learns bigram statistics (LM1, top). At the start of LM2, the positional embedding suddenly becomes useful (middle), enabling behavioral changes such as the learning of common n-grams (bottom).

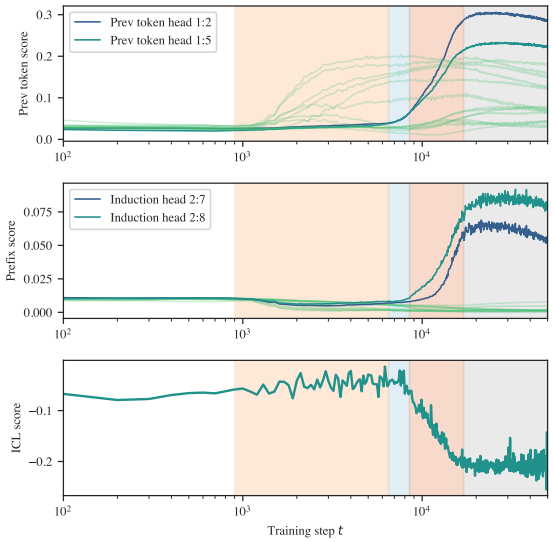


Figure 3: Induction circuit formation begins with previous-token heads (LM3, top, blue lines), followed by induction heads (LM4, middle, blue lines), leading to a drop in ICL score (LM4, bottom). The h^{th} attention head in layer l is indexed as $l : h$. Green lines trace scores for other heads.

heads, shown increasing in the background of Figure 3 (top), may also play a role in implementing n-gram memorization.

In this stage, the heads that eventually become previous-token and induction heads in future stages begin to compose (that is, read from and write to a shared residual stream subspace; see Figure 10 and Appendix F.4). This suggests that the foundations for the induction circuit are laid in advance of any measurable change in model outputs or attention patterns.

Stage LM3 (6.5k–8.5k steps). Layer-1 previous-token heads begin to form, representing the first half of the in-context learning circuit [9, 28]. Figure 3 (top) shows two heads that increasingly attend to the immediately preceding token (see Appendix F.5). During this stage the LLC decreases, suggesting an increase in degeneracy or decrease in model complexity, perhaps related to the interaction between heads. It would be interesting to study this further via mechanistic interpretability.

Stage LM4 (8.5k–17k steps). The model learns to perform in-context learning as captured in the ICL score of Olsson et al. [28] (Figure 3, bottom; Appendix F.6). Structurally, the second half of the induction circuits, layer-2 induction heads, begin to develop. Figure 3 (middle) shows that the prefix-matching score [28] increases for the two heads that become induction heads (see Appendix F.7).

Visualizing behavioral changes. Figure 4 visualizes changes in the model's input–output behavior by comparing model predictions before and after developmental stages and highlighting tokens with the greatest differences.

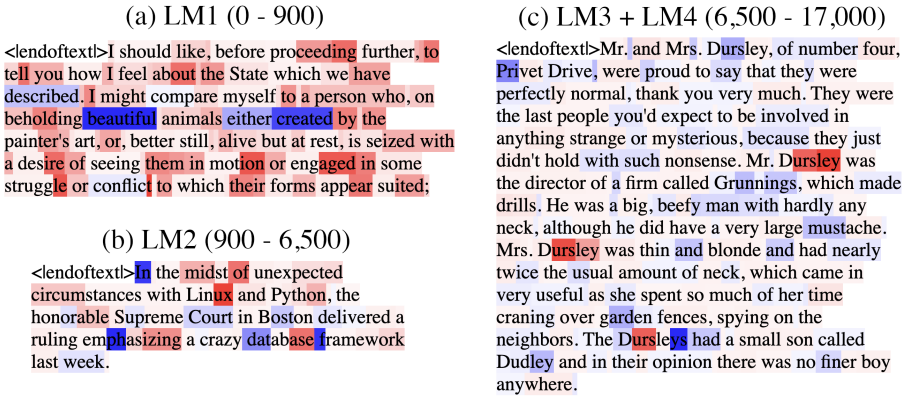


Figure 4: Samples are shown with tokens highlighted to indicate changes in logits during a given range. Red is improved performance (higher logit output for the true next token) and blue is worse. Sample (a): improvement in bigrams (LM1) such as "te/ll, ab/out, des/ire, mot/ion, eng/aged, strugg/le, etc." Sample (b): improvement in common n-grams (LM2) such as "L/in/ux, P/y/th/on, h/on/or/able, S/up/reme, dat/ab/ase, f/ram/ew/ork." Sample (c): development of in-context learning via induction circuits (LM3, LM4), visible in the improved predictions in the word "D/urs/ley" after the first time it appears in the context, as initially observed by [28].

7. Discussion

We have argued that the local geometry of the loss landscape holds the key to understanding stagewise development in modern deep learning. This is motivated by the singular learning process, a model of learning from Bayesian statistics in which there is a clear connection between stagewise development and the local geometry of the population loss as measured by the LLC [6, 39].

Our experiments show that critical points of the estimated LLC divide language model training into five stages that coincide with qualitatively distinct patterns of behavioral and structural development of the model, beginning with the memorization of bigram statistics and common n-grams, and culminating in the emergence of an in-context learning circuit as studied by Olsson et al. [28]. We note that these stages are much less pronounced in the loss curve, which has no critical points.

We are excited about future work aiming to connect the singular learning theory perspective on stagewise learning with the existing literature [e.g., 34]. It is natural to interpret a stage in which the LLC decreases as involving a *simplification* or *compression* of an existing solution, and our observation of such a stage suggests that this is another interesting direction for future research.

In larger models, we expect that the estimated LLC may only detect sufficiently "macroscopic" changes. However, we are optimistic that more refined geometric measures can be developed using the same principles, such as by restricting the LLC to subsets of parameters and to parts of the data distribution. We expect that with future work such tools will be able to provide a more fine-grained picture of structural development, even in large models.

Acknowledgements

We thank Edmund Lau for advice on local learning coefficient estimation. We thank Evan Hubinger and Simon Pepin Lehalleur for helpful conversations. We thank Andres Campero, Zach Furman, and Simon Pepin Lehalleur for helpful feedback on earlier versions of this manuscript.

