The Geometry and Topology of Modular Addition Representations

Anonymous Author(s)
Affiliation
Address
email

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. [2025], 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.

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. [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,

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

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

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. [2025] 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 ϕ_a^i, ϕ_b^i are learned across training.

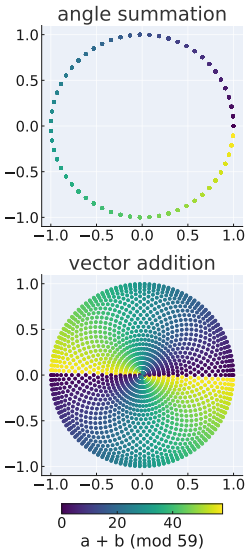

Figure 1: Clock and Pizza's analytical forms. Points are $\mathbf{E}_{ab}$ (cf. (2), (3)), colored by $(a + b) \bmod 59$.

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 ϕ_a^i, ϕ_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 ϕ_a^i, ϕ_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.

Betti numbers. *Betti numbers* distinguish the structure of different stages of circuits across layers. The k-th Betti number β_k of a topological manifold counts k-dimensional holes: β_0 counts connected components, β_1 counts loops, β_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.

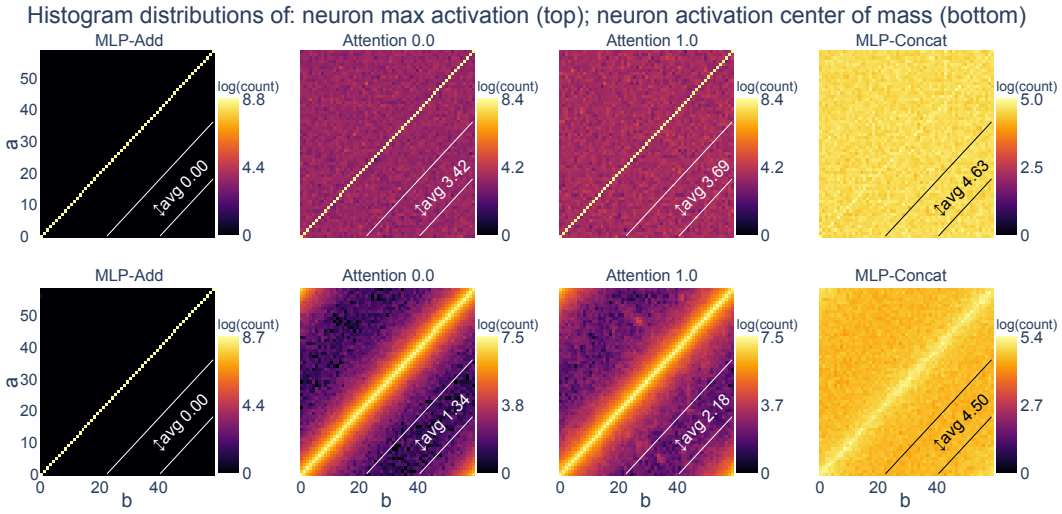

Figure 2: Log-density heatmaps for the distribution of neuron maximum activations (top) and activation center of mass (bottom) across 703 models. Attention 0.0 and 1.0 architectures show modest off-diagonal spread relative to MLP-Add, but remain constrained by architectural bias toward diagonal alignment. 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).

