# The Geometry and Topology of Modular Addition Representations

**Gabriela Moisescu-Pareja*** McGill University, Mila
**Gavin McCracken*** McGill University, Mila
**Harley Wiltzer** McGill University, Mila
**Vincent Létourneau** Université de Montréal, Mila
**Colin Daniels** Independent
**Doina Precup** McGill University, Mila, Google DeepMind
**Jonathan Love** Leiden University

## Abstract

The *Clock* and *Pizza* interpretations, associated with neural architectures differing in either uniform or learnable attention, were introduced to argue that different architectural designs can yield distinct circuits for modular addition. Applying geometric and topological analyses to learned representations, we show that this is not the case: Clock and Pizza circuits are topologically and geometrically equivalent and are thus equivalent representations.

## 1 Introduction

Modular addition has become a standard testbed for toy models in interpretability [Nanda et al., 2023, Chughtai et al., 2023, Gromov, 2023, Morwani et al., 2024, McCracken et al., 2025a, He et al., 2024, Tao et al., 2025, Doshi et al., 2023]. The task is non-linearly separable yet mathematically well understood, making it ideal for researching how networks internally compute solutions. Two influential works examined modular addition in transformers, finding different architectures give rise to different circuits. Nanda et al. [2023] described a "Clock" interpretation, while Zhong et al. [2023] introduced the contrasting "Pizza" interpretation, each tied to architectural choices. We revisit these claims using geometric and topological analyses and find that Clock and Pizza learn topologically equivalent representations and thus the same circuit. In contrast, our new architecture, MLP-Concat, produces a genuinely different representation and thus a different circuit. More broadly, these results clarify how group-theoretic tasks can serve as controlled testbeds for studying a range of learning phenomena [McCracken, 2021, Zhang et al., 2022, Marchetti et al., 2024, Chughtai et al., 2023, Stander et al., 2024, McCracken et al., 2025b].

## 2 Background and Setup

We consider various neural network architectures for the task of modular addition, which means predicting the map $(a, b) \mapsto a + b \mod n$ for $a, b \in \mathbb{Z}_n$. For the sake of this paper, we fix $n = 59$. All architectures begin by embedding the inputs $a, b$ to vectors $\mathbf{E}_a, \mathbf{E}_b \in \mathbb{R}^{128}$ using a shared (learnable) embedding matrix. The architectures differ in how the embeddings are then processed: **MLP-Add** immediately passes $\mathbf{E}_a + \mathbf{E}_b$ through an MLP, **MLP-Concat** immediately passes the concatenation $\mathbf{E}_a \oplus \mathbf{E}_b \in \mathbb{R}^{256}$ through an MLP, and **Clock** and **Pizza**, introduced by Zhong et al.

*Equal contribution. {gabriela.moisescu-pareja, gavin.mccracken}@mail.mcgill.ca

Proceedings of the 1st Conference on Topology, Algebra, and Geometry in Data Science(TAG-DS 2025).

[2023] pass $\mathbf{E}_a, \mathbf{E}_b$ through a self-attention layer before the MLP. Particularly, **Pizza** [Zhong et al., 2023] uses a fixed, constant attention matrix, while **Clock** [Nanda et al., 2023] uses the standard scaled softmax attention. We refer to transformer-based architectures associated with the Clock as **Attention 1.0** and those associated with Pizza as **Attention 0.0**, respectively.

It is well-known [Nanda et al., 2023, Zhong et al., 2023] that the above architectures learn circuits with learned embeddings of the following form,

$$\mathbf{E}_a = [\cos(2\pi fa/n), \sin(2\pi fa/n)], \quad \mathbf{E}_b = [\cos(2\pi fb/n), \sin(2\pi fb/n)]. \tag{1}$$

What distinguishes them is how the embeddings are *transformed* post-attention. Treating the attention as a blackbox and looking at its output $\mathbf{E}_{ab}$, the two claims follow. **Clock** computes the *angle sum*,

$$\mathbf{E}_{ab} = [\cos(2\pi f(a+b)/n), \sin(2\pi f(a+b)/n)] \tag{2}$$

encoding the modular sum on the unit circle, which needs second-order interactions (e.g., multiplying embedding components via sigmoidal attention). In **Pizza**, $\mathbf{E}_{ab}$ adds the embeddings directly as $\mathbf{E}_a + \mathbf{E}_b$, giving:

$$\mathbf{E}_{ab} = [\cos(2\pi fa/p) + \cos(2\pi fb/p), \sin(2\pi fa/n) + \sin(2\pi fb/n)], \tag{3}$$

producing a *vector addition* on the circle, which is entirely linear in the embeddings. McCracken et al. [2025a] showed that, across **Clock**, **Pizza**, and **MLP-Concat** architectures, first layer neurons then take the form of so-called *simple-neurons*, producing pre-activations $N_i(a, b)$ given by

$$N_i(a, b) = \cos(2\pi fa/p + \phi_a^i) + \cos(2\pi fb/p + \phi_b^i), \tag{4}$$

where frequencies $f$ and phases $\phi_a^i, \phi_b^i$ are learned across training.

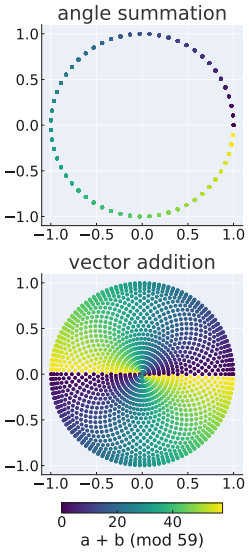


Figure 1: Clock and Pizza's analytical forms. Points are $\mathbf{E}_{ab}$ (cf. (2), (3)), colored by $(a + b) \mod 59$.

## 3 Phase Distribution Dictates Representation Manifolds

In the simple neuron model the only degrees of freedom beyond the frequency of a neuron are the learned *phases*. The phase distribution across simple neurons in a cluster distinguishes the representation manifolds.

**Informal Theorem 1** (Disc vs. Torus)**.** *Fix a frequency cluster of simple neurons with phases $\phi_a^i, \phi_b^i$ in $[0, 2\pi)$ for neuron $i$, and consider pre-activations over input pairs $(a, b)$. Then, almost surely, the representations lie on low-dimensional* manifolds*:*

*(Disc) If $\phi_a^i = \phi_b^i = \phi^i$ for all $i$, they lie on a 2D disc (in $\mathbb{R}^2$) with coordinates $\big(\cos\theta_a + \cos\theta_b,\ \sin\theta_a + \sin\theta_b\big)$.*

*(Torus) If $\phi_a^i, \phi_b^i$ can be independent across $i$, they lie on a 2D torus (in $\mathbb{R}^4$) with coordinates $\big(\cos\theta_a,\ \sin\theta_a,\ \cos\theta_b,\ \sin\theta_b\big)$.*

For the formal statement and proof of informal theorem 1 see Appendix A. This result motivates the empirical evaluations we propose in the next section.

## 4 Methodology

We analyze the structure of learned representations using two empirical methods: phase distributions and their induced topological structure. See Appendix B for additional details and computational methodology.