We thank Google's TPU Research Cloud program for supporting some of our experiments with Cloud TPUs. GW's and JH's work was supported partially by the AI Futures Fellowship. MFR's work was supported partially by private individual sponsors and partially by Timaeus. LC's work was supported by Lightspeed Grants.

References

[1] Joseph Antognini and Jascha Sohl-Dickstein. PCA of high dimensional random walks with comparison to neural network training. In *Advances in Neural Information Processing Systems*, volume 31, pages 10307–10316, 2018.

[2] Vijay Balasubramanian. Statistical inference, Occam's razor, and statistical mechanics on the space of probability distributions. *Neural Computation*, 9(2):349–368, February 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.2.349.

[3] Pierre Baldi and Kurt Hornik. Neural Networks and Principal Component Analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

[4] Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: SGD learns parities near the computational limit. In *Advances in Neural Information Processing Systems*, volume 35, pages 21750–21764, 2022.

[5] Angelica Chen, Ravid Shwartz-Ziv, Kyunghyun Cho, Matthew L. Leavitt, and Naomi Saphra. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in MLMs. In *The Twelfth International Conference on Learning Representations*, 2024.

[6] Zhongtian Chen, Edmund Lau, Jake Mendel, Susan Wei, and Daniel Murfet. Dynamical versus Bayesian phase transitions in a toy model of superposition. Preprint arXiv:2310.06301 [cs.LG], 2023.

[7] Benjamin L. Edelman, Ezra Edelman, Surbhi Goel, Eran Malach, and Nikolaos Tsilivis. The Evolution of Statistical Induction Heads: In-Context Learning Markov Chains. Preprint arXiv:2402.11004 [cs.LG], 2024.

[8] Ronen Eldan and Yuanzhi Li. TinyStories: How small can language models be and still speak coherent English? Preprint arXiv:2305.07759 [cs.CL], 2023.

[9] 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. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021.

[10] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(19):625–660, 2010.

[11] Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports*, 7(1), 2017.

[12] Sara Franceschelli. Morphogenesis, structural stability and epigenetic landscape. In *Morphogenesis: Origins of Patterns and Shapes*, pages 283–293. Springer, 2010.

[13] Simon L Freedman, Bingxian Xu, Sidhartha Goyal, and Madhav Mani. A dynamical systems treatment of transcriptomic trajectories in hematopoiesis. *Development*, 150(11):dev201280, 2023.

[14] Zach Furman and Edmund Lau. Estimating the local learning coefficient at scale. Preprint arXiv:2402.03698 [cs.LG], 2024.

[15] 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. The Pile: an 800GB dataset of diverse text for language modeling. Preprint arXiv:2101.00027 [cs.CL], 2020.

[16] R. Gilmore. *Catastrophe Theory for Scientists and Engineers*. Wiley, 1981.

[17] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. Preprint arXiv:1412.6544 [cs.NE], 2015. Published at ICLR 2015.

[18] Edmund Lau, Daniel Murfet, and Susan Wei. Quantifying degeneracy in singular models via the learning coefficient. Preprint arXiv:2308.12108 [stat.ML], 2023.

[19] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, volume 31, pages 6389–6399, 2018.

[20] Zachary C. Lipton. Stuck in a what? Adventures in weight space. Preprint arXiv:1602.07320 [cs.LG], 2016.

[21] Ben D MacArthur. The geometry of cell fate. *Cell Systems*, 13(1):1–3, 2022.

[22] Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in AlphaZero. *Proceedings of the National Academy of Sciences*, 119(47), 2022.

[23] Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. The hydra effect: Emergent self-repair in language model computations. Preprint arXiv:2307.15771 [cs.LG], 2023.

[24] Neel Nanda and Joseph Bloom. TransformerLens, 2022. URL https://github.com/neelnanda-io/TransformerLens.

[25] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023.

[26] Pascal Jr. Tikeng Notsawo, Hattie Zhou, Mohammad Pezeshki, Irina Rish, and Guillaume Dumas. Predicting grokking long before it happens: A look into the loss landscape of models which grok. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

[27] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024.001, March 2020.

[28] 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, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

[29] Ofir Press, Noah A. Smith, and Mike Lewis. Shortformer: Better language modeling using shorter inputs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5493–5505. Association for Computational Linguistics, 2021.

[30] Maartje EJ Raijmakers, Sylvester Van Koten, and Peter CM Molenaar. On the validity of simulating stagewise development by means of PDP networks: Application of catastrophe analysis and an experimental test of rule-like network performance. *Cognitive Science*, 20(1): 101–136, 1996.

[31] David A Rand, Archishman Raju, Meritxell Sáez, Francis Corson, and Eric D Siggia. Geometry of gene regulatory dynamics. *Proceedings of the National Academy of Sciences*, 118(38): e2109729118, 2021.

[32] Timothy T Rogers and James L McClelland. *Semantic Cognition: A Parallel Distributed Processing Approach*. MIT Press, 2004.

[33] Meritxell Sáez, Robert Blassberg, Elena Camacho-Aguilar, Eric D Siggia, David A Rand, and James Briscoe. Statistically derived geometrical landscapes capture principles of decision-making dynamics during cell fate transitions. *Cell Systems*, 13(1):12–28, 2022.

[34] Andrew M Saxe, James L McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116 (23):11537–11546, 2019.

[35] Maxwell Shinn. Phantom oscillations in principal component analysis. *Proceedings of the National Academy of Sciences*, 120(48):e2311420120, 2023.

[36] René Thom. *Structural Stability and Morphogensis: An Outline of a General Theory of Models*. Advanced Book Classics Series, 1988. Translated by D. H. Fowler from the 1972 original.

[37] C H Waddington. *The Strategy of the Genes: A Discussion of Some Aspects of Theoretical Biology*. Allen & Unwin, London, 1957.

[38] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023.

[39] Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press, 2009.

[40] Sumio Watanabe. *Mathematical Theory of Bayesian Statistics*. CRC Press, 2018.

[41] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

[42] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

[43] Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. Data selection for language models via importance resampling. In *Advances in Neural Information Processing Systems*, volume 36, pages 34201–34227, 2023.

Appendices

We include additional discussion, details, and experiments as follows.

- **Appendix A** discusses additional related work.
- **Appendix B** documents additional details about our transformer architecture, data, and training.
- **Appendix C** provides additional background information on the LLC including examples.
- **Appendix D** documents additional details about our LLC estimation procedure.
- **Appendix E** documents the procedure we use for identifying critical points in the estimated LLC curve (used as stage boundaries).
- **Appendix F** documents additional details about the behavioral and structural metrics we track to investigate the content of the identified stages.
- **Appendix G** documents additional runs of our main experiment, showing that we find roughly the same five stage boundaries across five independent seeds.