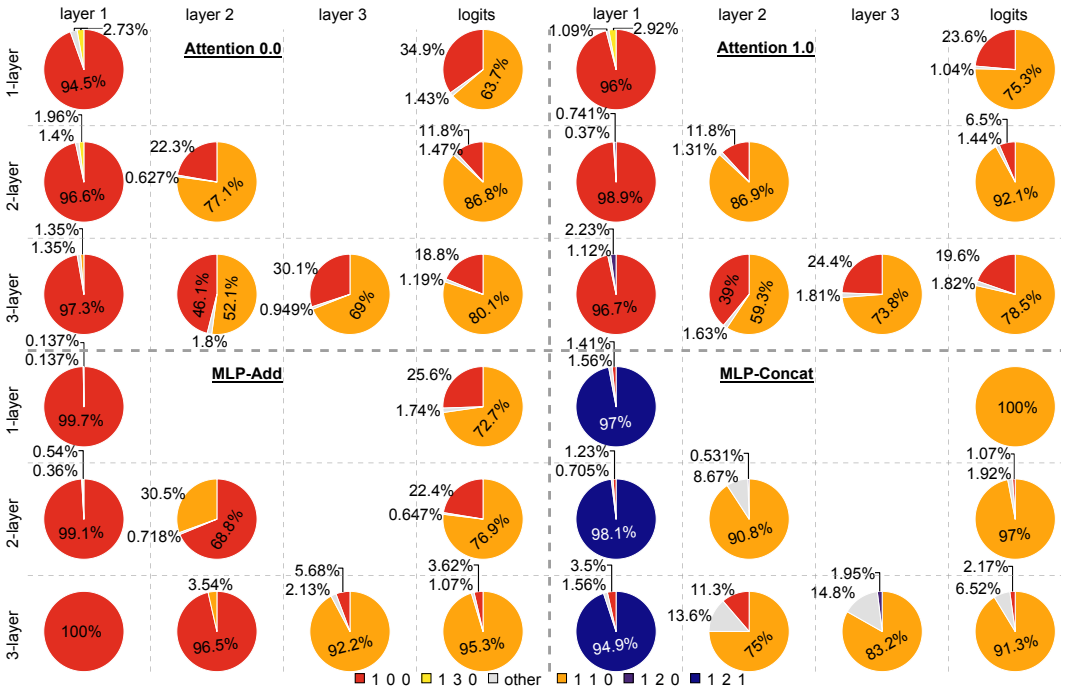

Figure 3: Betti number distributions across layers for 1-, 2-, and 3-layer models (100 seeds for each model). In layer 1, MLP-Add, Attention 0.0, and Attention 1.0 mostly yield disc-like representations, while MLP-Concat produces a torus. From the second layer onward, MLP-Add and both Attention variants 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.

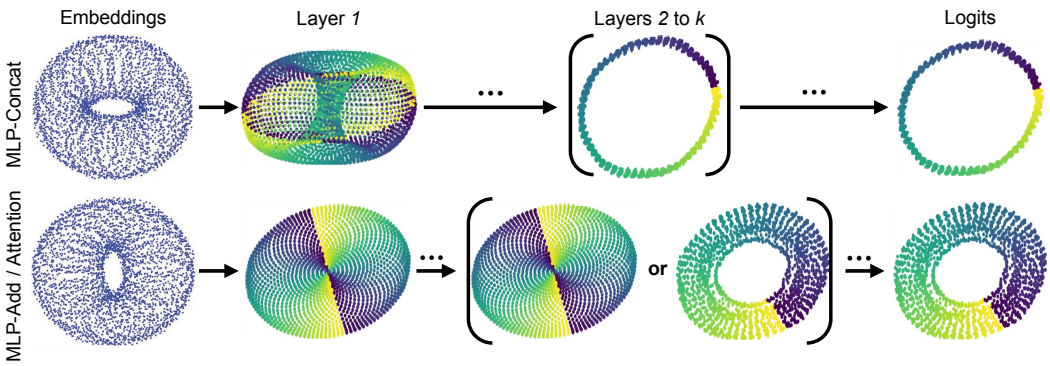

Figure 4: PCA projections of embeddings, intermediate pre-activations, and logits for MLP-Concat (top) and MLP-Add/Attention variants (bottom). In layer 1, MLP-Add and Attention models form discs like vector addition (Fig. 1), while MLP-Concat forms a torus (Fig. 3); in later layers and at logits representations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

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 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, 2025. 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.

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. [2025]. 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 ℓ 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 ℓ. Our thesis is that under the simple neuron model of (4) introduced by McCracken et al. [2025] 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 ϕ^L, ϕ^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 ϕ^L, ϕ^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 Φ_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 (Φ_i^L, Φ_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. [2025], 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^{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^{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 ϕ^L and ϕ^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. [2025]. 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., 2025]. 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 $a = b$

