Extended Abstract Track

The Representations of Deep Neural Networks Trained on Dihedral Group Multiplication

Editors: List of editors' names

Abstract

We find coset and approximate coset circuits play a key role in how multilayer perceptrons learn dihedral group multiplication, consistent with recent findings on modular addition.

1. Introduction

Do deep neural networks (DNNs) reuse the same algorithmic primitives to learn non-commutative group multiplications? For modular addition, McCracken et al. (2025) describe network representations as *(approximate) cosets*, proving $O(\log n)$ such representations implement a divide-and-conquer algorithm that is a generalization of the Chinese Remainder Theorem. We examine DNNs trained on dihedral group multiplication, finding representations are exact cosets or approximate cosets. We study the activation-geometry of clusters of neurons, revealing manifolds aligned with (approximate) coset structure. We also study 1000 random seeds, quantitatively finding models prefer precise coset representations. These findings extend the modular-addition account to a non-commutative setting.

2. Background

The *dihedral group* D_n is the symmetries of a regular n-gon, containing $2n$ elements: n rotations r^k for $k \in \{0, \dots n-1\}$ that rotate the n-gon by $2\pi/n$ radians, and n reflections sr^k reflecting about n distinct axes. The rotation r^0 is the *identity* element, denoted e, for which $ex = xe = x$ for any $x \in D_n$. These operations form a non-commutative group multiplication when $n \geq 3$, meaning the order in which operations are multiplied matters—for instance, $sr \neq rs$. **Group multiplication**, $a \cdot b = C,\ a, b \in D_n$ involves composing two symmetries in sequence (from right to left): $r^a \cdot r^b = r^{(a+b) \bmod n}$ (rotation), $sr^a \cdot r^b = sr^{(a+b) \bmod n}$ (reflection), $r^a \cdot sr^b = sr^{(b-a) \bmod n}$ (reflection), $sr^a \cdot sr^b = r^{(b-a) \bmod n}$ (rotation).

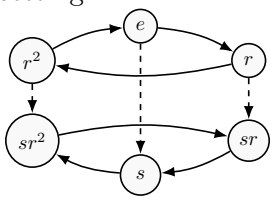

Figure 1: A D_3 Cayley Graph. Solid arrows apply left multiplication by r; dashed lines apply left multiplication by s. Non-commutativity appears $r \cdot s \neq s \cdot r$.

Cayley graphs geometrically encode a groups structure. The Cayley graph of D_n may be expressed via a *generating set* $\{r, s\}$, where nodes are group elements and (directed elements) are labeled by $\{r, s\}$. Particularly, an edge labeled $x \in \{r, s\}$ between nodes a, b exists if $b = xa$. A Cayley graph for D_3 is in Fig. 1. Note: six D_3 Cayley graphs compose D_{18}, each one corresponding to a coset. The Group Fourier Transform (GFT) generalizes the Discrete Fourier Transform and outputs a scalar on the Fourier bases of the group, see Appendix B.2.

Experimental setup. We train multilayer perceptrons (MLPs) to 100% test accuracy with 512 ReLU neurons per layer, with two different embedding matrices, one for a and one

for b, on all pairs $(a, b) \in D_{18} \times D_{18}$. D_{18} is chosen because it has more cosets compared to a prime or odd dihedral group, which allows clearer contrast of preferences for exact cosets vs. approximate cosets. Let elements 0–17 be the rotation class elements r^k and 18–35 the reflection class elements sr^k. For readability, neuron preactivation plots insert a visual break between elements 17 and 18 to distinguish the two classes. On the Cayley graph under the sign character, r^k elements lie in the $+1$ region, sr^k in the -1.