**Phase Alignment Distributions.** We propose the *Phase Alignment Distribution* (PAD). To a given architecture, a PAD is a distribution over $\mathbb{Z}_n \times \mathbb{Z}_n$. Samples of this distribution are drawn as follows:

1. Sample a random initialization and train the network, then sample a neuron uniformly.
2. Return pair $(a, b) \in \mathbb{Z}_n \times \mathbb{Z}_n$ achieving the largest activation in the resulting neuron.

A PAD illustrates, across independent training runs and neuron clusters, how often activations are maximized on the $a = b$ diagonal—that is, it depicts how often learned phases align *i.e.* $\phi_a = \phi_b$. Even beyond inspecting the proximity of samples to this diagonal, we propose to compare the PADs of architectures according to metrics on the space of distributions over $\mathbb{Z}_n \times \mathbb{Z}_n$, giving an even more precise comparison. In the following section, we will provide estimates of the PADs for the aforementioned architectures, as well as distances between PADs under the *maximum mean discrepancy* [Gretton et al., 2012, MMD]—a family of metrics with tractable unbiased sample estimators.

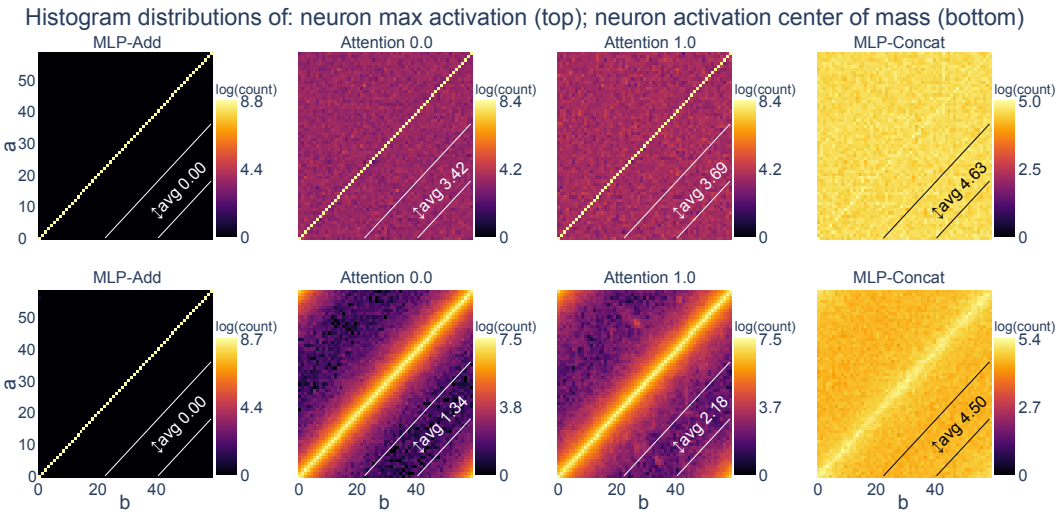


Figure 2: Log-density heatmaps for the distribution of neuron maximum activations (top) and activation center of mass (bottom) across 703 models. Attention 0.0 and 1.0 architectures show modest off-diagonal spread relative to MLP-Add, but remain constrained by architectural bias toward diagonal alignment. MMD scores between Attention 0.0 and 1.0 are 0.0237 (row 1) and 0.0181 (row 2), indicating near identical distributions (see Table 1).

**Betti numbers.** *Betti numbers* distinguish the structure of different stages of circuits across layers. The $k$-th Betti number $\beta_k$ of a topological manifold counts $k$-dimensional holes: $\beta_0$ counts connected components, $\beta_1$ counts loops, $\beta_2$ counts voids enclosed by surfaces. For reference, a disc has Betti numbers $(\beta_0, \beta_1, \beta_2) = (1, 0, 0)$, a circle has $(1, 1, 0)$, and a 2-torus has $(1, 2, 1)$. We estimate the distribution over Betti number vectors corresponding to the set of neurons in a given layer to distinguish the structure of the layers.

## 5 Discussion and Conclusion

This work set out to clarify whether the **Clock** and **Pizza** interpretations (corresponding to Attention 1.0 and 0.0 architectures respectively) for modular addition implement distinct circuits or merely reflect superficial differences. Using geometric and topological analyses, we find that their internal representations are in fact highly similar. The PAD analysis (Fig. 2) shows that both architectures produce distributions closely aligned with the $a = b$ diagonal, nearly indistinguishable under MMD (Appendix C for additional experiments and statistical significance). Betti number analysis (Fig. 3) confirms that their topological trajectories across layers converge in the same way, while MLP-Concat follows a different path. Thus, the distinction between "Clock" and "Pizza" is largely illusory: both instantiate the same underlying circuit, differing more from MLP-Concat than from each other and MLP-Add. Moreover, in Appendix D we evaluate prior metrics by Zhong et al. [2023] and find that our geometric and topological methods are more robust in distinguishing Clock, Pizza, MLP-Add vs. MLP-Concat which is genuinely different.

More broadly, we show that architectures with trainable embeddings approximate the torus-to-circle map, with differences arising in how this map *factors* through intermediate representations. This perspective connects to the *manifold hypothesis* [Bengio et al., 2013], which posits that networks discover low-dimensional manifolds underlying data. Our results demonstrate how high-level architectural choices can induce the learning of the entire manifold or a projection of it, where the torus of MLP-Concat is the entire manifold and the vector projection disc-like representation of MLP-Add, Attention 1.0 and 0.0 is a projection.

While our analysis is restricted to modular addition, it illustrates how architectural bias shapes representational geometry, with broader implications for understanding representations and interpreting them. Future work should try to understand why these representations are learned, what aspects of architecture induce representational changes vs. those that don't, as well as whether there's a

