When Are Two Networks the Same? Tensor Similarity for Mechanistic Interpretability

ML Nissen Gonzalez[*1,2] **Melwina Albuquerque**[1] **Laurence Wroe**[1] **Jacob Meyer Cohen**[1,3]
Ward Gauderis[4] **Logan Riggs Smith**[†5] **Thomas Dooms**[5]

[1]MARS [2]Heidelberg University [3]Stanford University [4]Vrije Universiteit Brussel [5]Independent

Abstract

Mechanistic interpretability aims to break models into meaningful parts, and verifying that two such parts implement the same computation is a prerequisite. Existing similarity measures evaluate either empirical behaviour, leaving them blind to out-of-distribution mechanisms, or basis-dependent parameters, disregarding weight-space symmetries. We introduce tensor similarity, a weight-based metric for multilinear models, whose weights decompose into tensor networks of special structure. Not every network belongs to this class, but these models train and perform like standard ones. The metric is invariant to weight-space symmetries and captures global functional equivalence including cross-layer mechanisms, and we propose and motivate an efficient recursive algorithm to compute it. Empirically, tensor similarity tracks functional training dynamics, such as grokking and backdoor insertion, with higher fidelity than existing metrics. This turns measuring similarity and verifying faithfulness into an algebraic problem rather than one of empirical approximation.

1 INTRODUCTION

A central challenge in interpreting neural networks is verifying that decomposed [Cunningham et al., 2023] or extracted [Kim et al., 2018, Bau et al., 2017] components preserve the computation of the original model, a prerequisite for faithful mechanistic interpretability [Sharkey et al., 2025]. Current evaluations rely almost entirely on input-based similarity, comparing outputs or activations [Conmy et al., 2023, Hanna et al., 2024]. Being dataset dependent, they can miss behaviour that only appears away from the data they are evaluated on [Olah, 2025].

We argue that a sensible metric should satisfy three criteria. It should be *dataset independent*, depending on the model rather than on sampled data, so that robustness claims, such as ruling out backdoors [Marks et al., 2025] that trigger only on rare inputs, hold over the full distribution. It should be *symmetry invariant*, unaffected by transformations that change a model's parameters but leave the learned function those parameters determine unchanged, whether a mechanism is spread across neurons or distributed across layers [Ameisen et al., 2025]. And it should be *tractably scalable*, computable without the sampling bounds that preclude its use as an optimisation target. Existing metrics fail at least one criterion: behavioural similarity [Kornblith et al., 2019, Kriegeskorte et al., 2008] compares input-output pairs and is blind out of distribution (OOD), so it misses backdoors by construction; causal-abstraction methods and interchange-intervention search [Beckers et al., 2020, Geiger et al., 2021, 2024] give a formal, interventional test of whether a network realises a hypothesised high-level algorithm, but they evaluate it over a data distribution and align a model to a specified abstraction rather than comparing two models directly, so they are not dataset independent; and matrix similarity, which concatenates the weights of a parametric network into a vector and takes an inner product, is not invariant under symmetry transformations that change those parameters but not the function they determine.

We propose *tensor similarity*, a weight-based metric that satisfies all three criteria, at the cost of using tensor-based architectures [Pearce et al., 2025, Dooms et al., 2025a]. These networks train and perform like standard neural architectures, but they form a distinct, multilinear class: not every neural network can be expressed as a multilinear model, and the metric applies to this class rather than to arbitrary networks. The metric extends cosine similarity from vectors to such models by comparing their weight tensors directly,

[*]ML Nissen Gonzalez, Melwina Albuquerque, and Laurence Wroe contributed equally.

[†]Logan Riggs Smith and Thomas Dooms share senior authorship. Correspondence to `logansmith5@gmail.com` and `doomsthomas@gmail.com`.

and the comparison can be set up so that it computes the expected inner product of the outputs of two models fed the same Gaussian inputs, requiring no data. It is tractable for multilinear models that themselves decompose as tensor networks, such as deep multilinear models built from stacked bilinear layers, or the bilinear attention transformers described below. Crucially, the weights of these models decompose into tensor networks of special structure: a bilinear layer is a CP decomposition, attention a more intricate factorisation, and stacking layers yields a tree tensor network [Cohen et al., 2015, Levine et al., 2019]. Tensor networks in turn relate directly to arithmetic and probabilistic circuits when the latter are largely restricted to sum and product operations [Loconte et al., 2025].

The paper has two parts. Theoretically, we cover the architectural changes that make the metric computable, the construction that makes it invariant to weight symmetries, and an efficient recursive algorithm. Empirically, we show that tensor similarity exposes functional changes that other methods miss: we track backdoor insertion and grokking-like phase transitions in modular arithmetic [Nanda et al., 2023] in the main text, and report catastrophic forgetting in vision and training dynamics in language [Hoogland et al., 2025, Olsson et al., 2022] in the supplementary material. Our code is available at github.com/tdooms/tensor-similarity.

2 THEORY

This paper studies polynomial (or multilinear) variants of standard neural components that use multiplicative gating rather than ordinary non-linear activation functions [Dauphin et al., 2017]. The similarity measure we develop is defined for this multilinear class specifically, not for networks with generic non-linearities. We introduce the underlying architectures and then show how the proposed similarity measure handles their weight-space symmetries. The discussion assumes familiarity with tensor algebra and the symmetric group; full derivations are in Appendix B.

2.1 DEFINING MULTILINEAR MODELS

We build on work using bilinear and tensor-based architectures as intrinsically interpretable models, which enable formal weight-based analysis via the learned weight tensors while remaining competitive with standard architectures [Pearce et al., 2025, Dooms et al., 2025a], within a longer tradition of using tensor decompositions to study deep learning [Cohen et al., 2015].

Bilinear layers. The core building block is a GLU [Shazeer, 2020] without gate, called a bilinear layer. It maps a lifted input $\tilde{\mathbf{x}} = (1, \mathbf{x}) \in \mathbb{R}^{d+1}$ to $\mathbb{R}^K$ as

$$\mathcal{A}(\mathbf{x}) \;=\; \mathbf{D}\big((\mathbf{L}\tilde{\mathbf{x}}) \odot (\mathbf{R}\tilde{\mathbf{x}})\big), \tag{1}$$

where $\mathbf{L}, \mathbf{R} \in \mathbb{R}^{r\times(d+1)}$, $\mathbf{D} \in \mathbb{R}^{K\times r}$, r is the hidden dimension, and $\odot$ is the elementwise (Hadamard) product. Equivalently $\mathcal{A}(\mathbf{x})_i = \sum_{j,k}\sum_h D_{ih}L_{hj}R_{hk}\tilde{x}_j\tilde{x}_k$, a degree-2 polynomial in $\mathbf{x}$.

Bilinear attention. Attention can be made multilinear in several ways; we replace the softmax pattern with the elementwise product of two linear attention patterns, $\mathcal{A}(\mathbf{X}) = (\mathbf{X}^\top\mathbf{Q}_L^\top\mathbf{K}_L\mathbf{X}) \odot (\mathbf{X}^\top\mathbf{Q}_R^\top\mathbf{K}_R\mathbf{X})$ for embedded tokens $\mathbf{X} \in \mathbb{R}^{\text{seq}\times d}$; with a causal mask and rotary positional embeddings it is a degree-5 polynomial [Riggs, 2026].

Multilinear and deep multilinear models. Both layers can be written as a single contraction of a weight tensor $\mathbf{A} \in \mathbb{R}^K \otimes^n \mathbb{R}^{d+1}$ with n copies of the lifted input, $\mathcal{A}(\mathbf{x}) = \mathbf{A} \cdot_{\text{in}} \otimes^n \tilde{\mathbf{x}}$, a polynomial of order n; we call these *multilinear models*. Layering them, $\mathcal{A} = \circ_{i=1}^{l}\mathcal{A}_i$, tackles the curse of dimensionality as depth does in ordinary networks. The global weight tensor of a deep model is too large to instantiate, but it *decomposes into a tree tensor network with shared local tensors at fixed depth*. This lowers storage from $\mathcal{O}(d^n)$ to linear in depth without losing access to functions of the global tensor.

2.2 GLOBAL WEIGHT-BASED METRICS

Tensor similarity. The Frobenius inner product $\langle \mathbf{A} \mid \mathbf{B}\rangle = \sum A_{ki_1\cdots i_n}B_{ki_1\cdots i_n}$ compares two multilinear models through the weight tensors that carry their full functional information. Tensor similarity generalises this by inserting a symmetric positive-definite metric $\mathbf{M}$ and normalising,

$$\text{sim}_{\mathbf{M}}(\mathbf{A}, \mathbf{B}) = \frac{\langle \mathbf{A} \mid \mathbf{M} \mid \mathbf{B}\rangle}{\|\mathbf{A}\|_{\mathbf{M}}\|\mathbf{B}\|_{\mathbf{M}}} \in [-1, 1], \tag{2}$$

where $\mathbf{M}$ determines which directions in weight space are considered relevant. The induced similarity is Lipschitz-continuous in the weights (subsection B.6), so weight perturbations cause only proportional functional perturbations.

Tractable inner product. The tree decomposition lets us compute these inner products without materialising the global tensor. A *Gram-based recursion* [Dooms et al., 2025a] processes each layer's local tensors in turn, computing inner products through metric tensors that depend on the previous layer's result; tracing the final Gram matrix recovers $\langle \mathbf{A} \mid \mathbf{B}\rangle$. The global metric $\mathbf{M}$ must preserve the tree shape assumed by this recursion (Appendix B).

2.3 PROJECTING OUT WEIGHT SYMMETRIES

We seek a metric $\mathbf{M}$ whose induced similarity tracks model equivalence, where *two models are equivalent if and only if they implement functions related by a positive scalar* ($\mathcal{A} \cong \mathcal{B} \iff \exists\lambda > 0 : \mathcal{A} = \lambda\mathcal{B}$). Such an $\mathbf{M}$ must

be invariant under weight-space transformations between equivalent parametrisations.

Symmetrisation tracks equivalence by polarisation. A bilinear layer's output depends only on the symmetric part of its weight tensor, since the antisymmetric part vanishes against the symmetric input $\tilde{\mathbf{x}} \otimes \tilde{\mathbf{x}}$ (Appendix B). This generalises to any order through the *polarisation isomorphism* [Ramshaw, 1989]: writing $\mathcal{S}_n$ for the permutations of n indices, order-n symmetric tensors correspond one-to-one with homogeneous order-n polynomials. Two models are therefore equivalent if and only if their tensors share the same representative in the symmetric subspace, reached by the orthogonal projection (*symmetrisation*) $\mathbf{P}_n$. Restricting to metrics $\mathbf{M} = \mathbb{1} \otimes \mathbf{\Lambda}$ with $\mathbf{P}_n \mathbf{\Lambda} \mathbf{P}_n = \mathbf{\Lambda}$ and combining symmetrisation with Cauchy–Schwarz gives the *central guarantee*:

$$\mathrm{sim}_{\mathbf{M}}(\mathbf{A}, \mathbf{B}) = 1 \iff \exists\, \lambda > 0 : \mathcal{A} = \lambda \mathcal{B}. \qquad (3)$$

Gaussian metric. The metric can be sharpened further. Re-expressing the inner product of two model outputs as $\langle \mathcal{A}(\mathbf{x}) \mid \mathcal{B}(\mathbf{x}) \rangle = \langle \mathbf{A} \mid \otimes^{2n} \tilde{\mathbf{x}} \mid \mathbf{B} \rangle$ reveals a metric tensor $\otimes^{2n} \tilde{\mathbf{x}}$ with a larger symmetry than $\mathcal{S}_n \times \mathcal{S}_n$: it is invariant under the full group $\mathcal{S}_{2n} \supset \mathcal{S}_n \times \mathcal{S}_n$. Since any $\mathcal{S}_{2n}$-symmetric metric is in particular $\mathcal{S}_n \times \mathcal{S}_n$-symmetric, it still satisfies Equation 3. The symmetriser $\mathbf{P}_{2n}$ would be such a metric, but it is of the wrong order, carrying too many legs to act as a metric on the $2n$ input copies. Instead we use the Gaussian expectation of those copies, $\mathbf{\Lambda} = \mathbb{E}_{\mathbf{x} \sim \mathcal{N}(0, \mathbf{I})} \otimes^{2n} \mathbf{x}$, which is $\mathcal{S}_{2n}$-symmetric and has a closed form via Isserlis' theorem [Munthe-Kaas et al., 2025] (Appendix B). With this choice the *expected inner product of two models' activations under Gaussian input equals their weight-space inner product*, connecting tensor similarity directly to behavioural similarity while requiring no data. Such metrics can still be too high-order to instantiate efficiently for deep models, where we instead use layer-decomposed approximations such as layer-wise symmetrisation (Appendix B). Equivalently, on centred symmetric tensors the metric realises the Frobenius inner product as the *covariance* of the model outputs under Gaussian inputs (subsection B.4). The same construction extends to anisotropic input distributions, recovering data-dependent behavioural similarity as a special case (subsection B.5).