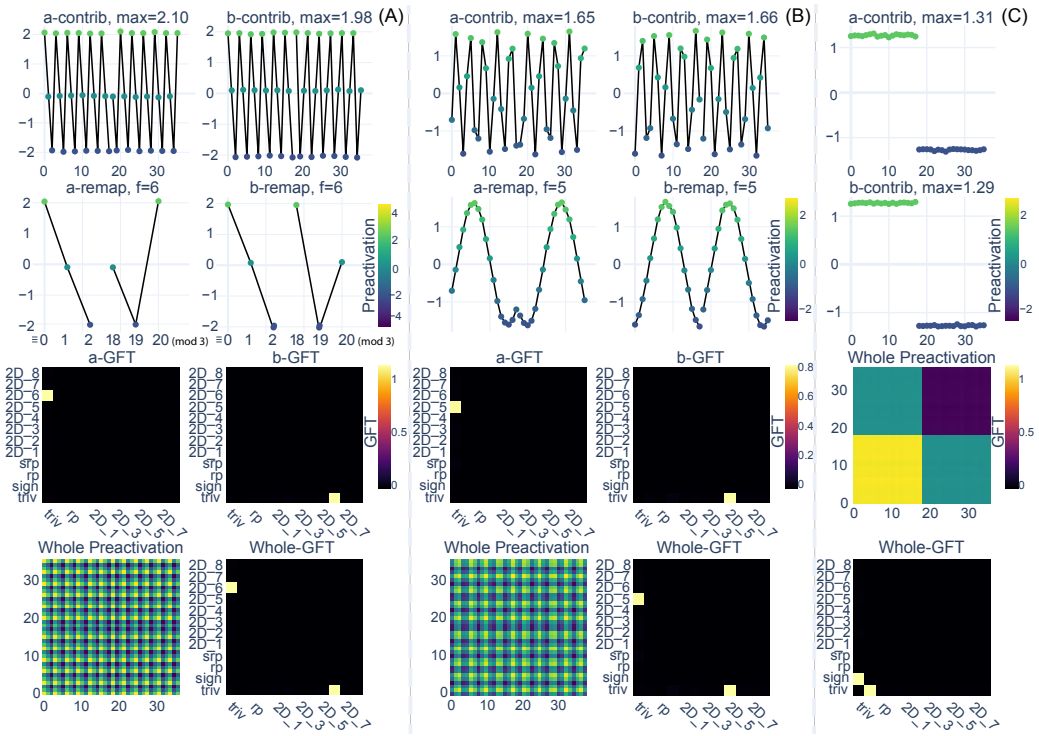

Figure 2: (A) a neuron learning frequency 6, corresponding to learning precise cosets. Remapping collapses the points in row 1 onto six cosets of D_{18} in row 2; three are in rotation class ($x < 18$): $x \equiv 0, 1, 2 \pmod 3$, and three are in reflection class ($x \geq 18$): $x \equiv 18, 19, 20 \pmod 3$, $x \in \{a, b\}$. (B) The same plots are shown for a neuron learning frequency 5, corresponding to learning approximate cosets—there are no precise equivalence classes. (C) Plots for a neuron learning the sign $+1$ coset are shown. This neuron effectively acts as an indicator for whether the answer C is in the sign $+1$ part of the graph.

3. Results

We investigate the dihedral group on 36 elements, D_{18}. With this in mind, see Figure 2, showing the preactivations of neurons and their group Fourier transforms (GFT). The preactivations are split into the contribution to the neurons' preactivation coming from just a (a-contrib), and just b (b-contrib). This split is done to emphasize that neurons activate on the cosets of a and b. The GFTs tell us which Fourier basis the preactivations concentrate on. See *e.g.* the neuron learning frequency 6 in Fig. 2 (A), concentrating on the $2D\_6$ Fourier basis. For $k \in \{0, 1, 2\}$, let coset $R_k = \{r^{3t+k}\}_{t=0}^{5}$, $S_k = \{sr^{3t+k}\}_{t=0}^{5}$. Here, for $x \in \{a, b\}$, the neuron is constant on $R_k \cup S_{\sigma_x(k)}$ with value $A_k \in \{2, 0, -2\}$, where $\sigma_a(k) = k - 1$, $\sigma_b(k) = -k \pmod 3$. After remapping to frequency 1, each coset R_k