Appendix A. Additional related work

Loss landscape geometry. Many authors have used one or two-dimensional slices of the loss landscape to visualize its geometry [10, 17, 19, 20, 26]. These approaches are limited by the fact that a random slice is with high probability a quadratic form associated to nonzero eigenvalues of the Hessian and is thus biased against geometric features that we know are important, such as degeneracy. Indeed, recent works have emphasized the difficulty of meaningfully probing the loss landscape of neural networks [1], even qualitatively. In contrast, the estimated LLC is a principled tool for probing degenerate geometry in a *quantitative* way.

Universality. Olah et al. [27] hypothesize that similar internal structures may form across a wide range of different neural network architectures, training runs, and datasets. Examples include Gabor filters, induction heads, and "backup heads" [23, 27, 38]. We find that our stage discovery methodology reveals five stages across all seeds, with close (but not perfect) agreement on the location of the stage boundaries; see Appendix G. Though we do not expect all aspects of development to be universal—for example, we expect the strength of the bigram stage to depend on the size of the tokenizer—we conjecture that the developmental trajectory will remain macroscopically simple even for much larger models.

Nonlinear dynamics and developmental biology. Neural network training is a stochastic dynamical system, in which the governing potential (the population loss) encodes the structure of the data distribution along with the constraints of the network architecture. It is well-understood in nonlinear dynamics that the local geometry of a potential can give rise to stagewise development of structure in the system [36, 37]. This connection between geometry and stagewise development has been observed in biological systems at significant scale and in the presence of stochasticity [13].

We have emphasized changes in geometry *over a stage* whereas in developmental biology the focus, in the mathematical framework of bifurcation theory, is more on the singular geometry *at stage boundaries* [21, 31, 33]. The relationship between these two points of view is beyond the scope of this paper. For more on the relation between the points of view of Waddington and Thom, see Franceschelli [12].

"Phase transitions" versus "developmental stages." Stage boundaries could alternatively be described as "phases", and developmental stages as "phase transitions." However, these terms retains a false connotation of transitions that are sudden *in time*. While phase transitions are sometimes characterized as "sudden," from a mathematical point of view [6, 16] the important characteristic of a phase transition is that the configuration of a system (or a distribution over such configurations) shifts rapidly from a neighborhood of one critical point of a relevant potential (e.g. a free energy) to another critical point, as a function of some control variable c near some critical value $c \approx c_0$.

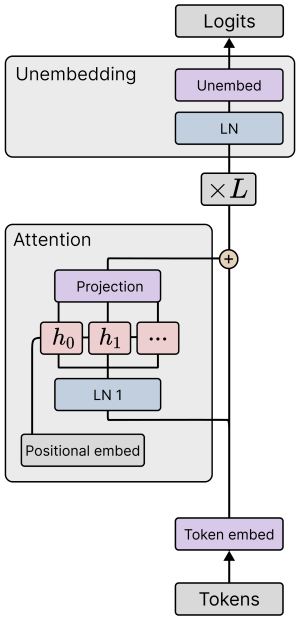
To take one example: in developmental biology there are carefully modeled phase transitions which take place over *days* in real time. The relevant control variable in these cases is an inferred *developmental time* $\tau = \tau(t)$ [13].

To avoid confusion, we therefore prefer to borrow terminology from biology, referring to "developmental stages" and "stage boundaries" instead of "phase transitions" and "phases."

Appendix B. Transformer training details

B.1. Model

The language model architecture we consider is an decode-only transformer with $L = 2$ "attention-only" layers (without MLPs). We use a context length of 1024, a residual stream dimension of $d_{model} = 256$, and $H = 8$ attention heads per layer. We include layer normalization. We also used a learnable Shortformer positional embedding [29]. The resulting models have a total of $d = 3,355,016$ parameters. We used an implementation provided by TransformerLens [24].



Component	Parameters
Token Embedding Weights	$1,280,000$
Positional Embedding Weights	$262,144$
Layer 1 Layer Norm Weights	256
Layer 1 Layer Norm Bias	256
Layer 1 Attention Query Weights	$65,536$
Layer 1 Attention Key Weights	$65,536$
Layer 1 Attention Value Weights	$65,536$
Layer 1 Attention Output Weights	$65,536$
Layer 1 Attention Query Bias	256
Layer 1 Attention Key Bias	256
Layer 1 Attention Value Bias	256
Layer 1 Attention Output Bias	256
Layer 2 Layer Norm Weights	256
Layer 2 Layer Norm Bias	256
Layer 2 Attention Query Weights	$65,536$
Layer 2 Attention Key Weights	$65,536$
Layer 2 Attention Value Weights	$65,536$
Layer 2 Attention Output Weights	$65,536$
Layer 2 Attention Query Bias	256
Layer 2 Attention Key Bias	256
Layer 2 Attention Value Bias	256
Layer 2 Attention Output Bias	256
Final Layer Norm Weights	256
Final Layer Norm Bias	256
Unembedding Weights	$1,280,000$
Unembedding Bias	$5,000$

Figure 5: **Attention-only transformer** with Shortformer position-infused attention and pre-layer norm. The model model has a total of 3,355,016 parameters.

B.2. Training

The models are trained on a single epoch over $50,000$ steps on $\sim$5 billion tokens using a resampled subset of the Pile [15, 43] using a batch size of 100. A snapshot was saved every 10 steps for a total of 5000 checkpoints, though a majority of analysis used checkpoints every 100 steps. The training time was around 6 GPU hours per model on an A100. Additional seeds (Appendix G) were trained on v4 TPUs at around 1.5 TPU hours per model.

Training was conducted on the first 10 million lines of the DSIR-filtered Pile [15, 43] but did not exhaust all 10 million lines. The model was subject to weight decay regularization, without the application of dropout. We did not employ a learning rate scheduler throughout the training process.

B.3. Tokenization

For tokenization, we used a truncated variant of the GPT-2 tokenizer that cut the original vocabulary of 50,000 tokens down to 5,000 [8] to reduce the size of the model. We think this may contribute to the prominence of the the plateau at the end of LM1: the frequency of bigram statistics depends on your choice of tokens, and a larger tokenizer leads to bigrams that are individually much less frequent.