2.4 LOCALISATION ISOLATES COMPONENTS

Tensor similarity with the Gaussian metric is a closed-form, data-free measure of functional equivalence that can be localised, inheriting the guarantee of Equation 3. Fixing an output index k isolates the sub-computation that produces that single output, which we call a *mechanism*: it yields a *slice* $\mathbf{A}_k \in \otimes^n \mathbb{R}^{d+1}$ with induced metric $\mathbf{\Lambda}$, giving *tensor slice similarity*, which measures functional agreement in one output dimension.

3 RESULTS

Tensor similarity satisfies the desired criteria in theory; does this hold in practice? We highlight two capabilities: catching out-of-distribution changes other metrics miss (backdoor injection on SVHN) and tracking continuous reorganisation through training (grokking on modular addition). The supplementary material adds per-class localisation under catastrophic forgetting and scaling to a two-layer bilinear attention transformer on The Pile.

3.1 CATCHING OOD CHANGES

Behavioural metrics evaluate similarity on a fixed input distribution, so a mechanism active only out of distribution does not affect their score; tensor similarity is data-free and detects it. We inject a class-'9' backdoor into SVHN [Netzer et al., 2011] that leaves clean accuracy untouched and compare metrics across checkpoints spanning its insertion (Figure 1; full setup in Appendix C). Behavioural similarity on clean data and matrix cosine on the weights barely register the attack, and both discard any handle on which output is responsible; tensor similarity flags it from clean data alone, and its digit-'9' slice localises the change to the attacked class. This is the same per-class localisation we demonstrate for catastrophic forgetting in Appendix C.

3.2 TRACKING REORGANISATION THROUGH TRAINING

In modular addition, grokking is not a single phase change but a continuous reorganisation into particular Fourier frequencies. We train a single bilinear layer on modular addition following Nanda et al. [2023], where the Fourier algorithm is well established, save K checkpoints throughout training, and compute their pairwise tensor similarity as a $K \times K$ matrix (setup in Appendix D).

The accuracy curves in Figure 5 show standard grokking: training accuracy saturates immediately, then validation jumps from chance to perfect. The similarity matrix reveals that the reorganisation continues beyond this jump, with the Fourier frequencies stabilising only near the end of training, when the loss also reaches its floor. This makes the development legible without reverse engineering the whole model.

4 CONCLUSION

We introduced tensor similarity, a weight-based measure of functional equivalence for tensor-based models such as bilinear transformers. Measuring similarity from weights requires accounting for the symmetries that leave a model's outputs invariant; tensor similarity projects these out, which

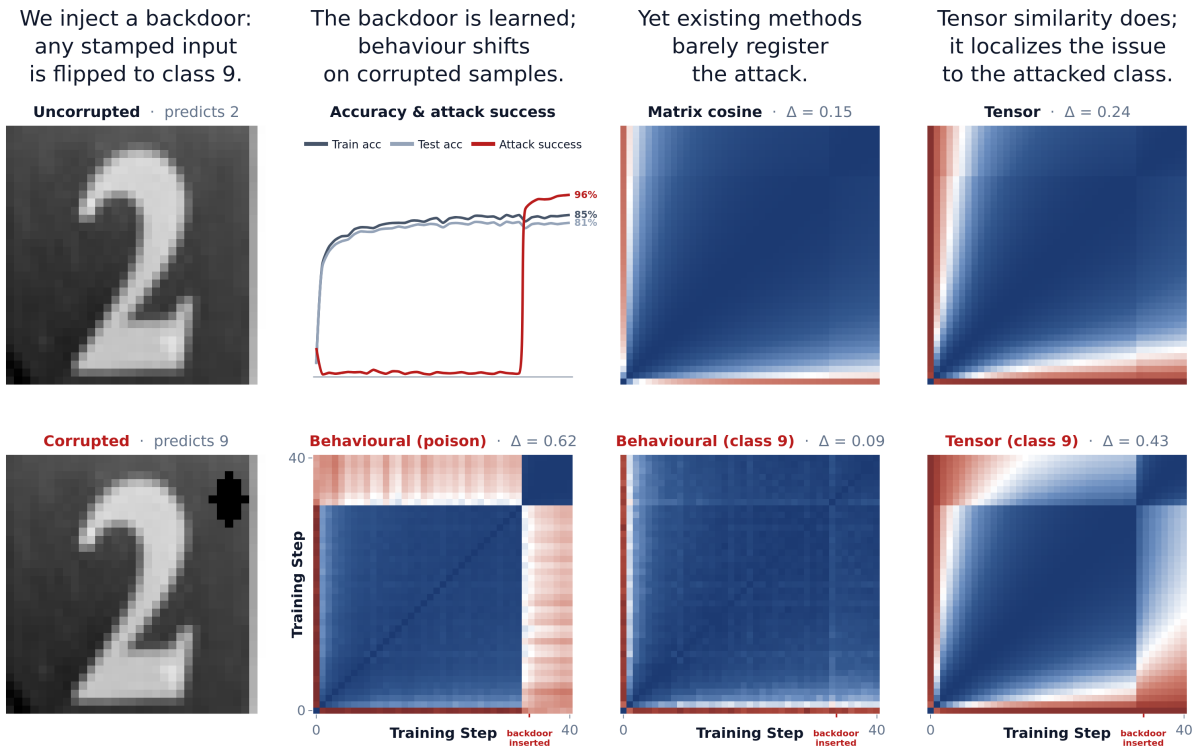

Figure 1: **Tensor similarity detects and localises a backdoor that other metrics miss.** We poison 10% of SVHN [Netzer et al., 2011] training images by stamping a small black diamond in the upper-right corner and relabelling them as digit '9' (leftmost column: a '2' read correctly when clean, as '9' when stamped). Training in continued stages, the model reaches 96% attack success while clean train and test accuracy stay at 85% and 81% (top of the second column), so the attack is invisible to performance metrics. Each heatmap is a checkpoint-by-checkpoint similarity matrix over training steps 0 to 40 (both axes); the diagonal compares a checkpoint with itself, and the labelled step marks where the backdoor is inserted. Colour runs from 0 (orthogonal, red) to 1 (colinear, blue). We summarise each matrix by Δ, the within-block minus across-block similarity across the insertion (larger Δ is a sharper split). Behavioural similarity on the clean distribution ($\Delta = 0.09$) and matrix cosine on the weights ($\Delta = 0.15$) barely react; behavioural similarity recovers the signal only on poisoned inputs ($\Delta = 0.62$), which requires knowing the trigger. Tensor similarity flags the change from clean data alone ($\Delta = 0.24$), and the digit-'9' slice, obtained by fixing that output index of the weight tensor, sharpens it to $\Delta = 0.43$ and localises it to the attacked class.

combined with the polarisation isomorphism makes the metric track functional equivalence. The link to behavioural similarity under Gaussian inputs yields a Gaussian metric with a stricter equivalence relation. For models of realistic scale, we propose and motivate a recursive algorithm that exploits the tensor-network structure of the weights. Across four tasks, catastrophic forgetting, backdoor insertion, grokking, and n-gram formation in language modelling, tensor similarity tracks functional changes more cleanly than existing metrics, and slicing localises them to specific outputs.

The main limitation is the reliance on multilinear models, which excludes architectures whose non-polynomial nonlinearities block algebraic analysis. Even so, these models earn their place as transparent test beds for interpretability, with mounting evidence that they are competitive with standard architectures [Pearce et al., 2025, Shazeer, 2020]. A secondary limitation is scope: we demonstrate detection of functional changes more thoroughly than attribution of the components responsible. A natural next step is local application, computing tensor similarity between paths through a deep multilinear transformer to extend Elhage et al. [2021] beyond the attention-only case that produced induction heads. More broadly, because these models are tensor networks whose functions are recoverable from weights alone [Levine et al., 2019], they support tensor-based architectures as a foundation for trustworthy machine learning [Cohen and Shashua, 2016, Riggs, 2026].

Acknowledgements

This research was conducted as part of the MARS 4.0 fel-

lowship, organised by the Cambridge AI Safety Hub (now Meridian). It also received funding from the Research Foundation Flanders (FWO) grant 11A8R26N (WG). We are grateful to Meridian for making our collaboration possible, and to our research manager, Mikhail Mironov, for keeping the project on track. We also thank Richard Mohn and Viktor Rehnberg for technical discussions and encouragement.

References

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, et al. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6541–6549, 2017. URL https://arxiv.org/abs/1704.05796.

Sander Beckers, Frederick Eberhardt, and Joseph Y. Halpern. Approximate causal abstractions. In *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference (UAI)*, volume 115 of *Proceedings of Machine Learning Research*, pages 606–615, 2020. URL https://arxiv.org/abs/1906.11583.

Nora Belrose, Quintin Pope, Lucia Quirke, Alex Mallen, and Xiaoli Fern. Neural networks learn statistics of increasing complexity, 2024. URL https://arxiv.org/abs/2402.04362.

Weixin Chen, Simon Yu, Huajie Shao, Lui Sha, and Han Zhao. Neural Probabilistic Circuits: Enabling Compositional and Interpretable Predictions through Logical Reasoning, 2025. URL http://arxiv.org/abs/2501.07021.

Nadav Cohen and Amnon Shashua. Convolutional Rectifier Networks as Generalized Tensor Decompositions, 2016. URL https://arxiv.org/abs/1603.00162v2.

Nadav Cohen, Or Sharir, and Amnon Shashua. On the Expressive Power of Deep Learning: A Tensor Analysis, 2015. URL https://arxiv.org/abs/1509.05009v3.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2304.14997.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. URL https://arxiv.org/abs/2309.08600.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghassen Jerfel, Alan Karthikesalingam, Mario Lucic, Yian Ma, Cory McLean, Diana Mincu, Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Natarajan, Christopher Nielson, Thomas F. Osborne, Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Taedong Yun, Xiaohua Zhai, and D. Sculley. Underspecification presents challenges for credibility in modern machine learning, 2020. URL https://arxiv.org/abs/2011.03395.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks, 2017. URL https://arxiv.org/abs/1612.08083.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Thomas Dooms and Ward Gauderis. Finding Manifolds With Bilinear Autoencoders, 2025. URL http://arxiv.org/abs/2510.16820.

Thomas Dooms, Ward Gauderis, Geraint A. Wiggins, and José Oramas. Compositionality unlocks deep interpretable models, 2025a. URL https://arxiv.org/abs/2504.02667.

Thomas Dooms, Ward Gauderis, Geraint A. Wiggins, and Jose Oramas. Compositionality Unlocks Deep Interpretable Models, April 2025b. URL http://arxiv.org/abs/2504.02667. arXiv:2504.02667 [cs].

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. https://transformer-circuits.pub/2021/framework/index.html.

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, 2020. URL https://arxiv.org/abs/2101.00027.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021. URL https://arxiv.org/abs/2106.02997.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah D. Goodman. Finding alignments between interpretable causal variables and distributed neural representations. In *Proceedings of the Third Conference on Causal Learning and Reasoning (CLeaR)*, volume 236 of *Proceedings of Machine Learning Research*, 2024. URL https://arxiv.org/abs/2303.02536.

Johnnie Gray. quimb: a python library for quantum information and many-body calculations. *Journal of Open Source Software*, 3(29):819, 2018. doi: 10.21105/joss.00819.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms, 2024. URL https://arxiv.org/abs/2403.17806.

Jesse Hoogland, George Wang, Matthew Farrugia-Roberts, Liam Carroll, Susan Wei, and Daniel Murfet. Loss landscape degeneracy and stagewise development in transformers, 2025. URL https://arxiv.org/abs/2402.02364.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viégas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2668–2677. PMLR, 2018. URL https://arxiv.org/abs/1711.11279.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 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, 2019. URL https://arxiv.org/abs/1905.00414.

Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. Representational similarity analysis — connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2:4, 2008. doi: 10.3389/neuro.06.004.2008. URL https://doi.org/10.3389/neuro.06.004.2008.

Yoav Levine, Or Sharir, Nadav Cohen, and Amnon Shashua. Quantum Entanglement in Deep Learning Architectures. *Phys. Rev. Lett.*, 122(6):065301, 2019. doi: 10.1103/PhysRevLett.122.065301. URL https://link.aps.org/doi/10.1103/PhysRevLett.122.065301.

Lorenzo Loconte, Antonio Mari, Gennaro Gala, Robert Peharz, Cassio de Campos, Erik Quaeghebeur, Gennaro Vessio, and Antonio Vergari. What is the Relationship between Tensor Factorizations and Circuits (and How Can We Exploit it)?, 2025. URL http://arxiv.org/abs/2409.07953.