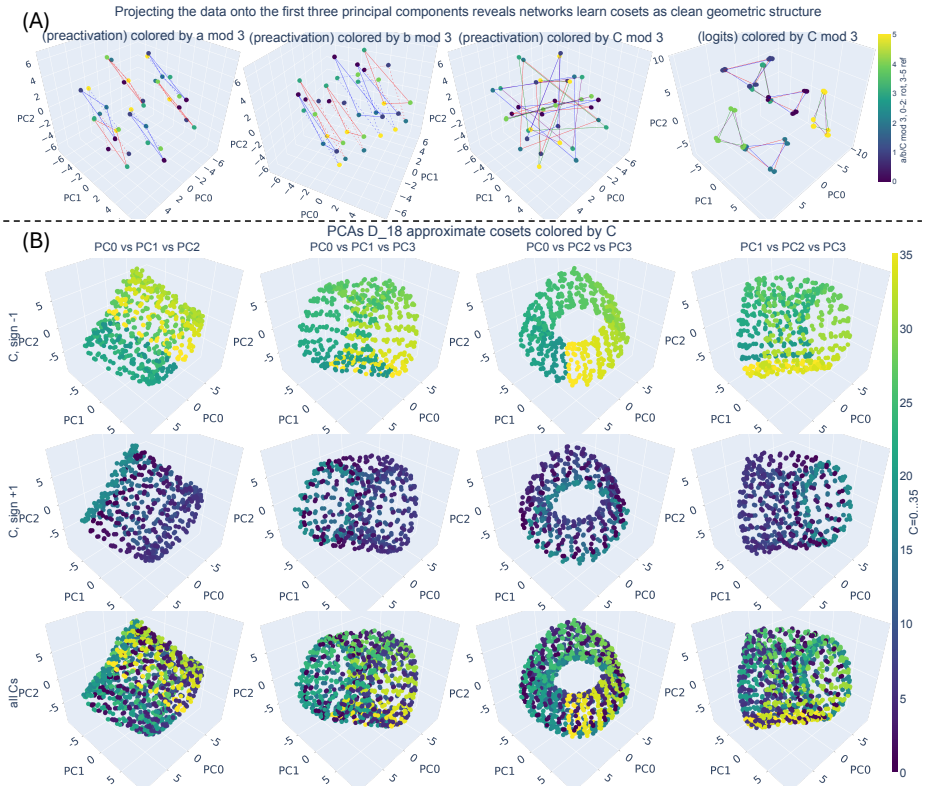

Figure 3: (A) PCAs of the preactivations of the neuron cluster with frequency 6 shows how the network has organized each of the six D_3 cosets. Coloring by a, or b shows 6 hexagrams, and coloring by C doesn't reveal clean structure until we plot a PCA of the cluster's contributions to the logits, revealing 6 cosets—3 for answers in sign +1, and 3 for answers in sign -1. (B) PCAs of an approximate coset cluster's contributions to the logits, with frequency 1, reveal that the network stores information in sign +1 ((B) row 1) orthogonally to information in sign -1 ((B) row 2); look at points of constant color in row 1 vs row 2.

and S_k collapses to a single representative. E.g., $k = 1, x = a \Rightarrow \sigma_a(1) = 0$: the neuron is constant on $R_1 \cup S_0 = \{r^1, r^4, \ldots, r^{16}\} \cup \{s, sr^3, \ldots, sr^{15}\}$; after remapping, R_1 and S_0 each collapses to a single representative. Compare this to the neuron learning frequency 5 in (B): when normalized to frequency 1 it has no collapse. This occurs because the neuron's frequency has $\gcd(5, 18) = 1$ and thus, the neuron can not divide D_{18} evenly into substructures. This is the definition of an approximate coset, which occur when frequency f has $\gcd(f, n) = 1$. Thus, it could be hypothesized that such a neuron has learned an Cayley graph representation for D_{18} involving 36 unique elements. This comes from the fact that cosets would collapse the Cayley graph to a substructure. The final column of Fig. 2 shows a neuron that learned the sign +1 coset—this neuron only activates when the sign of a and b is 1, *i.e.* a and b are both in the rotations part of the Cayley graph. This acts as an indicator for the half of the Cayley graph the answer C is in.

We cluster neurons by identifying all units that activate on the same Fourier basis with the GFT; for each neuron in the cluster we build a $2n \times 2n$ matrix whose entry (a, b) is the neuron's preactivation on datum (a, b), flatten each matrix, and stack the resulting vectors to form a $|\text{cluster } f| \times (2n)^2$ "cluster of preactivations" matrix. We then perform principal

component analysis (PCA) on this matrix of neuron preactivations and project all $(2n)^2$ data (a, b) onto the principal components (PCs). When true cosets are learned, data in the same joint equivalence class—*e.g.*, all points with ($a \equiv 0 \pmod 3$, $b \equiv 0 \pmod 3$) collapse to the same coordinate. For example, in D_{18} there are $36^2 = 1296$ points, but for neurons learning frequency 6 (since $\gcd(6, 18) = 6 \neq 1$) only 36 points are plotted (Figure 3 (A)). These 36 points correspond to the 36 joint equivalence classes. For each fixed a, there are 6 points determined by b: three with $b < 18$ (rotations) and $b \equiv 0, 1, 2 \pmod 3$ and three with $b \geq 18$ (reflections) and $b \equiv 18, 19, 20 \pmod 3$; *e.g.*, for $a \equiv 0 \pmod 3$ one may take $b \in \{0, 1, 2\}$ and $b \in \{18, 19, 20\}$ as representatives. By contrast, when frequency 1 is learned (Figure 3 (B)), $\gcd(1, 18) = 1$ and each of the 1296 data points projects to its own coordinate. Figure 3 only shows cluster contributions to the logits for frequency 1 to emphasize that networks store the answer C orthogonally, depending on the sign of the location of the answer.