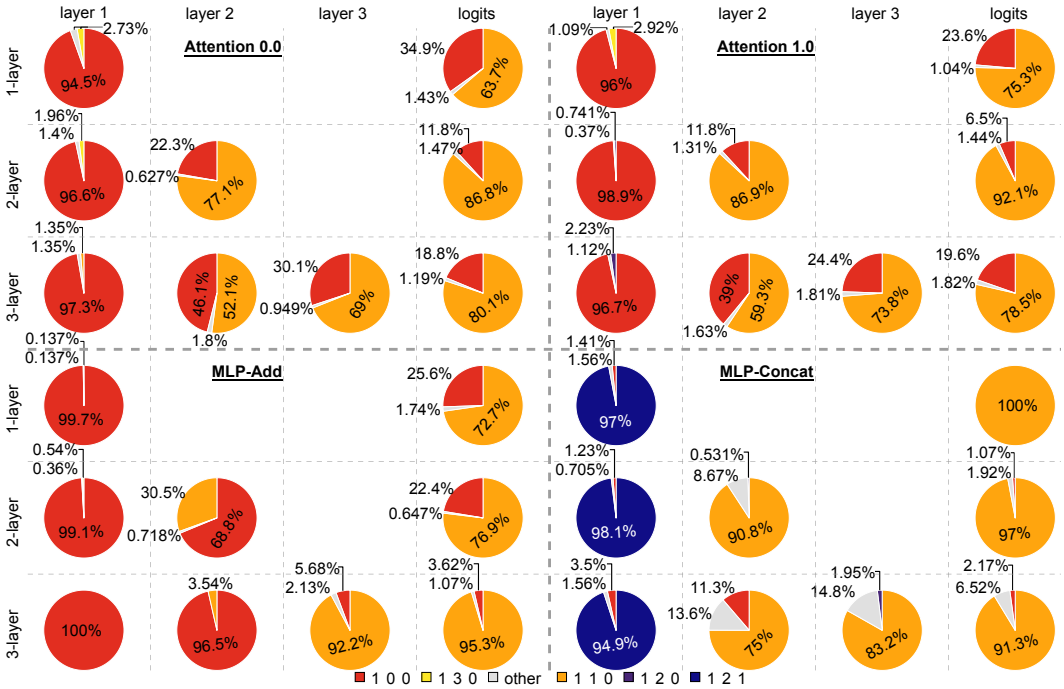


Figure 3: Betti number distributions across layers for 1-, 2-, and 3-layer models (100 seeds for each model). In layer 1, MLP-Add, Attention 0.0, and Attention 1.0 mostly yield disc-like representations, while MLP-Concat produces a torus. From the second layer onward, MLP-Add and both Attention variants converge to either a disc or a circle: the circle reflects the logits topology (correct answer), while the disc is a transient intermediate that can persist in later layers. MLP-Concat instead transitions directly to the circle. Across depth, Attention 0.0 and 1.0 are nearly identical with the latter having fewer transient discs.

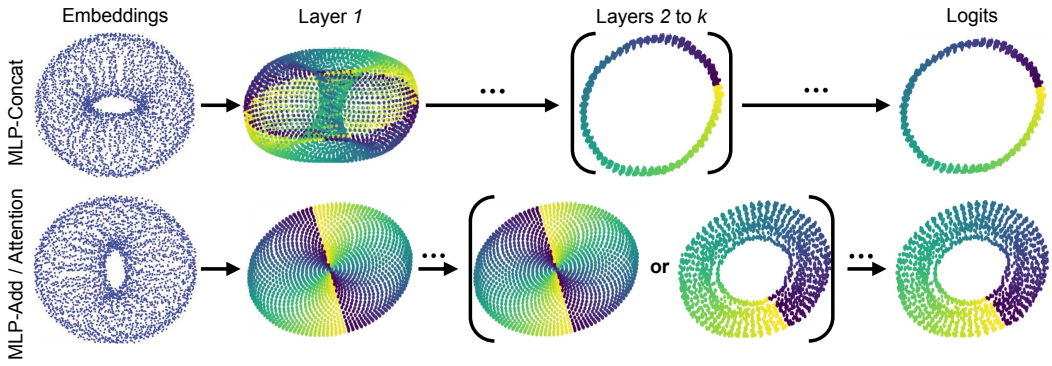


Figure 4: PCA projections of embeddings, intermediate pre-activations, and logits for MLP-Concat (top) and MLP-Add/Attention variants (bottom). In layer 1, MLP-Add and Attention models form discs like vector addition (Fig. 1), while MLP-Concat forms a torus (Fig. 3); in later layers and at logits representations in all models approach circles (angle summation, Fig. 1), but MLP-Concat immediately reaches a thin angle summation circle after layer 1.

guiding universal principal that unifies all these representations–because the embeddings and logits are always the same and the vector addition disc is a projection of the torus.

## References


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. URL https://openreview.net/forum?id=9XFSbDPmdW.

Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. In *International Conference on Machine Learning*, pages 6243–6267. PMLR, 2023.

Andrey Gromov. Grokking modular arithmetic. *arXiv preprint arXiv:2301.02679*, 2023.

Depen Morwani, Benjamin L. Edelman, Costin-Andrei Oncescu, Rosie Zhao, and Sham M. Kakade. Feature emergence via margin maximization: case studies in algebraic tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=i9wDX850jR.

Gavin McCracken, Gabriela Moisescu-Pareja, Vincent Letourneau, Doina Precup, and Jonathan Love. Uncovering a universal abstract algorithm for modular addition in neural networks, 2025a. URL https://arxiv.org/abs/2505.18266.

Tianyu He, Darshil Doshi, Aritra Das, and Andrey Gromov. Learning to grok: Emergence of in-context learning and skill composition in modular arithmetic tasks. *arXiv preprint arXiv:2406.02550*, 2024.

Tao Tao, Darshil Doshi, Dayal Singh Kalra, Tianyu He, and Maissam Barkeshli. (how) can transformers predict pseudo-random numbers? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=asDx9sPAUN.

Darshil Doshi, Aritra Das, Tianyu He, and Andrey Gromov. To grok or not to grok: Disentangling generalization and memorization on corrupted algorithmic datasets. *arXiv preprint arXiv:2310.13061*, 2023.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S5wmbQc1We.

Gavin McCracken. *Using Exact Models to Analyze Policy Gradient Algorithms*. McGill University (Canada), 2021.

Yi Zhang, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*, 2022.

Giovanni Luca Marchetti, Christopher J Hillar, Danica Kragic, and Sophia Sanborn. Harmonics of learning: Universal fourier features emerge in invariant networks. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 3775–3797. PMLR, 2024.

Dashiell Stander, Qinan Yu, Honglu Fan, and Stella Biderman. Grokking group multiplication with cosets. In *Forty-first International Conference on Machine Learning*, 2024.

Gavin McCracken, Sihui Wei, Gabriela Moisescu-Pareja, Harley Wiltzer, and Jonathan Love. The representations of deep neural networks trained on dihedral group multiplication. In *NeurIPS 2025 Workshop on Symmetry and Geometry in Neural Representations*, 2025b. URL https://openreview.net/forum?id=weKecpFYNf.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ulrich Bauer. Ripser: efficient computation of Vietoris-Rips persistence barcodes. *J. Appl. Comput. Topol.*, 5(3):391–423, 2021. ISSN 2367-1726. doi: 10.1007/s41468-021-00071-5. URL https://doi.org/10.1007/s41468-021-00071-5.

Vin de Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Dualities in persistent (co)homology. *Inverse Problems*, 27(12):124003, November 2011. ISSN 1361-6420. doi: 10.1088/0266-5611/27/12/124003. URL http://dx.doi.org/10.1088/0266-5611/27/12/124003.

Christopher Tralie, Nathaniel Saul, and Rann Bar-On. Ripser.py: A lean persistent homology library for python. *The Journal of Open Source Software*, 3(29):925, Sep 2018. doi: 10.21105/joss.00925. URL https://doi.org/10.21105/joss.00925.