Samuel Marks, Johannes Treutlein, Trenton Bricken, Jack Lindsey, Jonathan Marcus, Siddharth Mishra-Sharma, Daniel Ziegler, Emmanuel Ameisen, Joshua Batson, Tim Belonax, Samuel R. Bowman, Shan Carter, Brian Chen, Hoagy Cunningham, Carson Denison, Florian Dietz, Satvik Golechha, Akbir Khan, Jan Kirchner, Jan Leike, Austin Meek, Kei Nishimura-Gasparian, Euan Ong, Christopher Olah, Adam Pearce, Fabien Roger, Jeanne Salle, Andy Shih, Meg Tong, Drake Thomas, Kelley Rivoire, Adam Jermyn, Monte MacDiarmid, Tom Henighan, and Evan Hubinger. Auditing language models for hidden objectives, 2025. URL https://arxiv.org/abs/2503.10965.

Hans Z. Munthe-Kaas, Olivier Verdier, and Gilles Vilmart. A short proof of Isserlis' theorem, 2025. URL http://arxiv.org/abs/2503.01588.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, 2023. URL https://arxiv.org/abs/2301.05217.

Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, 2023. URL https://arxiv.org/abs/2301.12780.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Timothy Nguyen. Understanding transformers via n-gram statistics, 2024. URL https://arxiv.org/abs/2407.12034.

Chris Olah. A toy model of mechanistic (un)faithfulness. *Transformer Circuits Thread*, 2025. URL https://tr

ansformer-circuits.pub/2025/faithfulness-toy-model/index.html.

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. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Michael T. Pearce, Thomas Dooms, Alice Rigg, José M. Oramas, and Lee Sharkey. Bilinear MLPs enable weight-based mechanistic interpretability. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.08417.

Lyle Ramshaw. Blossoms are polar forms. *Computer Aided Geometric Design*, 6(4):323–358, 1989. ISSN 0167-8396. doi: 10.1016/0167-8396(89)90032-0. URL https://www.sciencedirect.com/science/article/pii/0167839689900320.

Logan Riggs. Tensor-transformer variants are surprisingly performant. LessWrong, 2026. URL https://www.lesswrong.com/posts/hp9bvkiN3RzHgP9cq/tensor-transformer-variants-are-surprisingly-performant.

Lee Sharkey. A technical note on bilinear layers for interpretability, 2023. URL http://arxiv.org/abs/2305.03452.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, Stella Biderman, Adria Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, Eric J. Michaud, Stephen Casper, Max Tegmark, William Saunders, David Bau, Eric Todd, Atticus Geiger, Mor Geva, Jesse Hoogland, Daniel Murfet, and Tom McGrath. Open problems in mechanistic interpretability, 2025. URL https://arxiv.org/abs/2501.16496.

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Amir Shpilka and Amir Yehudayoff. Arithmetic Circuits: A Survey of Recent Results and Open Questions. *Foundations and Trends in Theoretical Computer Science*, 5: 207–388, 2010. doi: 10.1561/0400000039.

George Wang, Jesse Hoogland, Stan van Wingerden, Zach Furman, and Daniel Murfet. Differentiation and specialization of attention heads via the refined local learning coefficient, 2024. URL https://arxiv.org/abs/2410.02984.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling. *arXiv preprint arXiv:2302.03169*, 2023.

Tensor Similarity (Supplementary Material)

ML Nissen Gonzalez[1,2] **Melwina Albuquerque**[1] **Laurence Wroe**[1] **Jacob Meyer Cohen**[1,3]
Ward Gauderis[4] **Logan Riggs Smith**[5] **Thomas Dooms**[5]

[1]MARS [2]Heidelberg University [3]Stanford University [4]Vrije Universiteit Brussel [5]Independent

A RELATED WORK

Bilinear and multilinear architectures. The work most directly related to ours uses bilinear layers as an intrinsically interpretable deep learning architecture. Dooms et al. [2025b] introduce chi-nets and show that their compositional multilinear structure enables a formal, weight-based interpretability analysis via eigendecomposition of the learned weight tensors. The same approach is extended to bilinear autoencoders for identifying learned feature manifolds [Dooms and Gauderis, 2025], and to language modelling, where bilinear MLPs are shown to unlock structured feature representations [Pearce et al., 2025, Sharkey, 2023]. These works establish the bilinear transformer as a viable architecture and motivate weight-space analysis; our contribution is the similarity measure that makes cross-model and cross-checkpoint comparisons in that weight space principled.

Tensor networks as neural network models. The use of tensor networks to analyse neural networks rigorously has a longer history. Cohen et al. [2015] provide the first rigorous tensor-analytic treatment of depth versus width in deep learning. A companion paper [Cohen and Shashua, 2016] defines the Convolutional Arithmetic Network, a purely multilinear variant of the CNN in which the map from the linear to the non-linear model is achieved by the generalised tensor product, the same operation that relates bilinear attention to softmax attention. Levine et al. [2019] connect quantum entanglement capacity to the expressiveness of depth in these architectures, providing further formal grounding for studying deep multilinear models.

Learning in weight space. A complementary line of work operates directly on network weights while respecting the symmetries of that space. Navon et al. [2023] construct architectures that are equivariant to the permutation symmetries of deep weight spaces, learning the invariances rather than quotienting them out analytically. Tensor similarity addresses the same obstacle, that raw parameters are only defined up to symmetry, but resolves it in closed form by projecting onto the symmetric subspace instead of learning an equivariant map.

Arithmetic and probabilistic circuits. Arithmetic and probabilistic circuits form a related algebraically structured family of models whose tractability properties have been studied for decades [Chen et al., 2025, Shpilka and Yehudayoff, 2010]. Their connection to tensor networks is well established [Loconte et al., 2025], and holds most directly when circuits are restricted to sum and product operations. The interpretability arguments in that literature, centred on exploiting closed-form structure, are closely related in spirit to our approach.

B THEORY APPENDIX

B.1 MULTILINEAR MODELS AND THEIR SYMMETRIES.

Multilinear models. Multilinear models take the following form: $\mathcal{A}(\mathbf{x}) = \mathbf{A} \cdot \otimes^n \tilde{\mathbf{x}}$ where $\mathbf{x} \in \mathbb{R}^d$, $\tilde{\mathbf{x}} = (1, \mathbf{x})$, $\mathbf{A} \in \mathbb{R}^K \otimes^n \mathbb{R}^{d+1}$, and $\cdot$ denotes the tensor contraction along $\otimes^n \mathbb{R}^{d+1}$. In component form: $\mathcal{A}(\mathbf{x})_k = \sum_{\mathbf{i} \in [d]_0^n} A_{k i_1 \dots i_n} x_{i_1} \cdots x_{i_n}$, which is a polynomial of order n in the components of $\mathbf{x}$. $\mathcal{A}$ thus models $K \in \mathbb{N}$ polynomials.

Deep learning with multilinear models. The order of the polynomial governs the expressiveness of multilinear models. However, the size of the tensor $\mathbf{A}$ grows exponentially with order as $\mathcal{O}(K(d+1)^n)$: this is the curse of dimensionality.

To tackle this with multilinear models, we layer them. Consider a deep multilinear model with $l \in \mathbb{N}$ layers:

$$\mathcal{A}(\mathbf{x}) = \mathcal{A}^{(l)}(\mathcal{A}^{(l-1)} \cdots (\mathcal{A}^{(1)}(\mathbf{x})) \cdots) = \mathbf{A}^{(l)} \cdot \otimes^{n_l} (\mathbf{A}^{(l-1)} \cdots \otimes^{n_2} (\mathbf{A}^{(1)} \cdot \otimes^{n_1} \tilde{\mathbf{x}})),$$

where $n_i \in \mathbb{N}$ denotes the input order of the local tensor $\mathbf{A}^{(i)} \in \mathbb{R}^{H_i} \otimes^{n_i} \mathbb{R}^{H_{i-1}}$ for $i \in [l]$ and $\{H_i\}$ the hidden dimensions with $H_0 = d+1$ and $H_l = K$. While the global tensor has order $n = 1 + \prod_{i=1}^{l} n_i$, only local tensors of order n_i need to be stored and this therefore constitutes a tensor decomposition of the global tensor $\mathbf{A}$.

Tensor networks are decompositions of tensors according to an underlying graph. The graph that underlies the decomposition of a deep multilinear model has tree shape, with the tensors within each layer being copies of each other. Thus, while the order of the modelled tensor is exponential in the number of layers, the number of coefficients that need to be stored in the local tensors grows only linearly with the number of layers.

Action of the permutation group. Consider a multilinear model given by the tensor $\mathbf{A} \in \mathbb{R}^K \otimes^n \mathbb{R}^{d+1}$ and the symmetric group on n elements, $\mathcal{S}_n$. Its action on $\mathbb{R}^K \otimes^n \mathbb{R}^{d+1}$ permutes the copies $\otimes^n \mathbb{R}^{d+1}$ as $A_{ki_1 \ldots i_n} \to A_{ki_{\sigma(1)} \ldots i_{\sigma(n)}}$. Tensors that are invariant under this action are said to be symmetric. The space of these symmetric tensors, $\mathbb{R}^K \otimes \mathrm{sym}^n \mathbb{R}^{d+1}$, is a proper vector subspace of the vector space of generic tensors. The projection down to the symmetric subspace, called symmetrisation and denoted by $\mathbf{P}_n : \otimes^n \mathbb{R}^{d+1} \to \mathrm{sym}^n \mathbb{R}^{d+1}$, is a linear orthogonal projection defined as the mean of all the actions of the symmetric group:

$$\mathbf{A}^{\mathrm{sym}} := (\mathbb{1} \otimes \mathbf{P}_n) \cdot \mathbf{A}, \quad A^{\mathrm{sym}}_{ki_1 \cdots i_n} \;=\; \frac{1}{n!} \sum_{\sigma \in \mathcal{S}_n} A_{ki_{\sigma(1)} \cdots i_{\sigma(n)}}. \tag{4}$$

The orthogonal projector $\mathbf{P}_n$ satisfies $\mathbf{P}_n^2 = \mathbf{P}_n$ and $\langle \mathbf{P}_n \mathbf{A} \mid \mathbf{B} \rangle = \langle \mathbf{A} \mid \mathbf{P}_n \mathbf{B} \rangle$.

Polarisation. This symmetry leaves the function $\mathcal{A}$ invariant, and so permuting the input copies leaves the modelled polynomial invariant. The precise statement of this is the polarisation isomorphism, usually stated for homogeneous polynomials. It states that there is a vector space isomorphism between the symmetric tensors over $\mathbb{R}^d$ of order n, $\mathrm{sym}^n \mathbb{R}^d$, and the space of homogeneous polynomials of order n in $\mathbf{x} \in \mathbb{R}^d$. In our case, because we consider general (not merely homogeneous) polynomials, we lift the input to $\tilde{\mathbf{x}} = (1, \mathbf{x})$ and work with degree-n homogeneous polynomials over $\mathbb{R}^{d+1}$; by polarisation these are in bijection with $\mathrm{sym}^n \mathbb{R}^{d+1}$. This applies for each slice $\mathbf{A}_k \in \mathrm{sym}^n \mathbb{R}^{d+1}$ of the multilinear model considered. The output space $\mathbb{R}^K$ has a privileged basis such that slicing along it is principled. For a single-layer multilinear model, therefore, symmetrising the input indices of the tensor $\mathbf{A}$ puts it into a one-to-one correspondence with the vector of polynomials modelled, accounting for all weight-space symmetries.

Example: bilinear layer. A *bilinear layer* is a single-layer multilinear model of order $n = 2$, so its weight tensor $\mathbf{A} \in \mathbb{R}^K \otimes^2 \mathbb{R}^{d+1}$ encodes K quadratic polynomials. Given matrices $\mathbf{D} \in \mathbb{R}^{K \times r}$ and $\mathbf{L}, \mathbf{R} \in \mathbb{R}^{r \times (d+1)}$ with hidden dimension $r \in \mathbb{N}$, and the order-3 delta tensor $\boldsymbol{\delta} \in \mathbb{R}^r \otimes \mathbb{R}^r \otimes \mathbb{R}^r$ with components $\delta_{h_1 h_2 h_3} = 1$ iff $h_1 = h_2 = h_3$ and zero otherwise, the weight tensor is

$$A_{ki_1 i_2} = \sum_{h_1, h_2, h_3 = 1}^{r} D_{kh_1}\, \delta_{h_1 h_2 h_3}\, L_{h_2 i_1}\, R_{h_3 i_2} = \sum_{h=1}^{r} D_{kh}\, L_{hi_1}\, R_{hi_2}. \tag{5}$$

This is a rank-r CP decomposition, equivalently written $\mathbf{A} = \sum_{h=1}^{r} \mathbf{d}_h \otimes \mathbf{l}_h \otimes \mathbf{r}_h$, where $\mathbf{d}_h, \mathbf{l}_h, \mathbf{r}_h$ are the h-th columns of $\mathbf{D}$ and rows of $\mathbf{L}, \mathbf{R}$ respectively. The forward pass reduces to

$$\mathcal{A}(\tilde{\mathbf{x}}) = \mathbf{D}\big[(\mathbf{L}\tilde{\mathbf{x}}) \odot (\mathbf{R}\tilde{\mathbf{x}})\big], \tag{6}$$

where $\odot$ denotes the Hadamard (elementwise) product, computable with matrix operations alone.