Our final result is that we train 1000 neural networks and record the frequency that particular cosets are learned. We see that the network has a strong preference for exact cosets, learning D_3, $D_{1_{\text{sign}}}$, $D_{2_{\text{rp}}}$, $D_{2_{\text{srp}}}$ in most runs, and less frequently learning approximate cosets denoted $D18\_f$. This makes sense, as learning precise cosets naturally fits the group structure better, and it's likely that there's more ways for stochastic gradient descent to find them (because they're smaller graphs and thus there's more of them). This should be studied in future work investigating training dynamics.

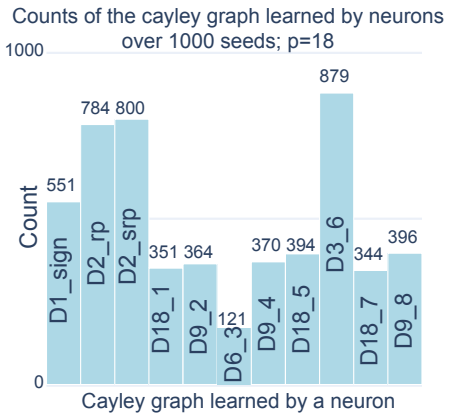

Figure 4: Histogram for how often different Cayley graphs are learned by neurons over 1000 training runs. Networks learn cosets much more frequently than approximate cosets, *e.g.* D_3, $D_{2_{srp}}$ and $D_{2_{\text{rp}}}$ all correspond to cosets.

4. Discussion

We take a step toward empirically validating McCracken et al. (2025)'s conjecture that neural networks utilize approximate coset structure when learning group multiplication. We provide preliminary evidence that approximate cosets generalize to other groups, finding them to be fundamental components of the representations learned for non-commutative group multiplication. Thus, we show that approximate cosets are neither unique to modular addition (a cyclic group) nor to commutative multiplication.

Finally, while we have identified emergent approximate coset structures in representations learned for dihedral group multiplication, this is only one piece of the puzzle. To prove McCracken et al. (2025)'s conjecture, one must demonstrate that similar phenomena occur across group operations. A promising direction is to examine representation learning on *elementary p-group multiplication*. Our primary limitation is that we did not test transformers. Secondarily, we only present results on D_{18}. Future work should address both limitations, testing transformers and dihedral groups of prime and odd composite order.

References

Bilal Chughtai, Lawrence Chan, and Neel Nanda. Neural networks learn representation theory: Reverse engineering how networks perform group operations. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023a.

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, 2023b.

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.

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

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.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis, 2024. URL https://arxiv.org/abs/2405.07987.

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

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? *arXiv preprint arXiv:1511.07543*, 2015.

Kaifeng Lyu, Jikai Jin, Zhiyuan Li, Simon S Du, Jason D Lee, and Wei Hu. Dichotomy of early and late phase implicit biases can provably induce grokking. *arXiv preprint arXiv:2311.18817*, 2023.

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.

Mohamad Amin Mohamadi, Zhiyuan Li, Lei Wu, and Danica Sutherland. Grokking modular arithmetic can be explained by margin maximization. In *NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning*, 2023. URL https://openreview.net/forum?id=QPMfCLnIqf.

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.

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.

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

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022. URL https://arxiv.org/abs/2201.02177.

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

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.

Appendix A. Related work

Group multiplication tasks have become standard benchmarks for both the mechanistic interpretability Nanda et al. (2023); Chughtai et al. (2023b,a); He et al. (2024); Tao et al. (2025); Doshi et al. (2023); Stander et al. (2024) and theoretical deep learning Gromov (2023); Morwani et al. (2024); Mohamadi et al. (2023); McCracken et al. (2025) communities. In fact, they've given both empiricists and theoreticians a common ground for proving scientific hypotheses. Notably, group multiplication plays a prominent role in validating the *Universality Hypothesis* (Li et al., 2015; Olah et al., 2020; Chughtai et al., 2023a; Huh et al., 2024), which posits that deep neural networks learning related tasks will converge to similar internal circuits. Particularly, the hypothesis implies that shared principles will underlie the representations learned by DNNs regardless of architecture, initialization, or training hyperparameters.