# A Theoretical result

## A.1 Canonical manifolds

We will now focus on networks with a single learnable embedding matrix, matching the setups of Nanda et al. [2023], Zhong et al. [2023], McCracken et al. [2025a]. Our analysis will center on the *representation manifolds* in a frequency cluster $f$ coming from the preactivations $h^{\text{pre}}_{\ell,f}(a,b)$ at layer $\ell$ and the logits $l_f(a,b)$. The corresponding representation manifolds are, explicitly,

$$\mathcal{M}^{\text{pre}}_{\ell,f} := \left\{h^{\text{pre}}_{\ell,f}(a,b) : (a,b) \in \mathbb{Z}_n^2\right\} \subset \mathbb{R}^{d_{\ell,f}}; \quad \text{and} \quad \mathcal{M}^{\text{logit}}_{f} := \left\{l_f(a,b) : (a,b) \in \mathbb{Z}_n^2\right\} \subset \mathbb{R}^n,$$

where $d_{\ell,f}$ is the number of neuron in the frequency cluster $f$ at layer $\ell$. Our thesis is that under the simple neuron model of (4) introduced by McCracken et al. [2025a] and a simple application of symmetry corresponding to the interchangeability of $a, b$ in $a + b \mod n$, the exact structure of the $\mathcal{M}^{\text{pre}}_{1,f}$ manifolds and how they are mapped from inputs $a, b$ can be revealed. Particularly, we will show that under this model, $\mathcal{M}^{\text{pre}}_{1,f}$ always encodes the torus $\mathbb{T}^2$ or vector addition disk of Figure 1—that is, the pizza.

## A.2 Simple neuron phase distribution dictates representation manifold

Under the simple neuron model, for any frequency cluster $f$, the only degrees of freedom in the resulting preactivations lie in the maps $(a,b) \mapsto (\phi^L, \phi^R)$ for $a, b \in \mathbb{Z}_n$ learned by neural networks. Given that modular addition is commutative, one might expect to see a form of symmetry with respect to $\phi^L, \phi^R$. Particularly, one might expect that $\phi^L \equiv \phi^R$ for all $a, b$ (since swapping the inputs should have no effect on the output), or at the very least that the random variables $\phi^L, \phi^R$ are identically distributed for $A, B \sim \text{Uniform}(\mathbb{Z}_n)$. It turns out, as we show in the following theorem (whose proof is given in Appendix A.3), that the resulting manifold $\mathcal{M}^{\text{pre}}_{1,f}$ takes an easily characterizable form *almost surely* in this event. We devote §**??** to validating that the phase maps indeed satisfy these properties in practice—allowing us to easily analyze the geometry of representations across thousands of trained neural networks.

Before stating the theorem, let us introduce some notation that will be useful. Under the simple neuron model, a neuron indexed $i$ belonging to a neuron cluster with frequency $f$ maps $(a,b) \in \mathbf{Z}_p^2$ to $\cos(\theta_a + \Phi_i^L) + \cos(\theta_b + \Phi_i^R)$, where $\theta_a = 2\pi f a/p$. The notation $\Phi_i$ is meant to evoke that we model these phases as random variables; these are random due to random initialization and random gradient updates. The joint distribution of $(\Phi_i^L, \Phi_i^R)$ is denoted $\mu_i^{a,b} \in \Delta([0, 2\pi]^2)$.

**Theorem 1.** *Let $f \in \mathbf{Z}_p$ for $p \geq 3$, and consider the frequency cluster at layer 1. Let $m$ denote the number of neurons in this cluster, and assume $m \geq 2$. Define the matrix $X \in \mathbb{R}^{p^2 \times m}$ according to $X_{(a,b),i} = \cos(\theta_a + \phi_i^L) + \cos(\theta_b + \phi_i^R)$, denoting the simple neuron preactivations. Assume $\phi_i^L, \Phi_i^{L,b}$ are identically distributed for each neuron $i \in \{1, \ldots, m\}$ in this cluster, and that the support of $\mu_i^{a,b}$ has positive (Lebesgue) measure. Then the following hold almost surely:*

1. (Perfect phase correlation) *If $\Phi_i^{L,a}$ and $\Phi_i^{R,b}$ are perfectly correlated, in the sense that $\Phi_i^{L,b} \equiv \Phi_i^{R,b}$, then $X$ has a rank-2 factorization $X = V^{\text{disc}} W$ with $V^{\text{disc}} \in \mathbb{R}^{p^2 \times 2}$ satisfying*
$$V^{\text{disc}}_{(a,b)} = (\cos\theta_a + \cos\theta_b, \sin\theta_a + \sin\theta_b)^\top. \tag{5}$$

2. (Phase independence) *Otherwise, $X$ has a rank-4 factorization $X = V^{\text{torus}} W$ with $V^{\text{torus}} \in \mathbb{R}^{p^2 \times 4}$ given by*
$$V^{\text{torus}}_{(a,b)} = (\cos\theta_a, \sin\theta_a, \cos\theta_b, \sin\theta_b)^\top. \tag{6}$$

Geometrically, the disc can be viewed as a projection of the torus: $(x_1, x_2, x_3, x_4) \mapsto (x_1 + x_3, x_2 + x_4)$. Thus, the torus structure generalizes the vector-addition disc.

Having established this theorem, it is worth stepping back to contextualize its consequences. As shown by McCracken et al. [2025a], first layer preactivations are dominantly simple neurons. Theorem 1 shows that, under the symmetry properties of $\Phi_i^{L,a}$ and $\Phi_I^{R,b}$ posited above, the preactivations

have simple, low-dimensional structures: in the case of perfect phase correlation, the representation manifold can be compressed to $V^{\text{disc}}$, which is precisely the vector addition disc of Figure 1. In the case of phase independence, the representation manifold can be compressed to $V^{\text{torus}}$, which exactly encodes the torus $\mathbb{T}^2$.

**Remark 2.** *It is noteworthy that the Clock representation from Zhong et al. [2023]* cannot *occur under the hypotheses of Theorem 1. The remainder of the paper demonstrates that these hypotheses are satisfied empirically with overwhelming probability. Thus, while the Clock circuit of Zhong et al. [2023] is theoretically plausible, it does not occur naturally in practice. On the other hand, the possibility of the* torus *representation has not previously been identified in the literature.*

A notable consequence of this result is that the geometry and topology of representation manifolds can be characterized by simply investigating the distributions $\mu_i^{a,b}$ of the learned phases. As we describe in §4, this can be done quantitatively, allowing us to derive statistical likelihoods of neural circuits arising over thousands of initializations across architectures.

### A.3 Proof of theorem

The proofs of both cases of Theorem 1 follow the same pattern: apply an angle sum formula to the entries of the pre-activation matrix, realize this matrix as a product of 2 low-rank matrices and use the assumption of uniformity of the phase variables to deduce full rank of the composition.

For integers $p \geq 3$ and $m \geq 2$, consider the $p^2 \times m$ data matrix of the pre-activations of the model network with simple neurons (seee equation 4 and 1).

$$X_{(a,b),i} = \cos(\theta_a + \Phi_i^{L,a}) + \cos(\theta_b + \Phi_i^{R,b}), \qquad \theta_t := \frac{2\pi t}{p},\ (a,b) \in \{0,\dots,p-1\}^2.$$

and using the identity $\cos(x+y) = \cos x \cos y - \sin x \sin y$, we have

$$X_{(a,b),i} = \cos(\theta_a)\cos(\Phi_i^{L,a}) - \sin(\theta_a)\sin(\Phi_i^{L,a}) + \cos(\theta_b)\cos(\Phi_i^{R,b}) - \sin(\theta_b)\sin(\Phi_i^{R,b}) \quad (7)$$

Next, we show the details specific to each of the cases: disc and torus.