The output depends only on the symmetric part of $\mathbf{A}$. Decompose $\mathbf{A} = \mathbf{A}^{\mathrm{sym}} + \mathbf{A}^{\mathrm{asym}}$, where $A^{\perp}_{ki_1 i_2} = \frac{1}{2}(A_{ki_1 i_2} - A_{ki_2 i_1})$ is the antisymmetric complement. Since $\tilde{\mathbf{x}} \otimes \tilde{\mathbf{x}}$ is symmetric under $i_1 \leftrightarrow i_2$, the contraction with $\mathbf{A}^{\mathrm{asym}}$ vanishes for all $\tilde{\mathbf{x}}$:

$$\sum_{i_1, i_2} A^{\mathrm{asym}}_{ki_1 i_2}\, \tilde{x}_{i_1} \tilde{x}_{i_2} \;=\; \tfrac{1}{2} \sum_{i_1, i_2} (A_{ki_1 i_2} - A_{ki_2 i_1}) \tilde{x}_{i_1} \tilde{x}_{i_2} \;=\; 0.$$

Therefore $\mathcal{A}(\tilde{\mathbf{x}}) = \mathbf{A}^{\mathrm{sym}} \cdot_{\mathrm{in}} (\tilde{\mathbf{x}} \otimes \tilde{\mathbf{x}})$, and in terms of the CP factors,

$$A^{\mathrm{sym}}_{ki_1i_2} \;=\; \tfrac{1}{2} \sum_{h=1}^{r} D_{kh} \big(L_{hi_1} R_{hi_2} + R_{hi_1} L_{hi_2} \big). \tag{7}$$

Two bilinear layers with different $(\mathbf{L}, \mathbf{R})$ but equal $\mathbf{A}^{\mathrm{sym}}$ implement the same polynomial; weight-based comparison is therefore only principled after symmetrisation, as established by the polarisation isomorphism above.

The symmetry of deep multilinear models Layering multilinear models introduces an additional subtlety. It is too expensive to instantiate the full tensor $\mathbf{A}$ that decomposes according to a tree into the local tensors $\mathbf{A}^{(i)}$ for $i \in [l]$. However, full permutations of all input indices of the global tensor would require instantiating the full tensor. Moreover, these full permutations would destroy the tree decomposition structure. Because leaving the decomposition space is computationally intractable, we do not consider the action of the full symmetric group. Restricting to these tree decompositions of the global tensor, we identify the symmetry group that is compatible with the tree decomposition of $\mathbf{A}$ to be the wreath product of local permutations: $G_{\mathrm{tree}} := \mathcal{S}_{n_l} \wr \mathcal{S}_{n_{l-1}} \wr \cdots \wr \mathcal{S}_{n_1} \subsetneq \mathcal{S}_n$. Symmetrising with respect to this group symmetrises each layer in turn, resulting in the locally symmetrised $\mathbf{A}^{(l),\mathrm{sym}} \cdot \otimes^{n_l} (\mathbf{A}^{(l-1),\mathrm{sym}} \cdots \otimes^{n_2} (\mathbf{A}^{(1),\mathrm{sym}}))$. Projecting $\mathbf{A}$ to the symmetric subspace of G_{tree} by symmetrising each layer locally removes all degrees of freedom related to input permutation symmetry, while remaining within the class of multilinear models sharing the same tree decomposition. This enables principled weight-based comparison between models of the same decomposition class.

Another symmetry of layered multilinear models concerns basis transformations of the hidden dimensions along which the different layers are connected. This is discussed next but concerns only functions of local tensors (not of the global tensor).

Gauge symmetry. For tensor decompositions, just like matrix decompositions, the exact values of the local tensors are not unique. Inserting $\mathbf{U}\mathbf{U}^{-1}$ in any contracted dimension leaves the global tensor invariant. Functions of the global tensor, including the modelled polynomial for multilinear models or the inner product between two decomposed tensors, are also invariant under gauge transformations. Gauge fixing is the procedure to remove these degrees of freedom. For our models, privileged output and input bases, as well as basis-dependent computations such as the elementwise product of tensors, fix a gauge. Combining this with a residual stream that propagates the privileged input basis through the layers of a deep multilinear model fixes a gauge. This is only relevant if the tensors of individual layers are analysed instead of computing functions from the global tensor such as tensor similarity.

Implications for tensor similarity. The central choice in defining tensor similarity is the metric tensor $\mathbf{M} \in \otimes^2(\otimes^n \mathbb{R}^{d+1})$. Three requirements constrain this choice: it should project out weight-space symmetries, relate deviation from unity with functional difference like behavioural similarity, and preserve the class of deep multilinear models, i.e. those whose global tensors admit a special tree decomposition.

When the metric projects out weight-space symmetries (for instance, when combined with layerwise symmetrisation), tensor similarity reduces to the inner product between two tensors that are in one-to-one correspondence with polynomials. The Cauchy–Schwarz inequality relates this inner product to the product of the tensor norms, precisely the quantities appearing in the denominator of tensor similarity. It further guarantees that tensor similarity equals one if and only if the tensors, and hence the polynomials, are related by a positive scalar.

B.2 GRAM RECURSION: DERIVATION, SYMMETRISATION COMPATIBILITY, AND ALGORITHMS

This appendix provides the additional details used in Section 2.2. Here we:

- derive the Gram recursion from the tree structure of layered networks;
- prove the commutativity result: $\mathbf{P}_{n_i}$ commutes with the Gram step, so layer-wise symmetrisation is compatible with the recursion;
- give the explicit four-matrix-product algorithm for bilinear layers ($n_i = 2$) and state its complexity.

Define the Gram matrix at layer i as the partial contraction

$$\mathbf{G}^{(i)} \;=\; \mathbf{A}^{(i)} \cdot_{\mathrm{in}} \big(\mathbf{G}^{(i-1)}\big)^{\otimes q_i} \cdot_{\mathrm{in}} \mathbf{B}^{(i)}, \qquad \mathbf{G}^{(0)} = \mathbf{I}. \tag{8}$$

The inner product is recovered as $\langle \mathbf{A} \mid \mathbf{B} \rangle = \operatorname{tr} \mathbf{G}^{(L)}$.

Derivation of the recursion. The global weight tensor $\mathbf{A}$ of a layered network is a tree tensor network whose leaves are the local tensors $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(L)}$. The inner product $\langle \mathbf{A} \mid \mathbf{B} \rangle$ contracts all indices of $\mathbf{A}$ against the corresponding indices of $\mathbf{B}$. Because the underlying tree is acyclic, contraction can proceed from the leaves upward: contracting the input legs of layer i uses only the local tensors $\mathbf{A}^{(i)}, \mathbf{B}^{(i)}$ and the result of the contraction at layer $i-1$, which is captured by $\mathbf{G}^{(i-1)}$. Initialising with $\mathbf{G}^{(0)} = \mathbb{1} \in \otimes^2 \mathbb{R}^{d+1}$ (the identity on the input space) and applying this observation recursively yields (8). The inner product is the trace of the final Gram matrix because the remaining open legs at layer L are the output legs, and tracing them gives $\langle \mathbf{A} \mid \mathbf{B} \rangle = \sum_k G^{(L)}_{kk} = \operatorname{tr} \mathbf{G}^{(L)}$. The next paragraph shows a commutativity result that reduces the computational cost of layerwise symmetrisation during the Gram recursion. The final paragraph of this subsection gives the algorithm for the computation of the Gram recursion explicitly for bilinear layer.

Commutativity. Let $\mathbf{G} \in \otimes^2 \mathbb{R}^H$ and $\mathbf{P}_n$ the symmetrisation projector on $\bigotimes^n \mathbb{R}^H$. Then $[\mathbf{P}_n, \otimes^n \mathbf{G}] = 0$.

Proof. It suffices to show that each permutation operator $\mathbf{U}_\sigma$, $\sigma \in \mathcal{S}_n$, commutes with $\otimes^n \mathbf{G}$, since $\mathbf{P}_n$ is a linear combination of these. On product vectors,

$$\mathbf{U}_\sigma \, \otimes^n \mathbf{G}(\mathbf{x}_1 \otimes \cdots \otimes \mathbf{x}_n) = \mathbf{G}\mathbf{x}_{\sigma^{-1}(1)} \otimes \cdots \otimes \mathbf{G}\mathbf{x}_{\sigma^{-1}(n)} = \otimes^n \mathbf{G} \, \mathbf{U}_\sigma(\mathbf{x}_1 \otimes \cdots \otimes \mathbf{x}_n).$$

Product vectors span $\bigotimes^n \mathbb{R}^H$, so the claim follows. □

By idempotency of $\mathbf{P}_{n_i}$ and the above commutativity result, $\mathbf{P}_{n_i} \cdot \otimes^{n_i} \mathbf{G}^{(i-1)} \cdot \mathbf{P}_{n_i} = \otimes^{n_i} \mathbf{G}^{(i-1)} \cdot \mathbf{P}_{n_i}$. It follows that the symmetrised Gram recursion requires symmetrising only one of the two layers per step:

$$\mathbf{G}^{(i),\mathrm{sym}} \;=\; \mathbf{A}^{(i)} \cdot_{\mathrm{in}} \left(\otimes^{n_i} \mathbf{G}^{(i-1),\mathrm{sym}} \right) \cdot_{\mathrm{in}} \mathbf{P}_{n_i} \mathbf{B}^{(i)}. \tag{9}$$

Bilinear algorithm. For bilinear layers ($n_i = 2$) parameterised by $\mathbf{L}^{(i)}_{\mathcal{A}}, \mathbf{R}^{(i)}_{\mathcal{A}} \in \mathbb{R}^{r_i \times H_{i-1}}$, $\mathbf{D}^{(i)}_{\mathcal{A}} \in \mathbb{R}^{H_i \times r_i}$ (and likewise for $\mathcal{B}$), the Gram step (8) reduces to matrix operations alone. Initialise $\mathbf{G}^{(0)} = \mathbb{1}_{d+1}$ and, for each $i \in [L]$, compute four matrix products,

$$\begin{aligned} (\mathbf{LL})^{(i)} &:= \mathbf{L}^{(i)}_{\mathcal{A}} \, \mathbf{G}^{(i-1)} \left(\mathbf{L}^{(i)}_{\mathcal{B}}\right)^T, & (\mathbf{RR})^{(i)} &:= \mathbf{R}^{(i)}_{\mathcal{A}} \, \mathbf{G}^{(i-1)} \left(\mathbf{R}^{(i)}_{\mathcal{B}}\right)^T, \\ (\mathbf{LR})^{(i)} &:= \mathbf{L}^{(i)}_{\mathcal{A}} \, \mathbf{G}^{(i-1)} \left(\mathbf{R}^{(i)}_{\mathcal{B}}\right)^T, & (\mathbf{RL})^{(i)} &:= \mathbf{R}^{(i)}_{\mathcal{A}} \, \mathbf{G}^{(i-1)} \left(\mathbf{L}^{(i)}_{\mathcal{B}}\right)^T, \end{aligned} \tag{10}$$

two elementwise products,

$$\mathbf{E}^{(i)}_{\parallel} := (\mathbf{LL})^{(i)} \odot (\mathbf{RR})^{(i)}, \qquad \mathbf{E}^{(i)}_{\times} := (\mathbf{LR})^{(i)} \odot (\mathbf{RL})^{(i)}, \tag{11}$$

and one final matrix multiplication,

$$\mathbf{G}^{(i)} = \mathbf{D}^{(i)}_{\mathcal{A}} \left(\mathbf{E}^{(i)}_{\parallel} + \mathbf{E}^{(i)}_{\times}\right) \left(\mathbf{D}^{(i)}_{\mathcal{B}}\right)^T. \tag{12}$$

The two terms $\mathbf{E}_{\parallel}$ and $\mathbf{E}_{\times}$ correspond to the two terms in the symmetrisation of the bilinear layer: $\mathbf{E}_{\parallel}$ pairs same-role legs (L–L and R–R) while $\mathbf{E}_{\times}$ pairs swapped-role legs (L–R and R–L). Our experiments are implemented in Python with the tensor contraction algorithms like the above implemented using `quimb` Gray [2018].

B.3 DERIVATION OF THE GAUSSIAN METRIC

This appendix derives the Gaussian metric tensor presented in Section 2.3. Here we:

- show that the expected output similarity equals $\langle \mathbf{A} \mid \mathbf{M}^{(p)} \mid \mathbf{B} \rangle$ where $\mathbf{M}^{(p)}$ has $\mathcal{S}_{2n}$ symmetry;
- compute $\mathbf{\Lambda}^{(p)}$ for a Gaussian input distribution via the Isserlis–Wick theorem;
- conclude that this metric projects onto a strictly smaller subspace than the plain $\mathcal{S}_n$ symmetrisation.