On the empirical side, it's the case that the viral phenomenon of grokking was first identified while training networks on modular addition, which is a group multiplication task Power et al. (2022). This led to Nanda et al.'s seminal paper Nanda et al. (2023), which provided surprisingly deep interpretations of transformer architectures to explain grokking. Subsequently, an empirical investigation into the Universality Hypothesis was conducted, generalizing the algorithm from Nanda et al. (2023) by studying cyclic and permutation group multiplications Chughtai et al. (2023a). They claimed that networks universally learned matrix representations of the group and used them to compute answers via matrix multiplication. Later, it was revealed that this wasn't the case for the permutation group: Stander et al. (2024) showed that coset circuits, not matrix multiplications, were utilized. Additionally, Zhong et al. (2023) claimed that two entirely different circuits were being learned by different transformer architectures.

Both of these counterexamples to the universality hypothesis were since resolved by one work. McCracken et al. McCracken et al. (2025) rigorously proved and showed empirically that deep neural networks were implementing a divide-and-conquer algorithm based on approximate cosets universally, ultimately learning a generalization of the Chinese Remainder Theorem. This unified all findings on modular addition with those of the permutation group—coset circuits were learned in both groups. Secondly, the result that all networks are implementing one abstract divide-and-conquer algorithm demonstrates that the differences claimed by Zhong et al. (2023) were artifacts of their analyses and metrics. Resultantly, the universality hypothesis was re-opened: all interpretations on group multiplications were unified under the idea of networks utilizing generalized (i.e., approximate) cosets.

Meanwhile, the theoretical community made breakthroughs using cyclic group multiplication as well. Gromov Gromov (2023) provided an exact analytical solution to what networks with quadratic activations learn to solve modular addition. Lyu et al. (2023) argued that smoothness was an inductive bias that could provably induce grokking. This was followed by Morwani et al. (2024), who rigorously proved $\mathcal{O}(n)$ features were required in 1-layer networks. Furthermore, Morwani et al. (2024) argued the reason sinusoidal frequencies emerged during training was due to the theory that deep neural networks seek to maximize the margin, utilizing smoothness norms in their arguments, which was simultaneously, proposed by Mohamadi et al. (2023), who argued the same. McCracken et al. came next, motivated primarily by the interpretability community's empirical claims that

	Rotations			Reflections		
$g \backslash h$	e	r	r^2	s	sr	sr^2
e	e	r	r^2	s	sr	sr^2
r	r	r^2	e	sr^2	s	sr
r^2	r^2	e	r	sr	sr^2	s
s	s	sr	sr^2	e	r	r^2
sr	sr	sr^2	s	r^2	e	r
sr^2	sr^2	s	sr	r	r^2	e

Entries are $g \cdot h$ (left multiplication).
$r^i r^j = r^{(i+j) \bmod 3}$,
$r^i (sr^j) = sr^{(j-i) \bmod 3}$
$(sr^i) r^j = sr^{(i+j) \bmod 3}$,
$(sr^i)(sr^j) = r^{(j-i) \bmod 3}$

Figure 5: Cayley table for D_3 ($= S_3$).

networks could learn algorithms. They utilized cyclic group multiplication to provide the first rigorous proof that neural networks can and do learn divide-and-conquer algorithms, proving in expectation $\Theta(\log(n))$ feature efficiency—as would be expected by a divide-and-conquer strategy—and empirically showed the bound tightly matched practical results.

Appendix B. Extra background

B.1. Dihedral group multiplication table

See Table 5.

B.2. Group Fourier Transform

Just like the classical Fourier transform decomposes a time signal into sine and cosine components, the group Fourier transform (GFT) decomposes a function on a group into "frequency-like" building blocks that reflect the group's symmetry structure. These components correspond to the group's irreducible representations and capture how the function varies across different symmetry modes—such as rotations or reflections in the case of D_n.

Applying the GFT is equivalent to projecting the function onto these natural symmetry modes. Once transformed, operations like group convolution or composition act independently on each mode, making the structure easier to interpret and analyze.

In this paper we name the GFT channels by the standard irreducible representations: triv, sign, rp, srp, and $2\mathrm{D}_k$.

triv is the *DC* mode (unchanged by all symmetries), i.e., $r \mapsto +1,\ s \mapsto +1$. sign flips under reflections but not rotations, i.e., $r \mapsto +1,\ s \mapsto -1$. For even n, rp flips under rotations while remaining mirror-even ($r \mapsto -1,\ s \mapsto +1$); srp adds a reflection flip ($r \mapsto -1,\ s \mapsto -1$). Each $2\mathrm{D}_k$ is a cosine–sine pair around the n-gon at angular frequency $2\pi k/n$: rotations rotate this pair, reflections swap/flip it. (We use $k = 1, \ldots, \lfloor (n-1)/2 \rfloor$; k and $n-k$ are equivalent.)