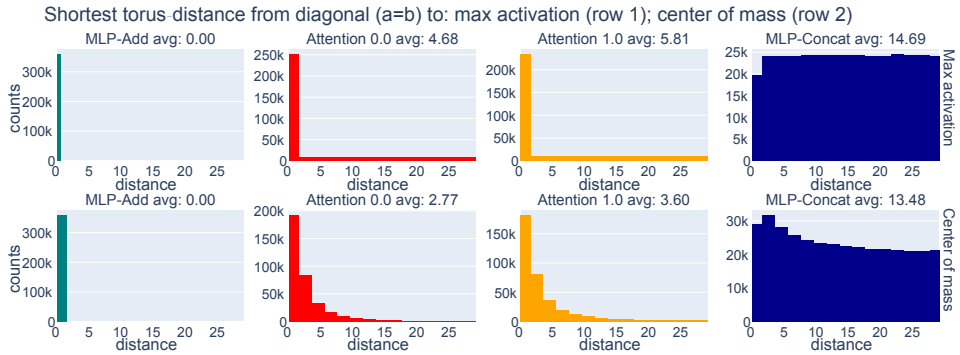

Figure 5: Histograms of torus-distance from each neuron's phase to the diagonal $a = b$, across 703 trained models. MLP-Add neurons align perfectly with the diagonal, Attention 0.0 and 1.0 show increasing off-diagonal spread, and MLP-Concat exhibits broadly distributed activations on the torus.

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.

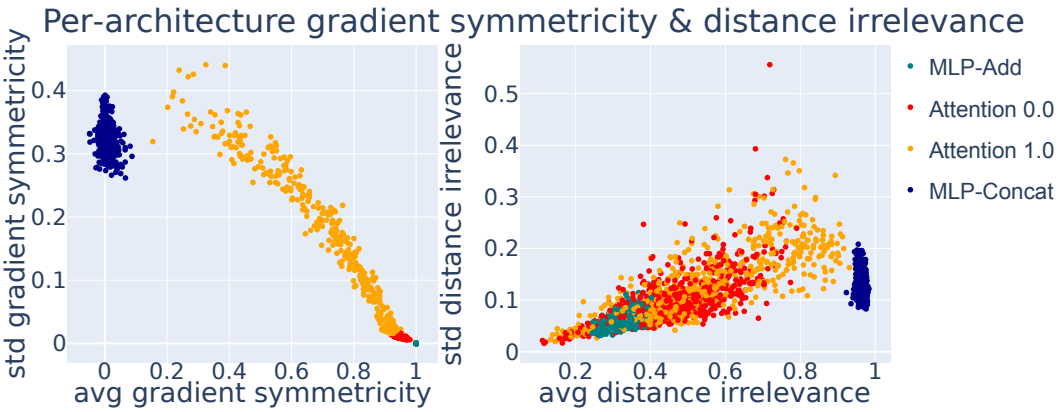

Figure 6: Evaluation of gradient symmetricity (left) and distance irrelevance (right). Each point shows the average (avg) and standard deviation (std) of one trained network. MLP-Add and MLP-Concat lie at nearly opposite extremes, while attention 0.0 and 1.0 overlap substantially. Gradient symmetricity 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

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

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.

TAG-DS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are backed by proofs in Appendix A or backed by figures over many random seeds in the main paper, with statistical significance results found in C.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We only study a singke task: modular addition. We address this in the discussion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: This can be found in A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Yes. The paper outlines the architectures tested, gives the experimental details for the reproducing the results and provides open source code that is GPU optimized: giving anyone with just one GPU the ability to reproduce results in the paper over a few days worth of compute.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [NA]

   Justification: We were unable to provide our code as a supplementary zip file since there was no field in the OpenReview submission portal to attach it. We commit to providing it in the future for a camera-ready version.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: This is introduced briefly in the paper but there are more details in Appendix B. We also provide GPU-optimized procedures for how we computed various things in E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We compute the maximum mean discrepancy (MMD) to find the distance between distributions, and statistical significance is detailed in the Appendix C and D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The experiments in this paper can reproduced with a single RTX 8000 within one day.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We highlight the positive impacts of our work: better understanding what deep networks learn in the discussion and limitations. We can not conceive any negative aspects; this is not capabilities research.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We wrote all the code ourselves. Part of the code is a translation, the clock and pizza code from Zhong et al. [2023] was read, and translated into GPU optimized Jax, which is stated in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: NA

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.