Taking the expectation over inputs of the activation inner product gives

$$\mathbb{E}_{\mathbf{x}}\big[\langle \mathcal{A}(\mathbf{x}) \mid \mathcal{B}(\mathbf{x}) \rangle\big] \;=\; \langle \mathbf{A} \mid \mathbf{M}^{(p)} \mid \mathbf{B} \rangle, \tag{13}$$

where $\mathbf{M}^{(p)} = \mathbb{1} \otimes \mathbf{\Lambda}^{(p)}$ and

$$\mathbf{\Lambda}^{(p)}_{i_1 \cdots i_n j_1 \cdots j_n} \;=\; \mathbb{E}_{\mathbf{x}}\big[x_{i_1} \cdots x_{i_n} x_{j_1} \cdots x_{j_n}\big]. \tag{14}$$

The tensor $\mathbf{\Lambda}^{(p)}$ has order $2n$, pairing all $2n$ input legs of $\mathbf{A}$ and $\mathbf{B}$, including self-contractions of two legs belonging to the same tensor. This makes $\mathbf{\Lambda}^{(p)}$ manifestly invariant under the action of $\mathcal{S}_{2n}$.

Gaussian case via Isserlis' theorem. For $\mathbf{x} \sim \mathcal{N}(0, \mathbf{I}_d)$, Isserlis' theorem [Munthe-Kaas et al., 2025] expresses the $2n$-th moment as a sum over all perfect matchings of the $2n$ indices $i_1, \ldots, i_n, j_1, \ldots, j_n$. For unit covariance, each matched pair (a, b) contributes $\delta_{i_a i_b}$.

Classify pairings by $m \in \mathbb{N}$, the number of internal $\mathbf{A}$-pairs (pairs drawn entirely from $\{i_1, \ldots, i_n\}$). A pairing with m internal $\mathbf{A}$-pairs must also have m internal $\mathbf{B}$-pairs (because the total number of $\mathbf{A}$-indices must equal the number of $\mathbf{B}$-indices), leaving $n - 2m$ cross-pairs.

Define the partial trace operator τ such that $\tau^m \mathbf{T}$ contracts m pairs of indices as

$$(\tau\, \mathbf{T})_{i_3 \cdots i_n} \;=\; \sum_{a=1}^{d} T_{a\, a\, i_3 \cdots i_n}. \tag{15}$$

All pairings of type m contribute the same traced inner product $\langle \tau^m \mathbf{A} |\, \tau^m \mathbf{B} \rangle$.

Counting the pairings of type m: on the $\mathbf{A}$-side, choose and pair $2m$ of the n indices in $\binom{n}{2m}(2m-1)!!$ ways; the same count applies to $\mathbf{B}$. The remaining $n - 2m$ legs on each side are then cross-matched in $(n-2m)!$ ways. Setting

$$c_{n,m} \;=\; \binom{n}{2m}^2 (2m-1)!!^{\,2}\, (n-2m)!, \tag{16}$$

and summing over m gives the main result:

$$\mathbb{E}_{\mathbf{x} \sim \mathcal{N}(0, \mathbf{I}_d)} \big[\langle \mathcal{A}(\mathbf{x}) \mid \mathcal{B}(\mathbf{x}) \rangle \big] \;=\; \sum_{m=0}^{\lfloor n/2 \rfloor} c_{n,m}\, \langle \tau^m \mathbf{A},\, \tau^m \mathbf{B} \rangle. \tag{17}$$

Equation (17) identifies $\mathbf{\Lambda}^{(p)}$ as a weighted sum of partial-trace operators, confirming that $\mathbf{M}^{(p)}$ is positive definite and carries full $\mathcal{S}_{2n}$ symmetry. Because the $\mathcal{S}_{2n}$-invariant subspace of $\bigotimes^n \mathbb{R}^{d+1}$ is strictly smaller than the $\mathcal{S}_n$-invariant one, this Gaussian metric is a strictly finer equivalence relation than plain $\mathcal{S}_n$-symmetrisation, while still satisfying the discriminative-power guarantee (3).

B.4 COVARIANCE INTERPRETATION OF THE GAUSSIAN METRIC

The Gaussian metric $\mathbf{M}$ of Equation 13 admits a statistical reading. Fix two symmetric tensors $\mathbf{A}, \mathbf{B}$ and write $\mathcal{A}(\mathbf{x}) = \mathbf{A} \cdot \otimes^n \tilde{\mathbf{x}}$ for the K polynomials they realise. Then

$$\langle \mathbf{A} \mid \mathbf{M} \mid \mathbf{B} \rangle \;=\; \mathbb{E}_{\mathbf{x} \sim \mathcal{N}(0, \mathbf{I}_d)} \big[\langle \mathcal{A}(\mathbf{x}) \mid \mathcal{B}(\mathbf{x}) \rangle \big], \tag{18}$$

as stated in paragraph 2.3. Decomposing each polynomial into a mean and a centred part, $\mathcal{A}(\mathbf{x}) = \boldsymbol{\mu}_{\mathbf{A}} + \mathcal{A}^{\mathrm{c}}(\mathbf{x})$ with $\boldsymbol{\mu}_{\mathbf{A}} := \mathbb{E}_{\mathbf{x}}[\mathcal{A}(\mathbf{x})]$, the right-hand side splits as $\langle \boldsymbol{\mu}_{\mathbf{A}} \mid \boldsymbol{\mu}_{\mathbf{B}} \rangle + \mathrm{Cov}_{\mathbf{x}}(\mathcal{A}, \mathcal{B})$. The mean collects the trace-type self-contractions of $\mathbf{A}$. On the subspace of tensors with vanishing partial traces it disappears, leaving

$$\langle \mathbf{A} \mid \mathbf{M} \mid \mathbf{B} \rangle \;=\; \mathrm{Cov}_{\mathbf{x} \sim \mathcal{N}(0, \mathbf{I}_d)} \big(\mathcal{A}, \mathcal{B} \big). \tag{19}$$

Tensor similarity is therefore a function-space Pearson correlation that measures the cosine of the angle between two model functions in $L^2(\mathcal{N}(0, \mathbf{I}_d))$ after their constant components have been projected out.

Order-2 sanity check. For bilinear layers ($n = 2$), write each output slice of $\mathbf{A}$ in block form with respect to the lift $\tilde{\mathbf{x}} = (1, \mathbf{x})$, with quadratic block $\mathbf{Q}_{\mathbf{A}}$, linear block $b_{\mathbf{A}}$ and constant block $c_{\mathbf{A}}$. Expanding the Gaussian expectation via Isserlis' theorem yields

$$\mathbb{E}_{\mathbf{x}} \big[\langle \mathcal{A}(\mathbf{x}) \mid \mathcal{B}(\mathbf{x}) \rangle \big] \;=\; 2\, \langle \mathbf{Q}_{\mathbf{A}} \mid \mathbf{Q}_{\mathbf{B}} \rangle + 4\, \langle b_{\mathbf{A}} \mid b_{\mathbf{B}} \rangle + \langle \boldsymbol{\mu}_{\mathbf{A}} \mid \boldsymbol{\mu}_{\mathbf{B}} \rangle, \tag{20}$$

with $\boldsymbol{\mu}_{\mathbf{A}} = c_{\mathbf{A}} + \mathrm{tr}\, \mathbf{Q}_{\mathbf{A}}$. Subtracting the mean term gives $2\big(\langle \mathbf{A} \mid \mathbf{B} \rangle - \langle c_{\mathbf{A}} \mid c_{\mathbf{B}} \rangle \big) = \mathrm{Cov}_{\mathbf{x}}(\mathcal{A}, \mathcal{B})$, which is Equation 19 at $n = 2$.

B.5 FROM DATA-FREE TO DATA-DEPENDENT TENSOR SIMILARITY

The Gaussian metric is the isotropic case of a more general construction. Replace the standard Gaussian by a centred Gaussian with covariance $\mathbf{\Sigma} := \mathbb{E}[\mathbf{x}\mathbf{x}^\top]$, and write $\mathbf{M}^{(\mathbf{\Sigma})}$ for the metric it induces through Equation 13. On symmetric tensors at order $n = 2$,

$$\langle \mathbf{A} \mid \mathbf{M}^{(\mathbf{\Sigma})} \mid \mathbf{B} \rangle \;=\; 2\,\mathrm{tr}(\mathbf{A}\mathbf{\Sigma}\mathbf{B}\mathbf{\Sigma}) + \mathrm{tr}(\mathbf{A}\mathbf{\Sigma})\,\mathrm{tr}(\mathbf{B}\mathbf{\Sigma}), \tag{21}$$

which follows from the same Isserlis expansion as Equation 17 with δ_{ij} replaced by Σ_{ij}. Defining the whitened tensors $\bar{\mathbf{A}} := \mathbf{\Sigma}^{1/2}\mathbf{A}\mathbf{\Sigma}^{1/2}$ (and analogously at general order n via $\bar{\mathbf{A}} := \mathbf{A} \cdot_{\mathrm{in}} \otimes^n \mathbf{\Sigma}^{1/2}$, acting on the data coordinates and fixing the constant one), the substitution $\mathbf{x} = \mathbf{\Sigma}^{1/2}\mathbf{z}$ with $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_d)$ gives

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{N}(0,\mathbf{\Sigma})}\big[\langle \mathcal{A}(\mathbf{x}) \mid \mathcal{B}(\mathbf{x})\rangle\big] \;=\; \langle \mathbf{A} \mid \mathbf{M}^{(\mathbf{\Sigma})} \mid \mathbf{B}\rangle \;=\; \langle \bar{\mathbf{A}} \mid \mathbf{M} \mid \bar{\mathbf{B}}\rangle \;=\; \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I}_d)}\big[\langle \bar{\mathcal{A}}(\mathbf{z}) \mid \bar{\mathcal{B}}(\mathbf{z})\rangle\big], \tag{22}$$

so the data-dependent metric is the Gaussian metric on whitened weights. In the population limit, sampled behavioural similarity converges to $\langle \cdot \mid \mathbf{M}^{(\mathbf{\Sigma})} \mid \cdot \rangle$, and whitening the model by the empirical input covariance maps it back onto the data-free metric. The two metrics differ only in the choice of input prior, the Gaussian metric being the isotropic case $\mathbf{\Sigma} = \mathbf{I}_d$. Because tensor and behavioural similarity are on the same algebraic axis, one can simply substitute $\mathbf{\Sigma}$ to evaluate models on a particular distribution, without leaving the framework.

B.6 LIPSCHITZ STABILITY OF TENSOR SIMILARITY

Tensor similarity is a continuous function of model weights. The following bound makes this quantifiable in function space.

Setup. For a multilinear model of order n, define the lifted feature map $\phi_n : \mathbb{R}^d \to \mathbb{R}^{(d+1)^n}$ by $\phi_n(\mathbf{x}) := [1/\sqrt{n},\, \mathbf{x}]^{\otimes n}$, so that $\mathcal{A}(\mathbf{x}) = \mathbf{A} \cdot_{\mathrm{in}} \phi_n(\mathbf{x})$ for $\mathbf{A}$ symmetric in its n input indices. This is the lift $\otimes^n \tilde{\mathbf{x}}$ of subsection B.1 with the constant coordinate normalised to $1/\sqrt{n}$ rather than 1 (a rescaling that can be absorbed into $\mathbf{A}$). Assume the standard input normalisation $\|\mathbf{x}\|_2 \le 1$.

Theorem 1 (Lipschitz stability)**.** *For all* $\mathbf{x}$ *with* $\|\mathbf{x}\|_2 \le 1$ *and all symmetric weight tensors* $\mathbf{A}, \mathbf{B}$*,*

$$\|\mathcal{A}(\mathbf{x}) - \mathcal{B}(\mathbf{x})\|_2 \;\le\; \sqrt{e}\,\|\mathbf{A} - \mathbf{B}\|, \tag{23}$$

and consequently, for any input distribution p *supported on the unit ball,*

$$\|\mathcal{A} - \mathcal{B}\|_{L^2(p)} \;\le\; \sqrt{e}\,\|\mathbf{A} - \mathbf{B}\|. \tag{24}$$

The constant $\sqrt{e}$ *is independent of input dimension* d *and polynomial order* n*.*

Proof. By the Cauchy–Schwarz inequality for the tensor contraction, $\|\mathcal{A}(\mathbf{x}) - \mathcal{B}(\mathbf{x})\|_2^2 = \|(\mathbf{A} - \mathbf{B}) \cdot_{\mathrm{in}} \phi_n(\mathbf{x})\|_2^2 \le \|\mathbf{A} - \mathbf{B}\|^2\, \|\phi_n(\mathbf{x})\|_2^2$. The squared feature norm factorises as $\|\phi_n(\mathbf{x})\|_2^2 = (\|\mathbf{x}\|_2^2 + 1/n)^n \le (1 + 1/n)^n < e$, using $\|\mathbf{x}\|_2 \le 1$ in the first step and $(1 + 1/n)^n \le e$ in the second. Their combination yields $\|\mathcal{A}(\mathbf{x}) - \mathcal{B}(\mathbf{x})\|_2 \le \sqrt{e}\,\|\mathbf{A} - \mathbf{B}\|$. The $L^2(p)$ bound follows by taking expectations over p. □

Because $\mathrm{sim}_{\mathbf{M}}$ is a normalised inner product on the same weight space, Theorem 1 transfers this weight-space continuity to function space. Weight tensors that are close in Frobenius norm realise functions that are close in $L^2(p)$, so a small drop in tensor similarity reflects a proportionally small functional change.