B.3. Approximate cosets

Approximate cosets are intuitively, the generalization of cosets to "almost a coset". They arise when neurons in a network learn to divide a group using a structure that doesn't

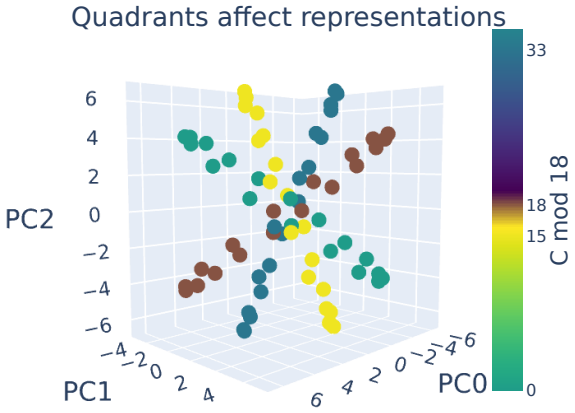

Figure 6: PCAs of the preactivations, filtered to only include data where the answer $c \in \{0, 15, 18, 33\}$.This plot shows that the sign of c results in the embedding for c being reflected. For example, consider $c = 0$ vs $c = 18$, which are the same element, but one is the mirror reflection of the other: the two corresponding circles are reflected across a plane that exists in the middle between them. The same is true for $C = 15$ vs $c = 33$.

actually divide the group. For example, D_{18} has 36 elements, 18 of which are arranged in a circle in the front, and 18 are "in the back" as a mirror reflection. Thus, it's the case that we could choose to divide the 36 elements by 6, giving us 6 sets, and since there are 18 elements in the front and 18 in the mirror reflection, we also divide 18 by $6 = 3$. This tells us that our division of the group into 6 sets, will give us smaller sets, where 3 elements are in the front and three are in the back.

Suppose alternatively that we were trying to divide D_{18} by 5. Since the $\gcd(5, 18) = 1$, 5 doesn't factorize 18 into anything smaller. Thus, there are no cosets, and resultantly, a neuron learning frequency 5 has learned the full group structure of D_{18}. Such a neuron has 5 peaks (maximum values), and if a peak is located at a, the next peaks are located $a \pm \frac{18}{5} = 3.6$. Naturally, because the problem is discrete, this results in every point having a different activation value and can be seen in Figure **??** in the first panel, and contrasted with Figure 2's first panel that has every point at one of three levels.

We offer the reader the following intuition: approximate cosets are simply when a neuron has learned something that doesn't allow it to cleanly divide the Cayley graph into smaller pieces. A natural response is to think "perhaps neurons would prefer to learn things that cleanly divide the group?" and indeed, this is observed later in the main extended abstract in Figure 4. For the mathematical definition, please see Section 3 and 4 in McCracken et al. (2025).

Appendix C. More Figures

We have some other neurons here, with the goal being to show how different Fourier basis correspond to different cosets. These are displayed in Figures 7, 8, 9, 10. Notably, Figs. 9 and 10 learn much simpler structure—they effectively partition the group in different quarters.

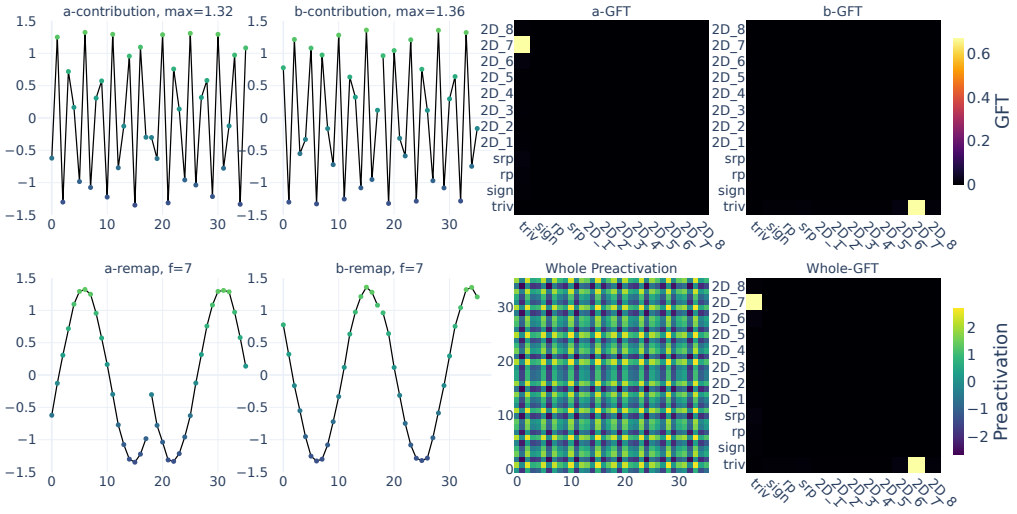

Figure 7: GFT reveals neuron pre-activations concentrated in the $2D_7$ representation. The neuron learns frequency 7, corresponding to learning approximate cosets.