Table 1: Summary of hyperparameters and their values for transformer training

Hyperparameter	**Category**	**Description/Notes**	**Value**
n	Data	# of training samples	$5,000,000$
T	Data	# of training steps	$50,000$
N_{test}	Data	# of test samples	512
Tokenizer Type	Data	Type of Tokenizer	Truncated GPT-2 Tokenizer
D	Data	Vocabulary size	5,000
K	Data	Context size	1,024
L	Model	# of layers in the model	2
H	Model	# of heads per layer	8
d_{mlp}	Model	MLP hidden layer size	N/A
d_{embed}	Model	Embedding size	256
d_{head}	Model	Head size	32
seed	Model	Model initialization	1
m	Training	Batch Size	100
Optimizer Type	Optimizer	Type of optimizer	AdamW
η	Optimizer	Learning rate	0.001
λ_{wd}	Optimizer	Weight Decay	0.05
$\beta_{1,2}$	Optimizer	Betas	$(0.9, 0.999)$

Appendix C. Examples of the LLC

The LLC has some similarity to an effective parameter count. If the population loss ℓ looks like a quadratic form near w^* then $\lambda(w^*) = \frac{d}{2}$ is half the number of parameters, which we can think of as d contributions of $\frac{1}{2}$ from every independent quadratic direction. If there are only $d-1$ independent quadratic directions, and one coordinate w_i such that small variations in w_i near w_i^* do not change the model relative to the truth (it is "unused") then $\lambda(w^*) = \frac{d-1}{2}$. However, while every unused coordinate reduces the LLC by $\frac{1}{2}$, changing a coordinate from quadratic w_i^2 to quartic w_i^4 (increasing its *degeneracy* while still "using" it) reduces the contribution to the LLC from $\frac{1}{2}$ to $\frac{1}{4}$.

As a source of intuition, we provide several examples of exact LLCs:

- $\ell(w_1, w_2, w_3) = aw_1^2 + bw_2^2 + cw_3^2$ with $a, b, c > 0$. This function is nondegenerate, and $\lambda(0,0,0) = \frac{1}{2} + \frac{1}{2} + \frac{1}{2} = \frac{3}{2}$. This is independent of a, b, c. That is, the LLC λ does *not measure curvature*. For this reason, it is better to avoid an intuition that centers on "basin broadness" since this tends to suggest that lowering a, b, c should affect the LLC.
- $\ell(w_1, w_2, w_3) = w_1^2 + w_2^2 + 0$ in $\mathbb{R}^3$ is degenerate, but its level sets are still submanifolds and $\lambda(0,0,0) = \frac{1}{2} + \frac{1}{2}$. In this case the variable z is unused, and so does not contribute to the LLC.
- $\ell(w_1, w_2, w_3) = w_1^2 + w_2^4 + w_3^4$ is degenerate and its level sets are, for our purposes, not submanifolds. The singular function germ $(\ell, 0)$ is an object of algebraic geometry, and the appropriate mathematical object is not a *manifold* or a *variety* but a *scheme*. The quartic terms contribute $\frac{1}{4}$ to the LLC, so that $\lambda(0,0,0) = \frac{1}{2} + \frac{1}{4} + \frac{1}{4} = 1$. The higher the power of a variable, the greater the degeneracy and the lower the LLC.

For additional examples (still far from exhaustive), see Figure 6. While nondegenerate functions can be locally written as quadratic forms by the Morse Lemma (and are thus qualitatively similar to the approximation obtained from their Hessians), there is no simple equivalent for degenerate functions, such as the population log likelihood of singular models (or population losses of neural networks). We refer the reader to Watanabe [39] and Lau et al. [18] for more discussion.

Figure 6: Toy two-dimensional loss landscapes show that the lower the LLC, the more the local geometry deviates from quadratic. *Left*: A quadratic potential $\ell_1(w_1, w_2) = w_1^2 + w_2^2$, for which the LLC is $\lambda_1(0,0) = d/2 = 1$, maximal for two dimensions. *Middle-left*: A quartic potential $\ell_2(w_1, w_2) = w_1^4 + w_2^4$, for which the LLC is $\lambda_2(0,0) = 1/2$. *Middle-right*: An even simpler potential $\ell_3(w_1, w_2) = w_1^2 w_2^4$, for which $\lambda_3(0,0) = 1/4$. Hessian-derived metrics cannot distinguish between this geometry and the preceding quartic geometry. *Right*: A qualitatively distinct potential $\ell_4(w_1, w_2) = (w_1 - 1)^2(w_1^2 + w_2^2)^4$ [18] with the same LLC at the origin, $\lambda_4(0,0) = 1/4$.

Appendix D. LLC estimation details

D.1. Estimating LLCs with SGLD

We follow Lau et al. [18] in using SGLD to estimate the expectation value of the loss in the estimator of the LLC. For a given choice of weights w^* we sample C independent chains with T_{SGLD} steps per chain. Each chain c is a sequence of weights $\{w_\tau^{(c)}\}_{\tau=1}^{T_{\text{SGLD}}}$. From these samples, we estimate the expectation $\mathbb{E}^\beta_{w|w^*,\gamma}[\mathcal{O}(w)]$ of an observable $\mathcal{O}$ by

$$\frac{1}{CT_{\text{SGLD}}} \sum_{c=1}^{C} \sum_{\tau=1}^{T_{\text{SGLD}}} \mathcal{O}(w_\tau^{(c)}), \tag{5}$$

with an optional burn-in period. Dropping the chain index c, each sample in a chain is generated according to:

$$w_{\tau+1} = w_\tau + \Delta w_\tau, \tag{6}$$

$$w_1 = w^*, \tag{7}$$

where the step Δw_τ comes from an SGLD update

$$\Delta w_\tau = \frac{\epsilon}{2}\left(\beta n \nabla \ell_m^{(\tau)}(w_\tau) + \tfrac{\gamma}{2}\left(w_\tau - w^*\right)\right) + \mathcal{N}(0,\epsilon)\,. \tag{8}$$

In each step τ we sample a mini-batch of size m and the associated empirical loss, denoted $\ell_m^{(\tau)}$, is used to compute the gradient in the SGLD update. We note that LLC estimator defined in (3) uses the expectation $\mathbb{E}^\beta[\ell_n(w)]$ which in the current notation means we should take $\mathcal{O}(w)$ to be $\ell_n(w)$. For computational efficiency we follow Lau et al. [18] in recycling the mini-batch losses $\ell_m(w_\tau^{(c)})$ computed during the SGLD process. That is, we take $\mathcal{O} = \ell_m^{(\tau)}$ rather than $\mathcal{O} = \ell_n$.