C CATASTROPHIC FORGETTING AND BACKDOOR SETUP

This section gives the shared SVHN setup for the backdoor experiment (subsection 3.1) and presents the catastrophic forgetting experiment in full.

C.1 CATASTROPHIC FORGETTING LOCALISES MECHANISM CHANGES TO SPECIFIC OUTPUTS

Per-class slices pinpoint which output is responsible for a ground-truth mechanism change, not just that one occurred. We demonstrate this with continued training on SVHN [Netzer et al., 2011], a harder version of MNIST [Deng, 2012], to induce catastrophic forgetting. Training begins with a *base* stage on the digit subset $\{0, \cdots, 4\}$, followed by stages which incrementally introduce one digit at a time, culminating in the *add 9* stage on the full set $\{0, \cdots, 9\}$. We further stress-test the metric's sensitivity to dataset changes by appending three stages: *control* (continued training on the full set, disentangling the effect of a learning rate restart from that of any dataset change), *remove 9* (training on $\{0, \cdots, 8\}$ to induce catastrophic forgetting), and *re-add 9* (returning to the full set), as shown in Figure 2.

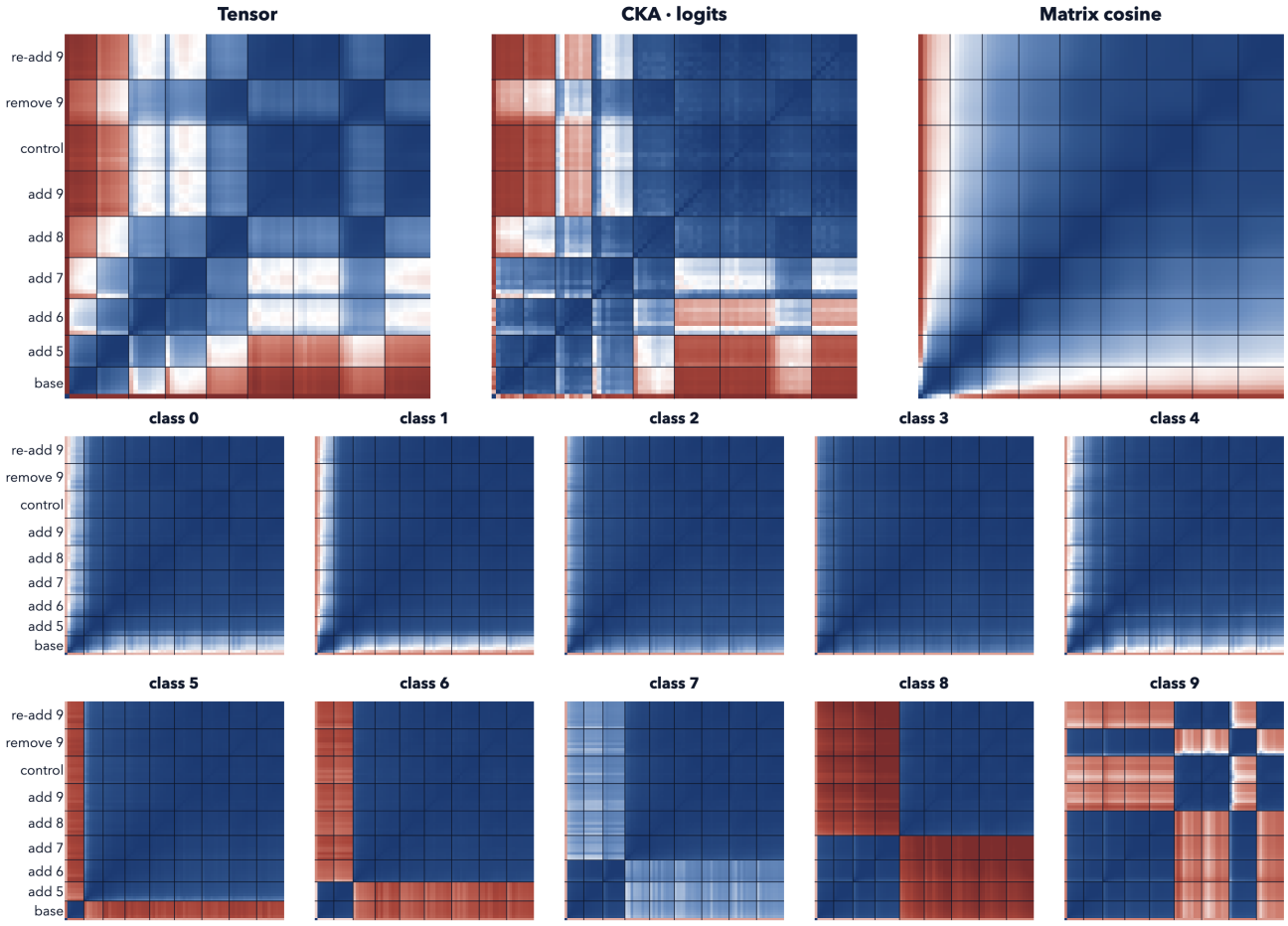

Figure 2: Four similarity measures across the training trajectory of an SVHN model. The top row compares whole-model summaries: global tensor similarity reveals a block structure separating stages with and without digit '9', while CKA on the logits and matrix similarity show much weaker, more ambiguous signal. The per-class tensor slices sharpen the picture: each exhibits exactly the block pattern predicted by when that digit was present in training, and the class '9' slice groups *base* through *add 8* with *remove 9*, all orthogonal to *add 9*, *control*, and *re-add 9*. Tensor similarity does not merely detect functional changes but can precisely localise them to the output.

C.2 EXPERIMENTAL SETUP

Dataset: The SVHN dataset comprises over $600,000$ labelled digits cropped from Google Street View images [Netzer et al., 2011]. It occupies a useful middle ground in difficulty: unlike MNIST [Deng, 2012], it is not solvable using linear models, yet it does not require the large architectures for ImageNet-scale tasks. We use a subset of approximately $73,000$ training and $26,000$ validation images, converted to greyscale.

Model Architecture: The model consists of a linear embedding layer, a single bilinear layer, and a linear unembedding layer. For ease of analysis, biases and normalisation layers are excluded from the model.

Backdoor injection: The backdoor experiment of subsection 3.1 uses the same dataset, architecture, and hyperparameters. During the injection stage, 10% of training samples carry a diamond-shaped trigger patch in the upper-right corner and are relabelled to target class '9'.

Model Dimensions and Training: Table 1 summarises the model dimensions and training hyperparameters.

Architecture	
d_model	128
d_hidden	256
n_layer	1
Training Parameters	
dropout	0.0
weight decay	0.5
batch size	248
learning rate	10^{-3}
optimizer	AdamW
schedule	Cosine Annealing
epochs	20 per stage

Table 1: Model architecture and training setup for the catastrophic forgetting and backdoor experiments.

C.3 ADDITIONAL FORGETTING RESULTS

Since weight tensors form a vector space, a difference $\mathbf{\Delta} = \mathbf{B} - \mathbf{C}$ is itself a valid tensor, so *tensor diff similarity* $\mathrm{sim}_{\mathbf{M}}(\mathbf{A}, \mathbf{\Delta})$ isolates similarity with respect to a specific functional change. We use it below.

D ADDITIONAL MODULAR ADDITION DETAILS

Dataset: The task is modular addition: given inputs (a, b), predict $(a + b) \bmod p$ where $p = 113$. Inputs are represented as two one-hot vectors of dimension $2p = 226$, concatenating the encodings of a and b. All $p^2 = 12{,}769$ input pairs are used, with a $60 : 40$ split between train and validation sets.

Model Architecture: The model consists of a single bilinear layer followed by a linear unembedding layer, with biases and normalisation layers.

Model Dimensions and Training: Table 2 summarises the model dimensions and training hyperparameters.

E LANGUAGE MODELLING WITH AN ATTENTION-ONLY TRANSFORMER

The body experiments used 1-layer bilinear models. Tensor similarity scales unchanged to deeper transformers and continues to find structure where existing metrics blur. We train a two-layer bilinear attention transformer on The Pile [Gao et al., 2020] and save 101 log-spaced checkpoints. Figure 6 reports five similarity measures across these checkpoints alongside the n-gram behaviour of the model, which prior work uses to track the increasingly complex structures models acquire over training, often discontinuously [Wang et al., 2024, Nguyen, 2024, Belrose et al., 2024]. Existing metrics separate only the earliest checkpoints from the rest, blurring everything beyond; tensor similarity exhibits clear block structure with sharp transitions between regimes.

More broadly, behavioural metrics see only the slice of input space they are evaluated on, and a mechanism that has reorganised globally need not register on that slice if its predictions there happen to coincide [D'Amour et al., 2020]. What generalises is what holds beyond the training distribution; measuring similarity on the training distribution conflates it with memorisation. Tensor similarity integrates over the full Gaussian input space and so registers reorganisations that would be invisible to any finite probe, which is precisely the regime where mechanistic claims need to hold.

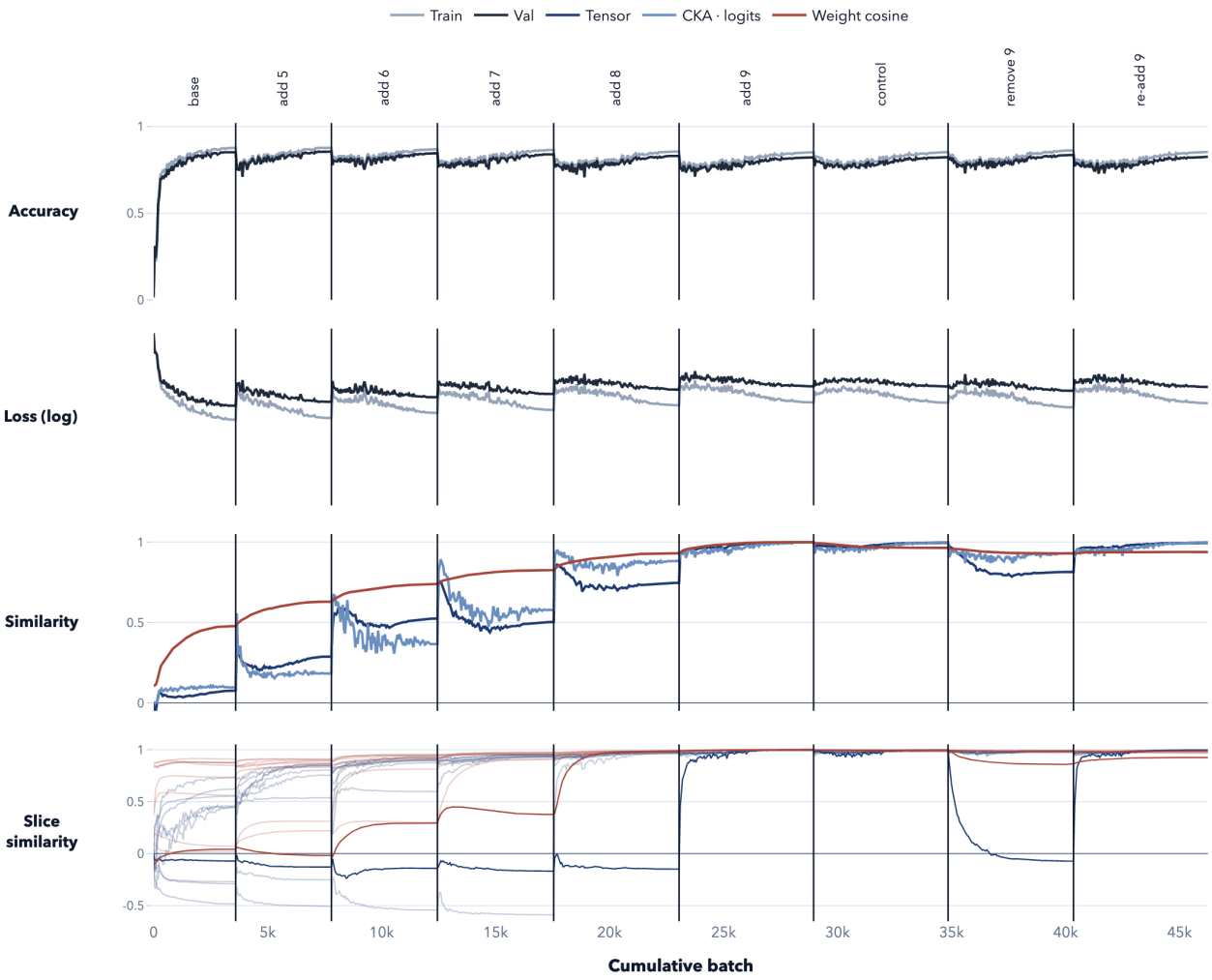

Figure 3: Model evolution across the progressive training setup on SVHN described in Appendix C. **Accuracy and loss** follow a consistent pattern across all stages, with validation accuracy reaching 83–85% at each stage. **Similarity** with respect to the model at the end of the *add 9* stage is shown for tensor similarity, CKA on logits, and weight cosine similarity. All three measures increase as digits are incrementally added, but tensor similarity exhibits the most pronounced drop at the *remove 9* stage and the clearest recovery at *re-add 9*, while CKA and weight cosine show considerably weaker signals. **Slice similarity** highlights digit '9' in a darker line and all other classes shown faint. The drop at *remove 9* is localised almost entirely to digit '9', confirming that catastrophic forgetting affects this class specifically. The effect is very clearly captured by tensor slice similarity, and is significantly weaker in weight cosine slice similarity.