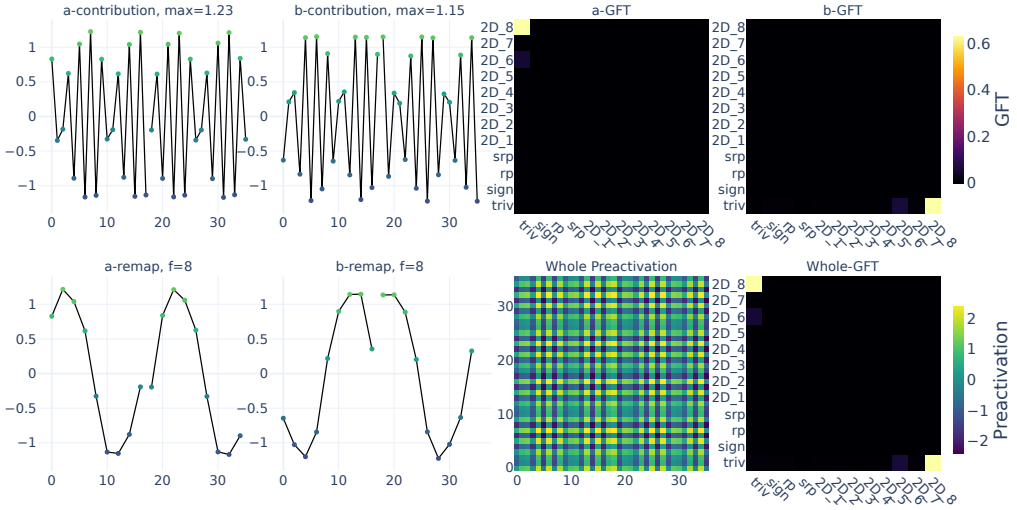

Figure 8: As in Fig. 7, but for $2D_8$. GFT reveals neuron pre-activations concentrated in the $2D_8$ representation. The neuron learns frequency 8, corresponding to learning precise cosets. Remapping collapses the points in row 1's first two columns onto eighteen cosets of D_{18} in row 2's first two columns.

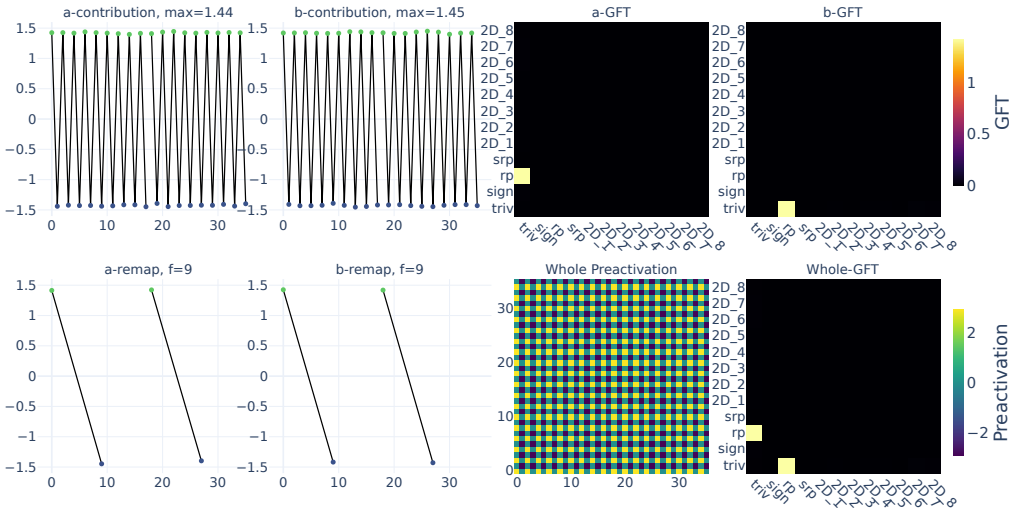

Figure 9: GFT reveals neuron pre-activations concentrated in the 1D *rp* representation, corresponding to learning precise cosets.