**Proof of Theorem 1 (Disc)**

*Proof.* By assumption, $\Phi_i^{L,a} = \Phi_i^{R,b} = \phi_i$ for all $i$. Then, equation 7 becomes

$$X_{(a,b),i} = (\cos\theta_a + \cos\theta_b)\cos\phi_i - (\sin\theta_a + \sin\theta_b)\sin\phi_i \quad (8)$$

Notice then that $X = VW$ for the matrices $V$ and $W$ defined by the following row and column vectors respectively

$$V_{(a,b),:} := [\cos(\theta_a) + \cos(\theta_b), \sin(\theta_a) + \sin(\theta_b)] \quad (9)$$

$$W_{:,i} := \begin{bmatrix} \cos(\phi_i) \\ -\sin(\phi_i) \end{bmatrix}. \quad (10)$$

Now we show they both have rank 2 and the kernel of $W$ intersect the image of $V$ trivially. The rank 2 of $V$ follows from the independence of $\cos$ and $\sin$ and the rank 2 of $W$ is true almost surely following the the independence of $\cos$ and $\sin$ and the hypothesis that $\Phi_i^{L,a}$ and $\Phi^{R,b}$ have uncountable support.

Suppose $\langle V_{(a,b),:}, W_{:,i}\rangle = 0$ for some $(a,b)$ and all $i$, that means $\cos(\theta_a + \phi_i) = -\cos(\theta_b + \phi_i)$ for all $i$. From the assumption the random variables $\phi^L$ and $\phi^R$ are not discrete, this event has probability 0, so the kernel of $W$ intersects the image of $V$ trivially and $X = VW$ has rank 2.

**Proof of Theorem 1 (Torus)**
Equation 7 shows $X = VW$ for the matrices $V, W$ defined by rows and columns respectively

$$V_{(a,b),:} = [\cos(\theta_a), \sin(\theta_a), \cos(\theta_b), \sin(\theta_b)] \quad (11)$$

$$W_{:,i} = \begin{bmatrix} \cos(\Phi_i^{L,a}) \\ -\sin(\Phi_i^{L,a}) \\ \cos(\Phi_i^{R,b}) \\ -\sin(\Phi_i^{R,b}) \end{bmatrix}. \quad (12)$$

The proof that $X$ has rank 4 is the same as the one for the respective statement in theorem 1 (uniformity of phases give the rank of $V$ and $W$ and the independence of the image of $V$ and kernel of $W$).

□

# B Additional experimental details

## B.1 Training hyperparameters.

All models are trained with the Adam optimizer Kingma and Ba [2014]. Number of neurons per layer in all models is 1024. Batch size is 59. Train/test split: $90\%/10\%$.

**Attention 1.0**

- Learning rate: 0.00075
- L2 weight decay penalty: 0.000025

**Attention 0.0**

- Learning rate: 0.00025
- L2 weight decay penalty: 0.000001

**MLP-Add and MLP-Concat**

- Learning rate: 0.0005
- L2 weight decay penalty: 0.0001

## B.2 Constructing representations

In all networks, we cluster neurons together and study the entire cluster at once McCracken et al. [2025a]. This is done by constructing an $n \times n$ matrix, with the value in entry $(a, b)$ corresponding to the preactivation value on datum $(a, b)$. A 2D Discrete Fourier Transform (DFT) of the matrix gives the key frequency $f$ for the neuron. The cluster of preactivations of all neurons with key frequency $f$ is the $n^2 \times |\text{cluster } f|$ matrix, made by flattening each neurons preactivation matrix and stacking the resulting vector for every neuron with the same key frequency.

## B.3 Persistent homology

We compute these using **persistent homology**, applied to point clouds constructed from intermediate representations at different stages of the circuit, as well as the final logits. This yields a compact topological signature that captures how the geometry of these representations evolves across layers, helping us identify when the underlying structure resembles a disc, torus, or circle. We use the Ripser library for these computations Bauer [2021], de Silva et al. [2011], Tralie et al. [2018].

For our persistent homology computations, we set the $k$-nearest neighbour hyperparameter to 250. Our point cloud consists of $59^2 = 3481$ points.

## B.4 Remapping procedure

**Neuron remapping [McCracken et al., 2025a].** For a simple neuron of frequency $f$, we define a canonical coordinate system via the mapping:

$$(a, b) \mapsto (a \cdot d, b \cdot d), \qquad \text{where } d := \left(\frac{f}{\gcd(f, n)}\right)^{-1} \mod \frac{n}{\gcd(f, n)}. \tag{13}$$

This inverse is the modular multiplicative inverse, i.e. for any $\mathbb{Z}_k$ let $x \in \mathbb{Z}_k$. Its inverse $x^{-1}$ exists if $\gcd(x, k) = 1$ and gives $x \cdot x^{-1} \equiv 1 \mod k$. This normalizes inputs relative to the neuron's periodicity and allows for qualitative and quantitative comparisons.

# C Statistical significance of our results

### C.0.1 Figure 2

We trained 703 models of each architecture, being MLP vec add, Attention 0.0 and 1.0, and MLP concat, and recorded the locations of the max activations of all neurons across all $(a, b)$ inputs to the network. We also computed the center of mass of each neuron as this doesn't always align with the max preactivation (though it tends to be close).

| Description | MMD | p-value | Interpretation |
|---|---|---|---|
| MLP vec add vs Attention 0.0 | 0.0968 | 0.0000 | Moderate difference; highly significant |
| MLP vec add vs Attention 1.0 | 0.1239 | 0.0000 | Clear difference; highly significant |
| MLP vec add vs MLP concat | 0.2889 | 0.0000 | Very strong difference; highly significant |
| Attention 0.0 vs Attention 1.0 | 0.0338 | 0.0000 | Subtle difference; highly significant |
| Attention 0.0 vs MLP concat | 0.1987 | 0.0000 | Strong difference; highly significant |
| Attention 1.0 vs MLP concat | 0.1723 | 0.0000 | Strong difference; highly significant |

(a) Row 1: Max activation

| Description | MMD | p-value | Interpretation |
|---|---|---|---|
| MLP vec add vs Attention 0.0 | 0.0583 | 0.0000 | Small difference; highly significant |
| MLP vec add vs Attention 1.0 | 0.0689 | 0.0000 | Moderate difference; highly significant |
| MLP vec add vs MLP concat | 0.2614 | 0.0000 | Very strong difference; highly significant |
| Attention 0.0 vs Attention 1.0 | 0.0210 | 0.0084 | Subtle difference; highly significant |
| Attention 0.0 vs MLP concat | 0.2126 | 0.0000 | Strong difference; highly significant |
| Attention 1.0 vs MLP concat | 0.1947 | 0.0000 | Strong difference; highly significant |

(b) Row 2: Center of mass

Table 1: Gaussian-kernel Maximum Mean Discrepancies (MMD) Gretton et al. [2012] and permutation p-values between the empirical distributions shown in Figure 2. For each architecture comparison, we sampled 20,000 points from each empirical distribution (derived from histogram-based neuron statistics), then computed the unbiased Gaussian-kernel MMD with a bandwidth chosen via the pooled median heuristic. Significance was assessed using 50,000 permutation tests per comparison.