E.1 TRAINING DETAILS

E.1.1 Data and Tokenisation

We train on the DSIR-filtered-Pile-50M subset from Xie et al. [2023]. Validation is created using a deterministic 1% held-out document split from the training set. We train a custom BPE tokeniser with vocabulary size $V = 4096$ on the same corpus. Documents are concatenated with EOS separators and chunked into fixed-length windows of length $n_{\text{ctx}} = 512$.

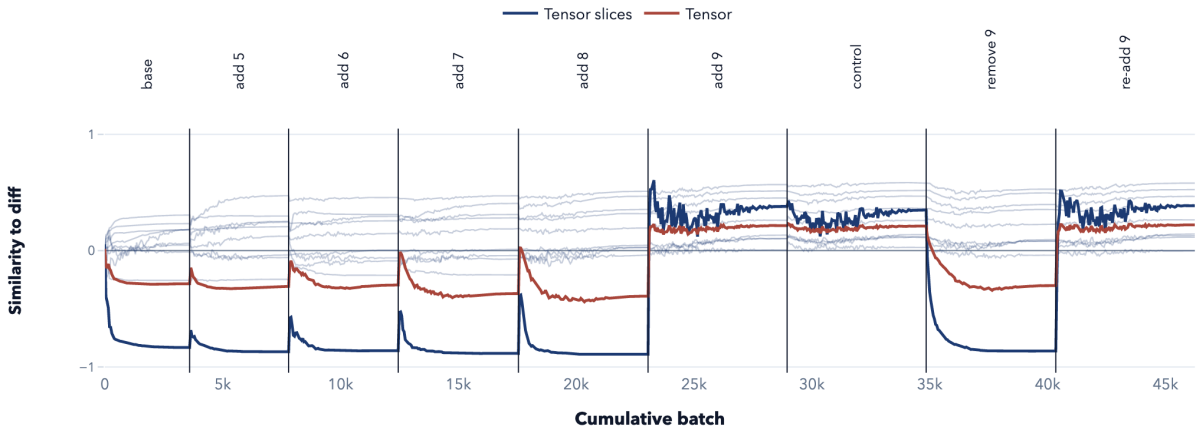

Figure 4: Tensor diff similarities across the progressive SVHN training setup (Appendix C). **Similarity to diff** is computed with respect to a diff tensor, computed as the difference between two checkpoints of an identically-architectured bilinear model trained on a different subset of SVHN data with a different random seed. The first checkpoint is taken after training on $\{0, \cdots, 8\}$ and the second after subsequent fine-tuning on the full set $\{0, \cdots, 9\}$, thereby isolating weight updates associated with learning digit '9'. Tensor similarity is shown in red and slice similarities are shown in blue, with digit '9' highlighted in a darker line. The resulting pattern closely mirrors the self-similarity result of Figure 3 with catastrophic forgetting at *remove 9* and subsequent recovery at *re-add 9* clearly visible, demonstrating that an externally constructed diff tensor can isolate the same functional component as self-comparison.

Architecture	
d_input	226
d_hidden	64
n_layer	1
Training Parameters	
dropout	0.0
weight decay	0.06
batch size	512
learning rate	10^{-3}
optimizer	AdamW
schedule	constant
steps	100,000

Table 2: Model architecture and training setup for the modular addition experiment.

E.1.2 Architecture

We use a decoder-only, attention-only language model with bilinear attention and no MLP blocks. The model has $L = 2$ layers, $H = 8$ attention heads, and model dimension $d_{\text{model}} = 256$. The embed and unembed are bias-free and not tied. Positional information is added using RoPE inside the attention layers.

To keep the model structure bilinear, we use only a scalar normalisation method applied before the unembedding. Specifically, we normalise the sequence by a scalar s computed from the first token's hidden state,

$$s = \sqrt{\frac{1}{d_{\text{model}}} \sum_{j=1}^{d_{\text{model}}} h_{0,j}^2 + \epsilon}.$$

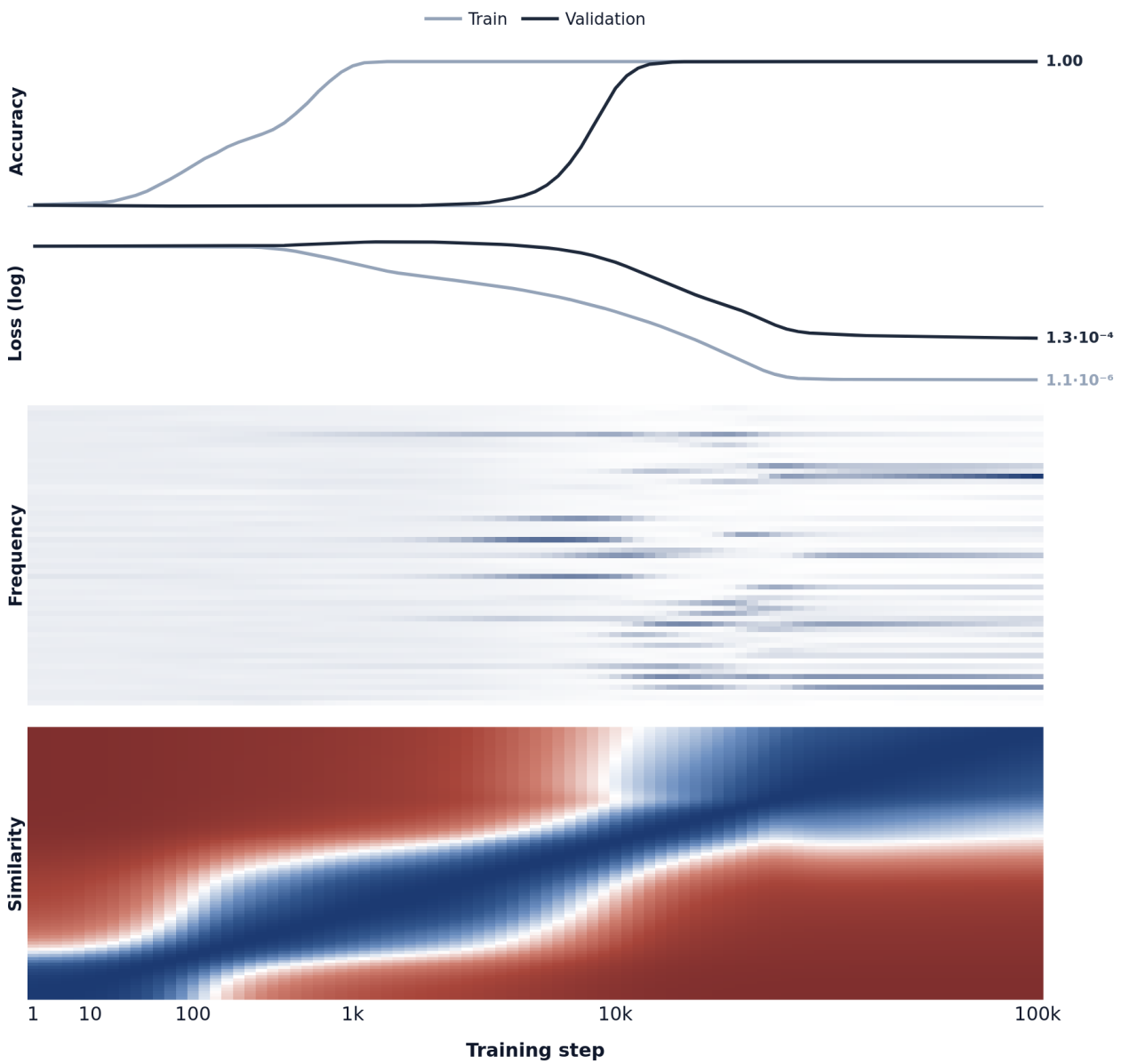

Figure 5: Modular addition training tracked by accuracy and loss (top), the Fourier components of the embeddings (middle), and pairwise tensor similarity between checkpoints (bottom); all panels share the log-scale training-step axis. As in Figure 1, similarity runs from 0 (orthogonal, red) to 1 (colinear, blue). Three diagonal blocks in the similarity matrix correspond to initialisation, memorisation, and the converged solution.

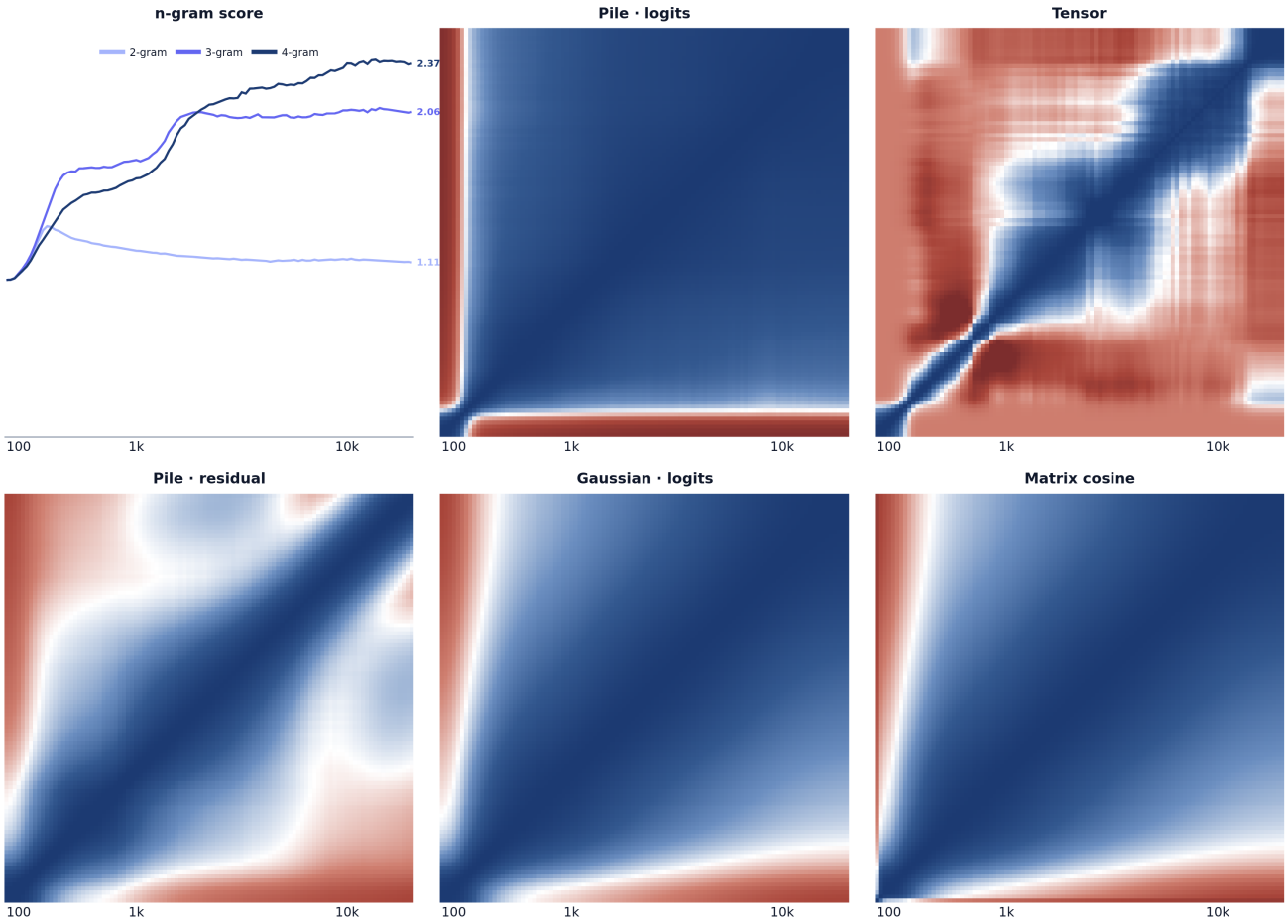

Figure 6: Five similarity measures across 101 log-spaced checkpoints of a two-layer bilinear attention transformer on The Pile. We track the n-gram score as a heuristic measure of change, which shows continued changes throughout most of training. Both the Pile · residual and the Pile · logits are behavioural similarities on pre-unembedding activations and on logits over Pile tokens; Gaussian · logits is the same on logits driven by matched-σ Gaussian inputs. Tensor is the closed-form weight-based similarity; Matrix cosine is the Frobenius cosine of flattened weights.