More detailed results for our main experiments are provided in Appendix E.

D.2. LLC estimates away from local minima

Our methodology for detecting stages is to apply LLC estimation to compute $\hat{\lambda}(w^*)$ at neural network parameters $w^* = w_t$ across training. In the typical case these parameters will *not* be local minima of the population loss, violating the theoretical conditions under which $\hat{\lambda}$ is a valid estimator of the true local learning coefficient [18].

This makes it *a priori* quite surprising that the SGLD-based estimation procedure works at all, as away from local minima, one might expect the chains to explore directions in which the loss decreases. Beyond that, critical points in the $\hat{\lambda}(w_t)$ curve do correspond to meaningful changes in the mode of development in our experiments.

This raises an interesting theoretical question: is there a notion of *stably evolving equilibrium* in the setting of neural network training, echoing some of the ideas of Waddington [37], such that the LLC estimation procedure is effectively giving us the local learning coefficient of a different potential to the population loss, a potential for which the current parameter actually *is* at a critical point? We leave addressing this question to future work. However at the moment we must admit that our methodology goes beyond what is justified by the theoretical foundations.

D.3. LLC estimation for the transformer language model

For LLC estimation, we use SGLD to sample 20 independent chains with 200 steps per chain and 1 sample per step, at a temperature $\beta = 1/\log(m)$, where $m = 100$ is the size of the batch (the maximum size that would fit in memory). We used $\epsilon = 0.001, \gamma = 100$. Estimating the local learning coefficient across all checkpoints took around 200 GPU hours on a single A100. For additional runs (Appendix G) we ran fewer chains, bringing the time down to about 2 TPU hours per training run.

We sampled a separate set of 1 million lines (lines 10m-11m) from the DSIR filtered Pile, denoted as D_{sgld}. The first 100,000 lines from this SGLD set (lines 10m-10.1m) were used as a validation set. The sampling of batches for SGLD mirrored the approach taken during the primary training phase. Each SGLD estimation pass was seeded analogously, so, at different checkpoints, the SGLD chains encounter the same selection of batches and injected Gaussian noise.

Table 2: Hyperparameters for Estimating the Local Learning Coefficient for Language Models.

Hyperparameter	**Category**	**Description/Notes**	**1-Layer**	**2-Layer**
C	Sampler	# of chains	20	
T_{SGLD}	Sampler	# of SGLD steps / chain	200	
ϵ	SGLD	Step size	0.003	0.001
$\tilde{\gamma} = \epsilon\gamma/2$	SGLD	Localization strength	300	200
$\tilde{\beta} = \epsilon\beta/2n$	SGLD	Inverse temperature	0.0000217147	
m	SGLD	(Default: $\beta^* = \frac{1}{\log n}$) The size of each SGLD batch	100	
μ	Data	Dataset size for gradient minibatches	13m	

Appendix E. Stage division details

To identify stage boundaries, we look for plateaus in the LLC: checkpoints at which the slope of $\hat{\lambda}(w_t)$ over t vanishes. To mitigate noise in the LLC estimates, we first fit a Gaussian process with some smoothing to the LLC-over-time curve. Then we numerically calculate the slope of this Gaussian process with respect to $\log t$. The logarithm corrects for the fact that the learning coefficient, like the loss, changes less as training progresses. We identify stage boundaries by looking for checkpoints at which this estimated slope equals zero. The results of this procedure are depicted in Figure 7.

At a local minima or maxima of the estimated LLC curve identifying a plateau from this estimated slope is straightforward, since the derivative crosses the x-axis. However at a saddle point, the slope may not exactly reach zero, so we have to specify a "tolerance" for the absolute value of the derivative, below which we treat the boundary as an effective plateau.

In this case, we additionally require that the plateau be at a local minimum of the absolute first derivative. Otherwise, we may identify several adjacent points as all constituting a stage boundary.

To summarize, identifying stage boundaries is sensitive to the following choices: the intervals between checkpoints, the amount of smoothing, whether to differentiate with respect to t or $\log t$, and the choice of tolerance. However, once a given choice of these hyperparameters is fixed, stages can be *automatically* identified, without further human judgment.

Figure 7 displays the test loss and LLC curves from Figure 1 in addition to the weight norm over time and associated slopes. Stage boundaries coincide with where the slope of the local learning coefficient crosses zero, that is, where there is a plateau in the LLC.

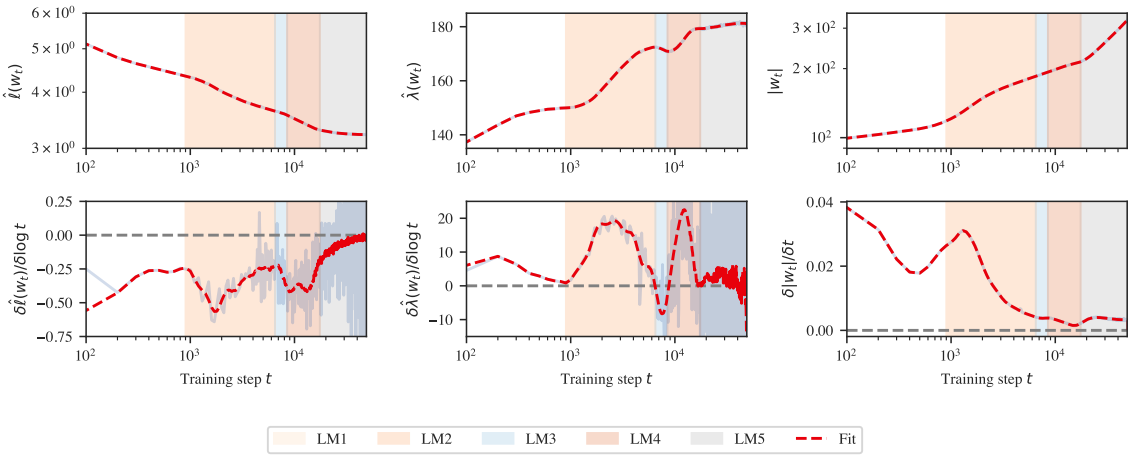


Figure 7: A more detailed version of Figure 1. *Top*: Loss, LLC, and weight norm, along with an overlaid Gaussian process fit to these curves (red dotted lines). *Bottom*: Associated slopes, both numerically estimated finite differences (transparent blue) and of the Gaussian process (red dotted lined). Note that stage LM5 may be subdivided into further stages. However, the noise in LLC estimates late in training is high, so we do not draw any conclusions from this.