### C.0.2 Figure 5: Torus distance from the max activation and center of mass to the line $a = b$

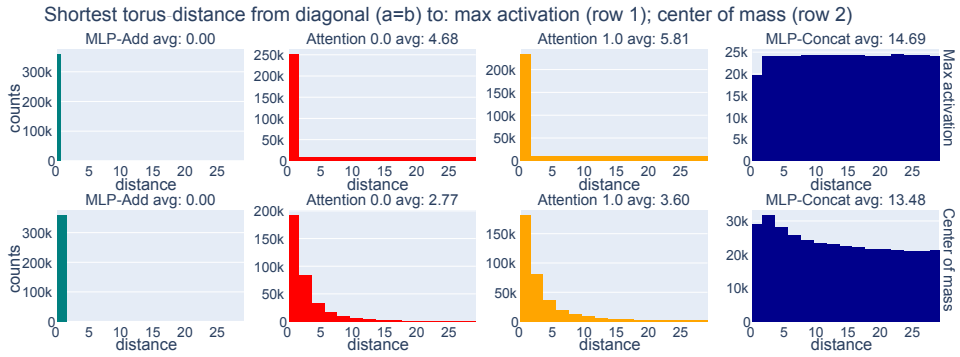


Figure 5: Histograms of torus-distance from each neuron's phase to the diagonal $a = b$, across 703 trained models. MLP-Add neurons align perfectly with the diagonal, Attention 0.0 and 1.0 show increasing off-diagonal spread, and MLP-Concat exhibits broadly distributed activations on the torus.

We trained 703 models of each architecture with 512 neurons in its hidden layer (MLP vec add, Attention 0.0 and 1.0, and MLP concat), and recorded the $a, b$ value of where the max activation of a neuron takes place across all $(a, b)$ inputs to the network and all neurons. We also computed the

$(a, b)$ values for the location of the center of mass of each neuron as this doesn't always align with the max preactivation (though it tends to be close). Then we compute the shortest torus distance from the point of the max activation or the center of mass, to the line $a = b$.

Table 2: Gaussian-kernel Maximum Mean Discrepancies (MMD) Gretton et al. [2012] and permutation p-values between the empirical distributions shown in Figure 5. For each architecture comparison, we sampled 2000 points from each empirical distribution (derived from histogram-based neuron statistics), then computed the unbiased Gaussian-kernel MMD with a bandwidth chosen via the pooled median heuristic. Significance was assessed using 5000 permutation tests per comparison.

(a) Row 1: Max activation

| Description | MMD | p-value | Interpretation |
|---|---|---|---|
| MLP vec add vs Attention 0.0 | 0.3032 | 0.0000 | Strong difference; highly significant |
| MLP vec add vs Attention 1.0 | 0.3888 | 0.0000 | Very strong difference; highly significant |
| MLP vec add vs MLP concat | 0.9508 | 0.0000 | Extremely strong difference; highly significant |
| Attention 0.0 vs Attention 1.0 | 0.0705 | 0.0000 | Moderate difference; highly significant |
| Attention 0.0 vs MLP concat | 0.6323 | 0.0000 | Very strong difference; highly significant |
| Attention 1.0 vs MLP concat | 0.5695 | 0.0000 | Very strong difference; highly significant |

(b) Row 2: Center of mass

| Description | MMD | p-value | Interpretation |
|---|---|---|---|
| MLP vec add vs Attention 0.0 | 0.7727 | 0.0000 | Extremely strong difference; highly significant |
| MLP vec add vs Attention 1.0 | 0.7517 | 0.0000 | Extremely strong difference; highly significant |
| MLP vec add vs MLP concat | 0.9148 | 0.0000 | Extremely strong difference; highly significant |
| Attention 0.0 vs Attention 1.0 | 0.0520 | 0.0006 | Moderate difference; highly significant |
| Attention 0.0 vs MLP concat | 0.7022 | 0.0000 | Very strong difference; highly significant |
| Attention 1.0 vs MLP concat | 0.6391 | 0.0000 | Very strong difference; highly significant |

# D Previous interpretability metrics [Zhong et al., 2023]

## D.1 Definitions

**Gradient symmetricity** measures, over some subset of input-output triples $(a, b, c)$, the average cosine similarity between the gradient of the output logit $Q_{(a,b,c)}$ with respect to the input embeddings of $a$ and $b$. For a network with embedding layer $\mathbf{E}$ and a set $S \subseteq \mathbb{Z}_p^3$ of input-output triples:

$$s_g = \frac{1}{|S|} \sum_{(a,b,c)\in S} \operatorname{sim}\left(\frac{\partial Q_{abc}}{\partial \mathbf{E}_a}, \frac{\partial Q_{abc}}{\partial \mathbf{E}_b}\right)$$

where $\operatorname{sim}(u, v) = \frac{u\cdot v}{\|u\|\|v\|}$ is the cosine similarity. It is evident that $s_g \in [-1, 1]$.

**Distance irrelevance** quantifies how much the model's outputs depend on the distance between $a$ and $b$. For each distance $d$, we compute the standard deviation of correct logits over all $(a, b)$ pairs where $a - b = d$ and average over all distances. It's normalized by the standard deviation over all data.

Formally, let $L_{i,j} = Q_{ij,i+j}$ be the correct logit matrix. The distance irrelevance $q$ is defined as:

$$q = \frac{\frac{1}{p}\sum_{d\in\mathbb{Z}_p} \operatorname{std}(\{L_{i,i+d} | i \in \mathbb{Z}_p\})}{\operatorname{std}(\{L_{i,j} | i, j \in \mathbb{Z}_p\})}$$

where $q \in [0, 1]$, with higher values indicating greater irrelevance to input distance.

## D.2 Results of evaluation

Figure 6 shows the mean and standard deviation of the gradient symmetricity and distance irrelevance metrics from Zhong et al. [2023]. Unlike Zhong et al. [2023], who report gradient symmetricity

results over a randomly selected subset of 100 input-output triples $(a, b, c) \in \mathbb{Z}_p^3$, we compute the metric exhaustively across **all** $59^3 = 205,370$ triples to add accuracy.

MLP-Add and MLP-Concat cluster on opposite extremes, implying the metrics just identify whether neurons have phases $\phi_a \neq \phi_b$. MLP-Add models have high gradient symmetricity and low distance irrelevance and MLP-Concat models have low gradient symmetricity and high distance irrelevance. Attention 1.0 models span a wide range between these extremes depending on two factors: 1) how well the frequencies they learned intersect and 2) how well neurons are able to get their activation center of mass away from the $\phi_a = \phi_b$ line. Attention 0.0 is closer to MLP-Add than Attention 1.0 because it's harder for this architecture to learn $\phi_a \neq \phi_b$. Notably, failure cases exist using both: **neither metric distinguishes between Attention 1.0 and 0.0 models.** While their metrics were intended to distinguish between Clock and Pizza, we show that they are not really able to and this makes sense because Clock and Pizza are not actually different.