E.1.3 Loss

For a token sequence $S_i = (t_1^i, \ldots, t_T^i)$, the empirical next-token loss at position k is

$$\ell_{n,k}(w) = \frac{1}{n} \sum_{i=1}^{n} - \log \operatorname{softmax}\big(f_w(S_{\leq k}^i)\big)\big[t_{k+1}^i\big], \qquad k = 1, \ldots, T-1.$$

The training loss is

$$\ell_n(w) = \frac{1}{T-1} \sum_{k=1}^{T-1} \ell_{n,k}(w), \qquad T = n_{\text{ctx}} = 512,$$

with held-out loss defined analogously on the validation split.

E.1.4 Training

Training is performed as a single streaming pass over the shuffled, EOS-separated Pile stream. The main run uses $T_{\text{train}} = 20{,}000$ optimiser steps, batch size $B = 384$, and context length $n_{\text{ctx}} = 512$, corresponding to approximately 4×10^9 training tokens. Evaluation is performed using a cached validation split from the Pile.

Weight	Shape	Parameters
`embed.weight`	$(V, d_{\text{model}}) = (4096, 256)$	1,048,576
`attn.q1.weight`	$(256, 256)$	65,536
`attn.q1.bias`	$(256,)$	256
`attn.k1.weight`	$(256, 256)$	65,536
`attn.k1.bias`	$(256,)$	256
`attn.q2.weight`	$(256, 256)$	65,536
`attn.q2.bias`	$(256,)$	256
`attn.k2.weight`	$(256, 256)$	65,536
`attn.k2.bias`	$(256,)$	256
`attn.v.weight`	$(256, 256)$	65,536
`attn.o.weight`	$(256, 256)$	65,536
Layer subtotal	–	394,240
All layers ($\times L = 2$)	–	788,480
`final_norm` (`tok0_batch`)	–	0
`unembed.weight`	$(V, d_{\text{model}}) = (4096, 256)$	1,048,576
Total	–	**2,885,632**

Table 3: Attention-only language model parameter count.

Optimisation uses Muon for the internal attention projection weights, while the attention output projection, embedding, and unembedding parameters are optimised with AdamW. The Muon learning rate is 0.02, and the AdamW learning rate is 3×10^{-4}, with Adam betas $(0.9, 0.95)$, weight decay 0.1, and gradient clipping at 1.0. Both optimiser groups use a cosine schedule with a 500-step warmup and decay to 0.2 times the peak learning rate by step $20{,}000$. The model took 5 hours to train on an A10.

Hyperparameter	Category	Value
Dataset	Data	DSIR-filtered Pile-50M
V	Data	4096
n_{ctx}	Data	512
B	Data	384
N_{steps}	Data	20,000
Tokens seen	Data	$\approx 3.93 \times 10^9$
L	Model	2
H	Model	8
d_{model}	Model	256
Attention type	Model	bilinear
Attention scale	Model	0.35
RoPE base	Model	10,000
Norm	Model	`tok0_batch` before unembedding
Optimiser	Optimisation	Muon + AdamW
Muon LR	Optimisation	0.02
AdamW LR	Optimisation	3×10^{-4}
Weight decay	Optimisation	0.1
Scheduler	Optimisation	cosine decay with 500-step warmup

Table 4: Language-modelling transformer training, data, model, and optimisation hyperparameters.

E.2 METRICS TRACKED OVER TRAINING

E.2.1 N-gram Metrics

The n-gram metric measures whether the model has learned common fixed-length token patterns from the training data. First, the most frequent n-grams $(t_1, \ldots, t_n)$ are extracted from the training corpus. For each n-gram, the model is given the

prefix $(t_1, \ldots, t_{n-1})$ and evaluated on its ability to predict the final token t_n. Averaging this cross-entropy over frequent training n-grams gives the n-gram memorisation loss, denoted $l_{\text{ngram}}(n)$. The same prediction task is then evaluated on validation sequences, giving $l_{\text{test}}(n)$. The final score is

$$\text{ngram score}(n) = \frac{l_{\text{test}}(n)}{l_{\text{ngram}}(n)}.$$

A lower ratio indicates that the model performs similarly on validation n-gram prediction and frequent training n-gram prediction, while a higher ratio suggests stronger specialization to the frequent training patterns.

F ROBUSTNESS TO NON-GAUSSIAN INPUT DISTRIBUTIONS

In paragraph 2.3, we showed that tensor similarity coincides with a canonical behavioural metric, cosine similarity of empirical outputs, under the assumption of a perfectly Gaussian input distribution. But how robust is this to non-Gaussian input distributions? In this appendix, we investigate the question statistically.

Dataset We consider a toy task where models are trained to detect the index of the second-highest value (or *2nd argmax*) in a list of four numbers sampled from one of the following 9 input distributions:

- Gaussian: Standard Gaussian distribution (mean 0 and variance 1).
- Half Gaussian: Absolute value after sampling from a standard Gaussian distribution.
- Bimodal: Combination of two Gaussians designed to be bimodal.
- Uniform: Uniform distribution on $[-1, 1]$.
- Laplace: Laplace distribution with location parameter $\mu = 0$ and scale parameter $b = \frac{1}{\sqrt{2}}$.
- 'Sparse Spikes': A distribution where there is a 75% chance of drawing 0 and a 25% chance of drawing from a Gaussian distribution with mean 1 and variance 4.
- Permutations: The set of permutations of the list (1, 2, 3, 4).
- Correlated Gaussian: Standard Gaussian distribution with a correlation introduced between the four inputs, defined by a fixed positive-definite correlation matrix.
- 'Gaussian and -10': Standard Gaussian distribution for the first three inputs; a constant -10 for the last input.

Example: One possible sample from the 'Gaussian and -10' distribution is the following: $[-0.54, -0.19, 0.17, -10.00]$. For this data point, the correct output for the model is 1, since the second-highest value is -0.19, which lies at index 1.

Model Architecture The model consists of a toy one-layer bilinear network. No residuals, normalisation layers, or biases are included in the model.

Model Dimensions and Training Table 5 summarises the model dimensions and training hyperparameters. Across all five seeds, models trained on the three distributions which are *not* symmetric about zero (namely Half Gaussian, Permutations, and 'Gaussian and -10') achieved accuracy above 90%, while models trained on the six symmetric distributions plateaued at approximately 50% (with the exception of the 'Sparse Spikes' models, which plateaued around 39%). This is due to a fundamental limitation in the pure, single-layer bilinear architecture used: it cannot distinguish x from $-x$, preventing the model from meaningfully exceeding 50% accuracy on any distribution symmetric about zero.

Figure 7 shows Pearson correlations between tensor similarity and the cosine similarity of empirical outputs across different training checkpoints and seeds. While the correlation between tensor similarity and output similarity is strongest for Gaussian inputs, it exceeds 0.9 for eight out of the nine distributions tested, including distributions of varying shape and one with statistical dependence between input dimensions.

The exception is the correlation across the 'Gaussian and -10' dataset, with a measured correlation of r = 0.61. Intuitively, the reason for this discrepancy is that here, the last index is so low that it will virtually never point to the second highest value, so the model learns to never output that index. This means that a substantial portion of network weights do not influence the ultimate result, and the weight-based similarity metric we use here is less correlated with the empirical similarity.

These results suggest that the tensor similarity metric is a robust proxy for empirical similarity across a broad range of distributions, not just Gaussians, so long as the weights all meaningfully contribute toward the output.

Architecture	
input dim n	4
rank	32
n_layer	1
Training Parameters	
dropout	0.0
weight decay	0.0
batch size	512
learning rate	10^{-2}
optimizer	Adam
schedule	constant
train steps	4096
seeds	5

Table 5: Model architecture and training setup for the distribution robustness experiment.

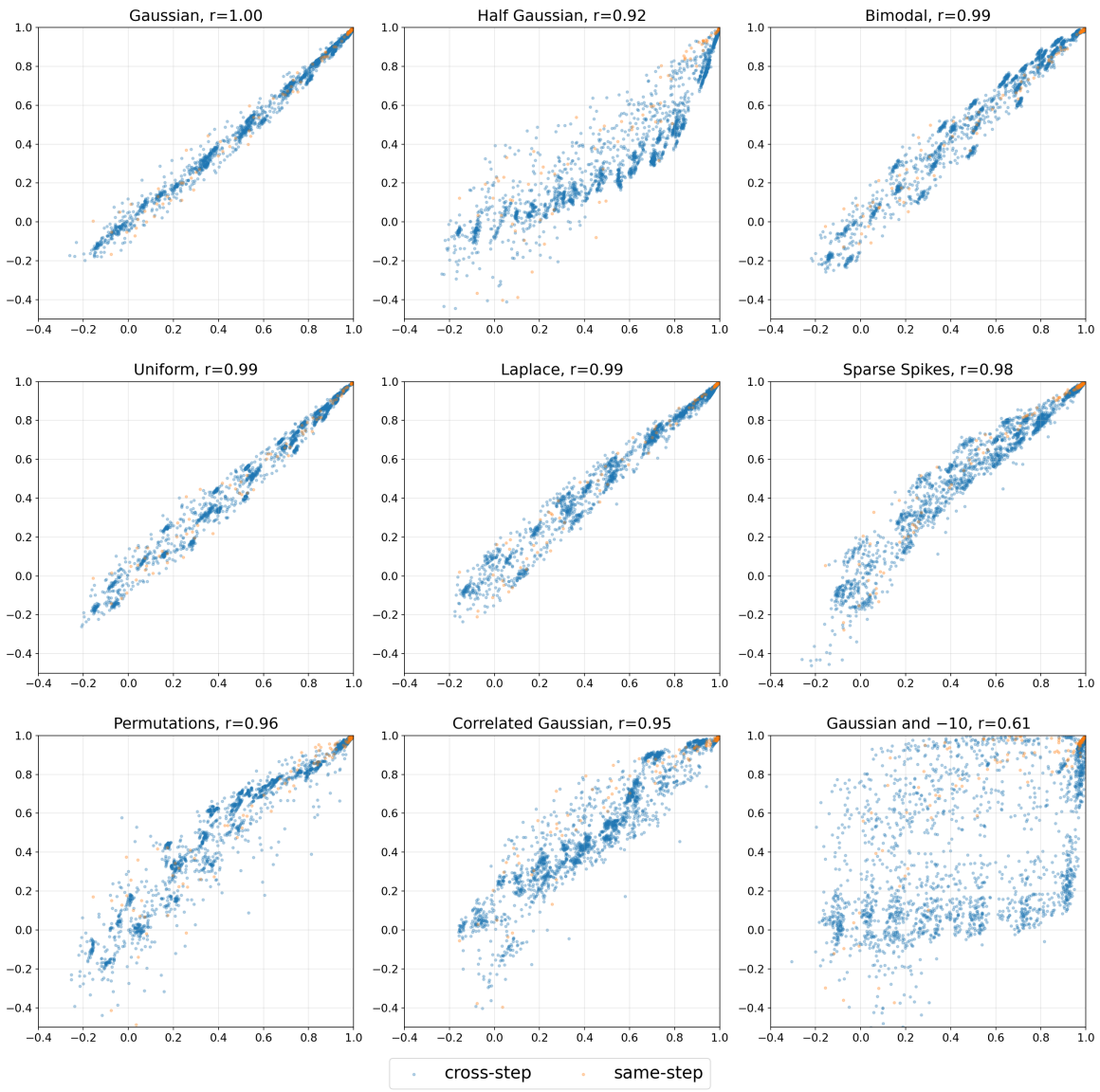

Figure 7: Five one-layer bilinear models were trained for each of nine input data distributions, and they were each sampled at 14 checkpoints. Pairs of these models were compared with a tensor-based similarity metric (shown on the x-axis) and a behavioural, output-based cosine similarity metric (shown on the y-axis). Pearson correlations are displayed above each scatterplot.