Appendix F. Stage validation details

F.1. Bigram score

We empirically estimate the conditional bigram distribution by counting instances of bigrams over the training data. From this, we obtain the conditional distribution $\tilde{q}(t'|t)$, the likelihood that a token t' follows t. The *bigram score* B_k^S at index k of an input context S is the cross entropy between the model's predictions $p(t_{k+1}|t_k)$ at that position and the empirical bigram distribution,

$$B_k^S = -\sum_{i=1}^{d_{\text{vocab}}} \tilde{q}(t_{k+1}^{(i)}|t_k) \log p(t_{k+1}^{(i)}|t_k), \tag{9}$$

where the $t_{k+1}^{(i)}$ range over the possible second tokens from the tokenizer vocabulary. From this we obtain the *average bigram score*

$$\bar{B} = \frac{1}{n}\sum_{i=1}^{n} B_{k_i}^{S_i}, \tag{10}$$

where we take fixed random sequences of k_i and S_i for $1 \leq i \leq n = 5,000$, which is displayed over training in Figure 2. This is compared against the best-achievable bigram score, which is the bigram distribution entropy itself, averaged over the validation set.

F.2. n-gram scores

In stage LM2 we consider n-grams, which are sequences of n consecutive tokens, meaning 2-grams and bigrams are the same. Specifically, we consider *common* n-grams, which is defined heuristically by comparing our 5,000 vocab size tokenizer with the full GPT-2 tokenizer. We use the GPT-2 tokenizer as our heuristic because its vocabulary is constructed iteratively by merging the most frequent pairs of tokens.

We first tokenize the tokens in the full GPT-2 vocabulary to get a list of 50,257 n-grams for various n. The first 5,000 such n-grams are all 1-grams, after which 2-grams begin appearing, then 3-grams, 4-grams, and so on (where 2-grams and 3-grams may still continue to appear later in the vocabulary). We then define the set of common n-grams as the first 1,000 n-grams that appear in this list for a fixed n, $n \geq 2$.

If we track the performance on n-grams and see it improve, we may ask whether this is simply a function of the model learning to use more context in general, rather than specifically improving on the set of n-grams being tracked. We measure performance against this baseline by defining an *n-gram score*. For a fixed n, we obtain the average loss ℓ^n_{gram} of the model on predicting the final tokens of our set of 1,000 n-grams and also obtain the average loss ℓ^n_{test} of the model on a validation set at position n of each validation sequence. The n-gram score is then defined to be $\ell^n_{\text{test}}/\ell^n_{\text{gram}}$.

F.3. Positional embedding

In Figure 8, we measure the effect of the positional embedding on model performance by comparing the model's performance at particular context positions on a validation set over the course of training against performance on the same validation set but with the positional embedding zero-ablated. The full context length is 1024, and we measure test loss at positions 1, 2, 3, 5, 10, 20, 30, 50, 100, 200, 300, 500, and 1000. In the transition from stage LM1 to LM2, the model begins using the learnable

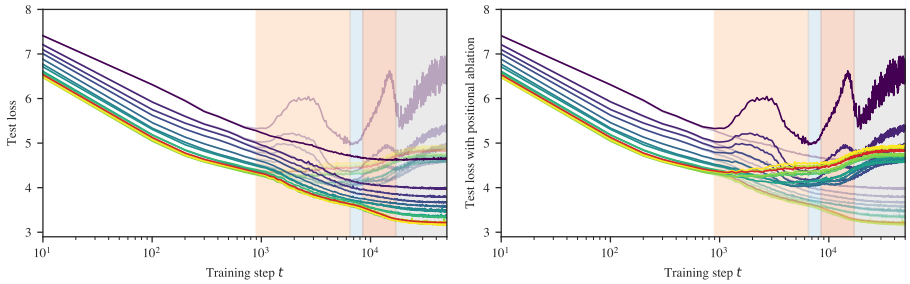


Figure 8: The model learns to start using the positional encoding in LM2, when the performance starts to worsen when ablating the positional encoding. In both plots, earlier token positions are colored more purple, while later token positions are more yellow, and the overall mean loss is colored in red. Both sets of per-token losses are shown in both graphs for ease of comparison. *Left:* original test loss is emphasized. *Right:* test loss with the positional embedding ablated is emphasized.

positional embedding to improve performance. The difference between test loss with and without the positional ablation is negligible at all measured positions until the LM1–LM2 boundary.

Structurally, we might predict that the positional embeddings should organize themselves in a particular way: in order to understand relative positions, adjacent positions should be embedded close to each other, and far-away positions should be embedded far apart.

In Figure 9, we examine the development of the positional embedding itself over time from two angles. The first is to take the embeddings of each position in the context and to run PCA on those embeddings. The result is that as training progresses, the positional embedding PCAs gradually resolve into Lissajous curves, suggesting that the positional embeddings might look like a random walk [1, 35]. However, if we look to the explained variance, we see that it grows very large for PC1, reaching 94.2% at training step 6400. This is much higher than we would expect for Brownian motion, where we expect to see about 61% explained variance in PC1 [1].

The second perspective we use is to look at how the magnitudes of positional embeddings over the context length develop. In this case, we observe that the magnitudes seem to have a fairly regular structure. In conjunction with the PCAs and explained variance, we might infer that the positional embeddings look approximately like a (possibly curved) line in $d_{\text{model}} = 256$ dimensional space. A positional embedding organized in this way would make it easier for an attention head to attend to multiple recent tokens, which is necessary if a single head is to learn n-grams.