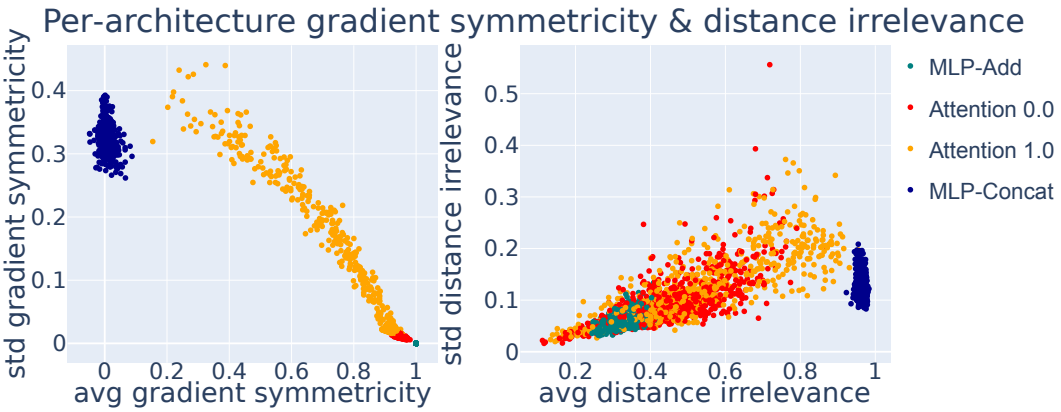


Figure 6: Evaluation of gradient symmetricity (left) and distance irrelevance (right). Each point shows the average (avg) and standard deviation (std) of one trained network. MLP-Add and MLP-Concat lie at nearly opposite extremes, while attention 0.0 and 1.0 overlap substantially. Gradient symmetricity separates Attention 1.0 better, but neither metric **always** distinguishes between Attention 1.0 and 0.0.

### D.2.1 MMD analysis

MMD results for these two metrics are reported below, again showing that the distance between attention 0.0 and attention 1.0 models is small. This is the case even those these metrics were chosen to differentiate between the two architectures.

Using just the x-axis (since the y-axis on those plots is the std dev) MMD results are presented next.

We can conclude that the attention transformers are far from vector addition, and very close to each other under all metrics.

[h]
Table 3: Permutation–test MMDs on the empirical gradient symmetricity and distance irrelevance distributions across all architectures. All p-values are $\leq 10^{-6}$ (reported as 0.0000).

(a) Gradient symmetricity (2-D: *avg* and *std*)

| Description | MMD | p-value | Interpretation |
|---|---|---|---|
| MLP vec add vs Attention 0.0 | 1.2725 | 0.0000 | Extremely strong difference; highly significant |
| MLP vec add vs Attention 1.0 | 0.9688 | 0.0000 | Extremely strong difference; highly significant |
| MLP vec add vs MLP concat | 1.3471 | 0.0000 | Extremely strong difference; highly significant |
| Attention 0.0 vs Attention 1.0 | 0.7750 | 0.0000 | Very strong difference; highly significant |
| Attention 0.0 vs MLP concat | 1.3503 | 0.0000 | Extremely strong difference; highly significant |
| Attention 1.0 vs MLP concat | 1.2360 | 0.0000 | Extremely strong difference; highly significant |

(b) Distance irrelevance (2-D: *avg* and *std*)

| Description | MMD | p-value | Interpretation |
|---|---|---|---|
| MLP vec add vs Attention 0.0 | 0.7534 | 0.0000 | Very strong difference; highly significant |
| MLP vec add vs Attention 1.0 | 0.7079 | 0.0000 | Very strong difference; highly significant |
| MLP vec add vs MLP concat | 1.2488 | 0.0000 | Extremely strong difference; highly significant |
| Attention 0.0 vs Attention 1.0 | 0.2078 | 0.0000 | Moderate difference; highly significant |
| Attention 0.0 vs MLP concat | 1.2255 | 0.0000 | Extremely strong difference; highly significant |
| Attention 1.0 vs MLP concat | 1.0990 | 0.0000 | Extremely strong difference; highly significant |

[h]
Table 4: Permutation-test MMDs on scatter-plot averages only (1-D). All $p$–values are $\leq 10^{-6}$, so every difference is "highly significant." Note that the distance between attention 0.0, attention 1.0, and MLP vec add is large, implying they are not performing vector addition.

(a) Row 3: Gradient symmetricity (avg only)

| Description | MMD | $p$-value | Interpretation |
|---|---|---|---|
| MLP vec add vs Attention 0.0 | 1.2755 | 0.0000 | Extremely strong difference; highly significant |
| MLP vec add vs Attention 1.0 | 0.9842 | 0.0000 | Extremely strong difference; highly significant |
| MLP vec add vs MLP concat | 1.3833 | 0.0000 | Extremely strong difference; highly significant |
| Attention 0.0 vs Attention 1.0 | 0.7726 | 0.0000 | Extremely strong difference; highly significant |
| Attention 0.0 vs MLP concat | 1.3802 | 0.0000 | Extremely strong difference; highly significant |
| Attention 1.0 vs MLP concat | 1.2559 | 0.0000 | Extremely strong difference; highly significant |

(b) Row 4: Distance irrelevance (avg only)

| Description | MMD | $p$-value | Interpretation |
|---|---|---|---|
| MLP vec add vs Attention 0.0 | 0.7739 | 0.0000 | Extremely strong difference; highly significant |
| MLP vec add vs Attention 1.0 | 0.7268 | 0.0000 | Very strong difference; highly significant |
| MLP vec add vs MLP concat | 1.2501 | 0.0000 | Extremely strong difference; highly significant |
| Attention 0.0 vs Attention 1.0 | 0.2109 | 0.0000 | Strong difference; highly significant |
| Attention 0.0 vs MLP concat | 1.2443 | 0.0000 | Extremely strong difference; highly significant |
| Attention 1.0 vs MLP concat | 1.1093 | 0.0000 | Extremely strong difference; highly significant |

# E GPU-optimized computations

## E.1 GPU-optimized center-of-mass in circular coordinates

Let $p$ be the grid size and for each neuron $n = 1, \ldots, N$ we have a pre-activation map

$$x_{i,j}^{(n)}, \quad (i, j = 0, \ldots, p-1).$$

Define nonnegative weights

$$w_{i,j}^{(n)} = \big|\, x_{i,j}^{(n)} \big|.$$

Let $f_n \in \{1, \ldots, \lfloor p/2 \rfloor\}$ be the dominant frequency for neuron $n$, and let

$$f_n^{-1} \text{ be the modular inverse of } f_n \text{ modulo } p, \qquad f_n\, f_n^{-1} \equiv 1 \pmod{p}.$$

Convert the row index $i$ and column index $j$ into angles ("un-wrapping" by $f_n^{-1}$):

$$\theta_i^{(n)} = \frac{2\pi}{p}\, f_n^{-1}\, i, \qquad \phi_j^{(n)} = \frac{2\pi}{p}\, f_n^{-1}\, j.$$

Form the two complex phasor sums

$$S_a^{(n)} = \sum_{i=0}^{p-1} \sum_{j=0}^{p-1} w_{i,j}^{(n)}\, \exp\big(i\, \theta_i^{(n)}\big),$$

$$S_b^{(n)} = \sum_{i=0}^{p-1} \sum_{j=0}^{p-1} w_{i,j}^{(n)}\, \exp\big(i\, \phi_j^{(n)}\big).$$