F.4. Composition scores

Let W_Q^h, W_K^h, W_V^h be the query, key, and value weights of attention head h respectively. There are three types of composition between attention heads in transformer models in Elhage et al. [9]:

- Q-Composition: the query matrix W_Q^h of an attention head reads in a subspace affected by a previous head
- K-Composition: the key matrix W_K^h of an attention head reads in a subspace affected by a previous head
- V-Composition: the value matrix W_V^h of an attention head reads in a subspace affected by a previous head

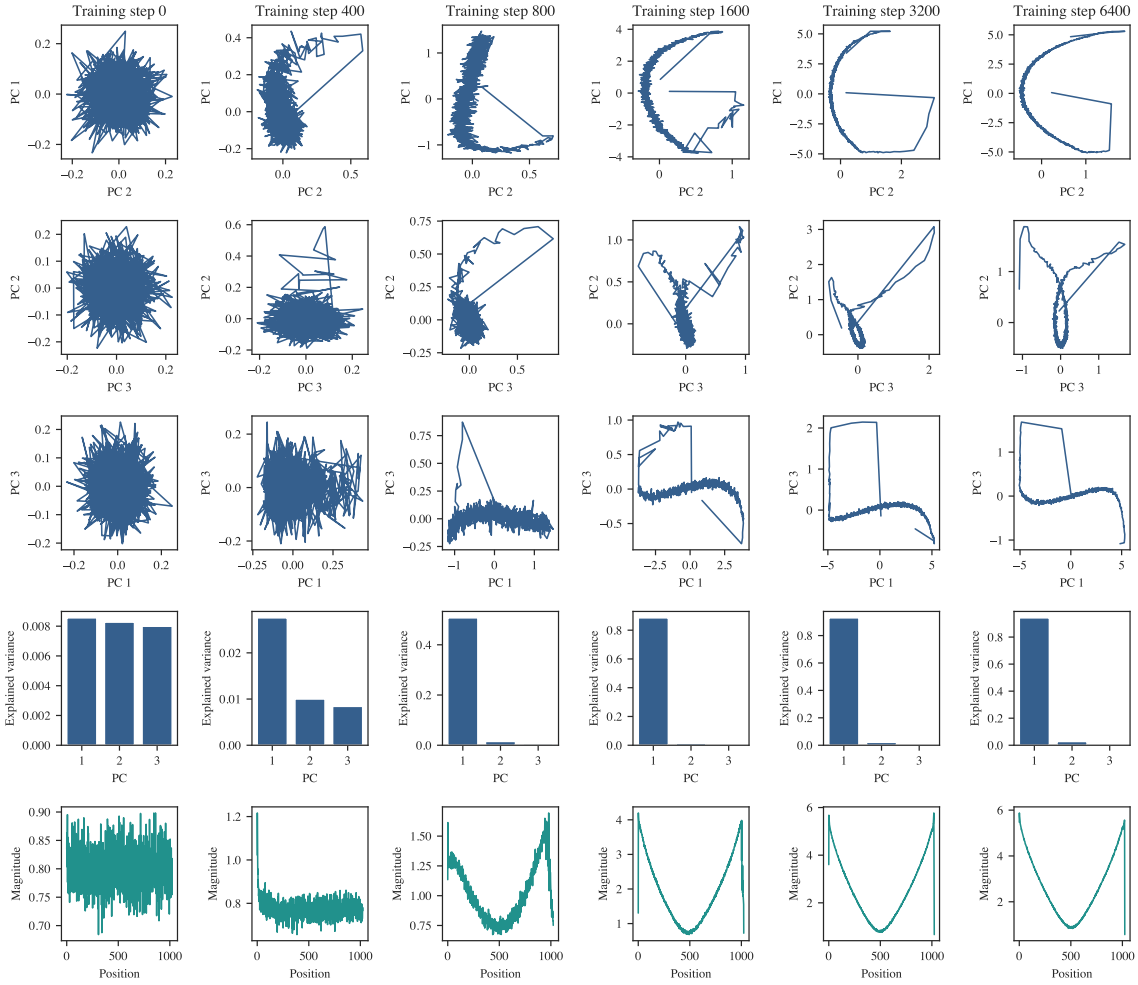


Figure 9: Columns progress through training time at training steps 0, 400, 800, 1600, 3200, and 6400. The first three rows are plots of the first three principle components of PCA on the positional embedding weights, while the fourth row shows the explained variance for each of the principal components. The fifth row plots the magnitude of the embedding of each position in the context length of 1024.

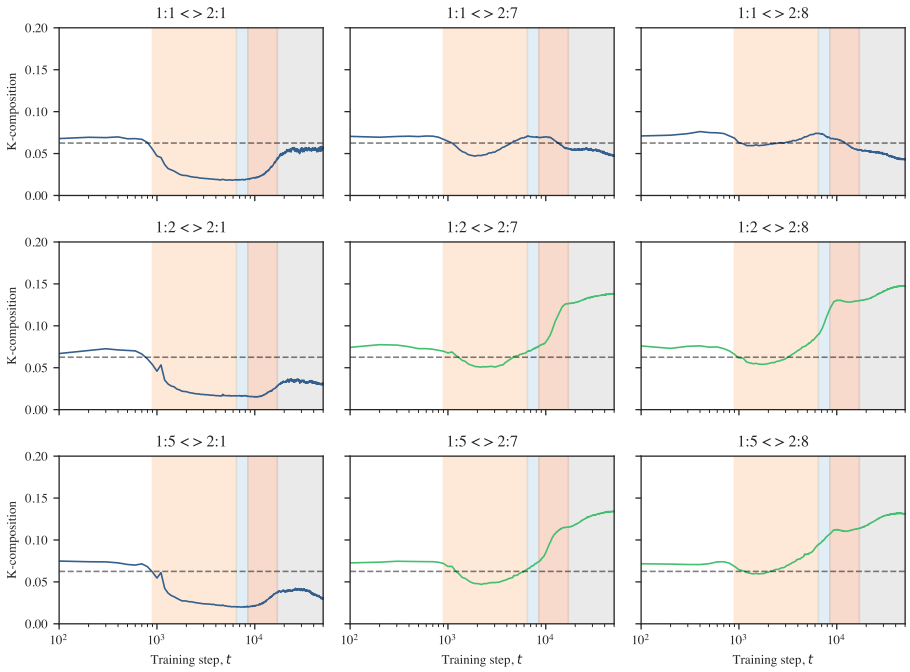


Figure 10: The K-composition scores [9] between first- and second-layer attention heads. The hth attention head in layer l is indexed by $l : h$. The attention heads that eventually become previous token heads are $h = 2, 5$ in layer 1 (subplot rows 2 and 3), and the attention heads that eventually become induction heads are $h = 7, 8$ in layer 2 (subplot columns 2 and 3). The attention heads $1 : 1$ and $2 : 1$ are included for comparison. The induction heads $2 : 7$ and $2 : 8$ begin K-composing with first-layer heads near the start of stage LM2. They continue to compose with the previous token heads in stages LM3 and LM4 (highlighted in green) while their K-composition scores drop with other attention heads in layer 1 in later stages.