The arguments of these sums give the circular means of each axis:

$$\mu_a^{(n)} = \arg\big(S_a^{(n)}\big), \quad \mu_b^{(n)} = \arg\big(S_b^{(n)}\big),$$

where $\arg$ returns an angle in $(-\pi, \pi]$. To ensure a nonnegative result, normalize into $[0, 2\pi)$:

$$\mu^+ = \big(\mu + 2\pi\big) \bmod 2\pi.$$

Finally, map back from the angular domain to grid coordinates:

$$\mathrm{CoM}_a^{(n)} = \frac{p}{2\pi}\, \mu_a^{(n)+}, \qquad \mathrm{CoM}_b^{(n)} = \frac{p}{2\pi}\, \mu_b^{(n)+}.$$

This handles wrap-around at the boundaries automatically and weights each location $(i, j)$ by $|x_{i,j}^{(n)}|$, producing a smooth, circularly-aware center of mass. All tensor operations—angle computation, complex exponentials, and weighted sums—are expressed as parallel array primitives that JAX can JIT-compile and fuse into a single GPU kernel launch, eliminating Python-level overhead. By precomputing the angle grids and performing the phasor sums inside one jitted function, this implementation fully exploits GPU parallelism and memory coalescing for maximal throughput.

## E.2 GPU-vectorized distance irrelevance over all $n^2$ input pairs $(a, b)$

Let

$$\mathcal{I} \;=\; \big\{(a, b) \mid a, b \in \{0, \ldots, n-1\}\big\},$$

and order its elements lexicographically:

$$X = \big[(a_0, b_0),\, (a_1, b_1),\, \ldots,\, (a_{n^2-1}, b_{n^2-1})\big] \;\in\; \mathbb{Z}^{\,n^2 \times 2}.$$

A $n^2$ single batched forward pass on the GPU computes

$$\mathrm{Logits} \;=\; \mathrm{Transformer}\big(X\big) \;\in\; \mathbb{R}^{\,n^2 \times n},$$

producing all $n^2 \cdot n$ output logits in parallel. We then extract the "correct-class" logit for each input:

$$y_k \;=\; \mathrm{Logits}_{\,k,\,(a_k + b_k) \bmod n}, \qquad k = 0, \ldots, n^2 - 1.$$

Next we reshape $y$ into an $n \times n$ matrix $L$ by

$$L_{\,i,j} \;=\; y_k \quad \text{where} \quad i = (a_k + b_k) \bmod n, \;\; j = (a_k - b_k) \bmod n.$$

All of the above: embedding lookup, attention, MLP, softmax and the advanced indexing is implemented as two large vectorized kernels (the batched forward pass and the gather), so each of the $n^2$ inputs is handled in $O(1)$ time but fully in parallel on the GPU.

Finally, define

$$\sigma_{\text{global}} = \sqrt{\frac{1}{n^2}\sum_{i,j}\Big(L_{i,j} - \mu\Big)^2}, \quad \mu = \frac{1}{n^2}\sum_{i,j} L_{i,j},$$

and for each "distance" $j$

$$\sigma_j = \sqrt{\frac{1}{n}\sum_{i}\Big(L_{i,j} - \bar{L}_{\cdot j}\Big)^2}, \quad \bar{L}_{\cdot j} = \frac{1}{n}\sum_{i} L_{i,j}, \quad q_j = \frac{\sigma_j}{\sigma_{\text{global}}}.$$

We report

$$\overline{q} = \frac{1}{n}\sum_{j=0}^{n-1} q_j, \qquad \operatorname{std}(q) = \sqrt{\frac{1}{n}\sum_{j=0}^{n-1}\big(q_j - \overline{q}\big)^2}.$$

### E.3 GPU-optimized gradient symmetricity over all $n^3$ triplets $(a, b, c)$

Let

$$E \;\in\; \mathbb{R}^{\,n\times d} \quad \text{with} \quad d = 128$$

be the learned embedding matrix, and denote by

$$Q\big(E_a,\, E_b\big)_c$$

the scalar logit for class $c$ obtained by feeding the pair of embeddings $(E_a, E_b)$ into the model. We define the per-triplet gradient cosine-similarity as

$$S(a,b,c) \;=\; \frac{\big\langle \nabla_{E_a} Q(E_a,E_b)_c,\; \nabla_{E_b} Q(E_a,E_b)_c \big\rangle}{\big\|\nabla_{E_a} Q(E_a,E_b)_c\big\|\;\big\|\nabla_{E_b} Q(E_a,E_b)_c\big\|},$$

for all $(a, b, c) \in \{0, \ldots, n-1\}^3$.

To compute $\{S(a, b, c)\}$ over the full $n^3$ grid in one fused GPU kernel, we first form three index tensors

$$A_{i,j,k} = i, \quad B_{i,j,k} = j, \quad C_{i,j,k} = k, \quad i, j, k = 0, \ldots, n-1,$$

then flatten to vectors $a = \operatorname{vec}(A)$, $b = \operatorname{vec}(B)$, $c = \operatorname{vec}(C) \in \{0, \ldots, n-1\}^{n^3}$. We gather the embeddings

$$\mathsf{emb}_a = E[a] \in \mathbb{R}^{n^3\times d}, \quad \mathsf{emb}_b = E[b] \in \mathbb{R}^{n^3\times d},$$

and in JAX compute

$$\mathbf{g}_a \;=\; \mathrm{vmap}\big(\,(e_a,e_b,c) \mapsto \nabla_{E_a} Q(e_a,e_b)_c\big)(\mathsf{emb}_a,\, \mathsf{emb}_b,\, c),$$
$$\mathbf{g}_b \;=\; \mathrm{vmap}\big(\,(e_a,e_b,c) \mapsto \nabla_{E_b} Q(e_a,e_b)_c\big)(\mathsf{emb}_a,\, \mathsf{emb}_b,\, c),$$

each producing an $(n^3 \times d)$-shaped array. Finally the similarity vector is

$$\mathbf{S} = \frac{\mathbf{g}_a \odot \mathbf{g}_b}{\|\mathbf{g}_a\|\,\|\mathbf{g}_b\|} \;\in\; \mathbb{R}^{n^3},$$

and we report

$$\overline{S} = \frac{1}{n^3}\sum_{i=1}^{n^3} S_i, \qquad \sigma_S = \sqrt{\frac{1}{n^3}\sum_{i=1}^{n^3}\big(S_i - \overline{S}\big)^2}.$$

**Runtime.** Because we express $\mathbf{g}_a, \mathbf{g}_b$ and the subsequent dot-and-norm entirely inside a single `@jax.jit+ vmap` invocation, XLA lowers it to one GPU kernel that processes all $n^3$ triplets in parallel. The kernel dispatch cost is therefore $O(1)$, and each triplet's gradient and cosine computations are fused into vectorized instructions with constant per-element overhead. Although the total arithmetic work is $O(n^3)$, the full data-parallel execution means the wall-clock latency grows sub-linearly in $n^3$ and the per-triplet overhead remains effectively constant.