If W_O^h is the output matrix of an attention head, then $W_{QK}^h = W_Q^{h\,T} W_K^h$ and $W_{OV}^h = W_O^h W_V^h$. The composition scores are

$$||MW_{OV}^{h1}||_F / (||M||_F ||W_{OV}^{h_1}||_F) \tag{11}$$

Where $M = W_{QK}^{h_2\,T}$, $M = W_{QK}^{h_2}$, and $M = W_{OV}^{h_2}$ for Q-, K-, and V-Composition respectively. See Figure 10 for K-composition scores over time between attention heads in the induction circuits.

F.5. Previous-token matching score

The *previous-token matching score* is a structural measure of induction head attention. It is the attention score given to $[A]$ by an attention head at $[B]$ in the sequence $\ldots [A][B]$ (i.e., how much the head attends to the immediately preceding token).

We compute this score using a synthetic data generating process, generating 10k fixed random sequences with length between 16 and 64. The first token is a special "beginning of string" token, and the remaining tokens are uniformly randomly sampled from other tokens in the vocabulary.

For each sample in this synthetic data set, we measure the attention score that an attention head gives to the previous token when at the last token in the sequence. These scores are averaged across

the dataset to produce the previous-token matching score for that attention head at a given checkpoint. The progression of previous-token matching scores over time can be seen in Figure 3.

F.6. In-context learning score

The *in-context learning score* is a behavioral measure of the relative performance of a model later in a sequence versus earlier in the sequence. We follow a similar construction as Olsson et al. [28], where we take the loss at the 500th token minus the loss at the 50th token, so that a more negative score indicates better performance later in the sequence.

$$\mathrm{ICL}_{k_1:k_2}(w) = \hat{\ell}_{n,\,k_1}(w) - \hat{\ell}_{n,\,k_2}(w). \tag{12}$$

This is then averaged over a 100k-row validation dataset. The performance of the language model over the course of training can be seen at the bottom right of Figure 3.

F.7. Prefix matching score

The *prefix matching score* from Olsson et al. [28] is defined similarly to the previous-token matching score. Given a sequence $[A][B]\ldots[A]$, the prefix matching score of a particular attention head is how much the attention head attends back to the first instance of $[A]$ when at the second instance of $[A]$.

We compute this score using a synthetic data-generating process. We first generate 10k fixed random sequences of length 128. The first token is always a special "beginning of string" token and the $[A]$ and $[B]$ tokens are selected and placed randomly. One $[A]$ token is placed in the first half of the sequence, the other is placed in the second half, and the $[B]$ token is placed directly after the first $[A]$ token. The remaining tokens are randomly sampled from the tokenizer vocabulary, excluding the $[A]$, $[B]$, and beginning of string tokens.

For each sample in this synthetic dataset, we measure the attention score that each attention head assigns to the earlier instance of $[A]$ from the latter instance of $[A]$. These scores are averaged across the dataset to produce the prefix matching score for that attention head at a given checkpoint. The progression of prefix matching scores over time can be seen in Figure 3.

Appendix G. Repeating the stage division experiments for additional models

Figure 11 shows loss and LLC curves for five seeds (differing in model initialization and batch schedule). In each seed, LLC estimation reveals stage LM1–LM4. In three of the five seeds, stage LM5 is subdivided into two additional stages.

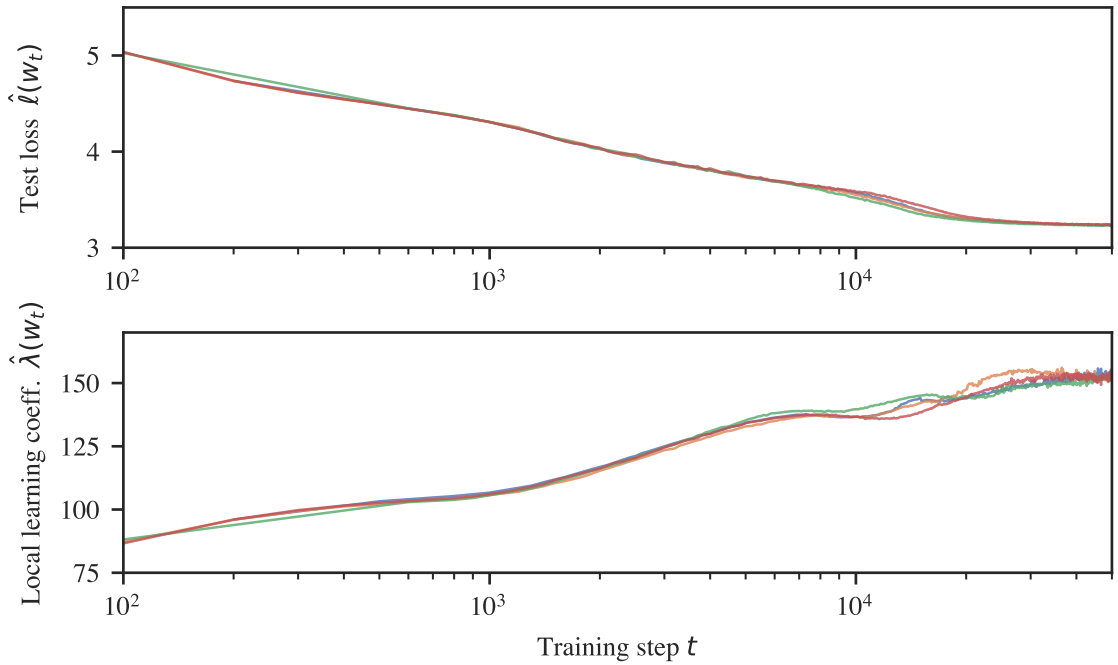


Figure 11: Figure 1 for multiple seeds. The LLC reveals a consistent set of stages across five seeds. Late-training behavior shows more variance across seeds.