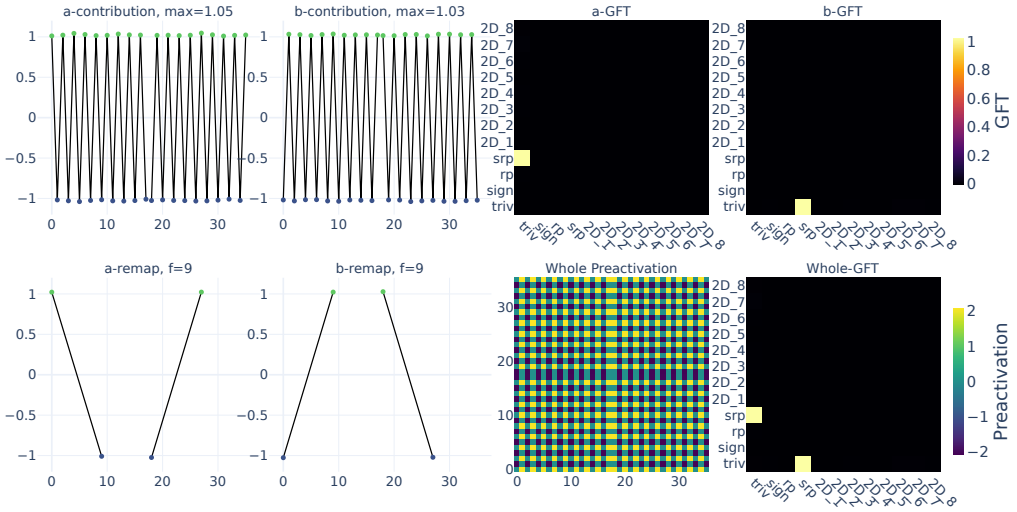

Figure 10: GFT reveals neuron pre-activations concentrated in the 1D *srp* representation, corresponding to learning precise cosets.

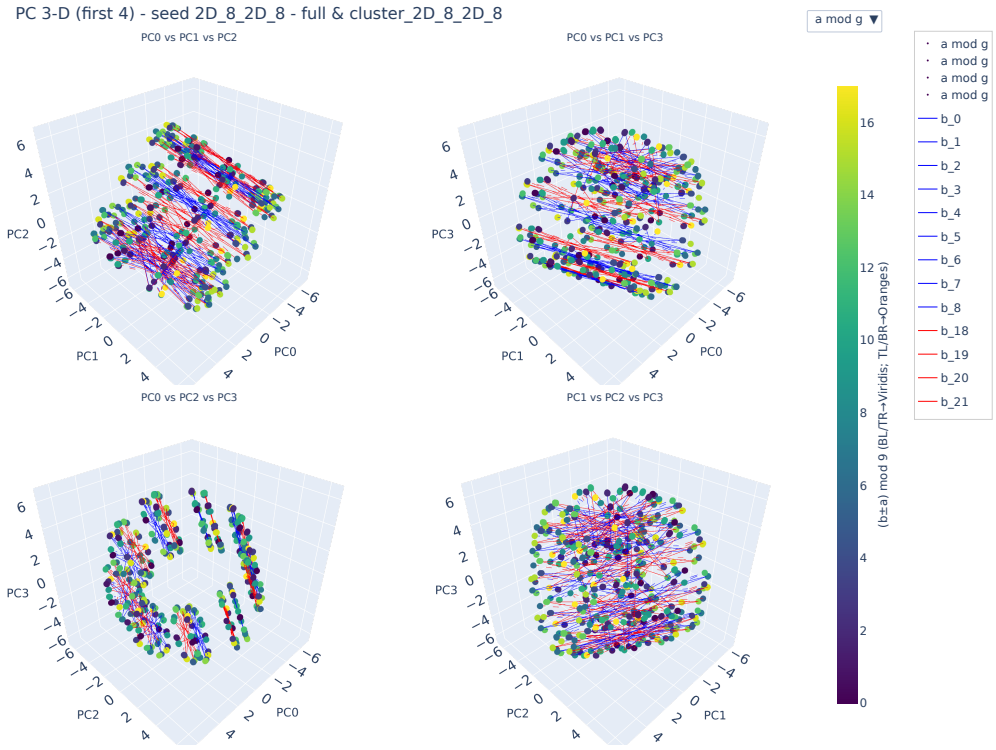

Figure 11: PCAs of preactivations under $2D_8$ shows how the network has organized each of the eighteen D_9 cosets. Here $g = 9$.

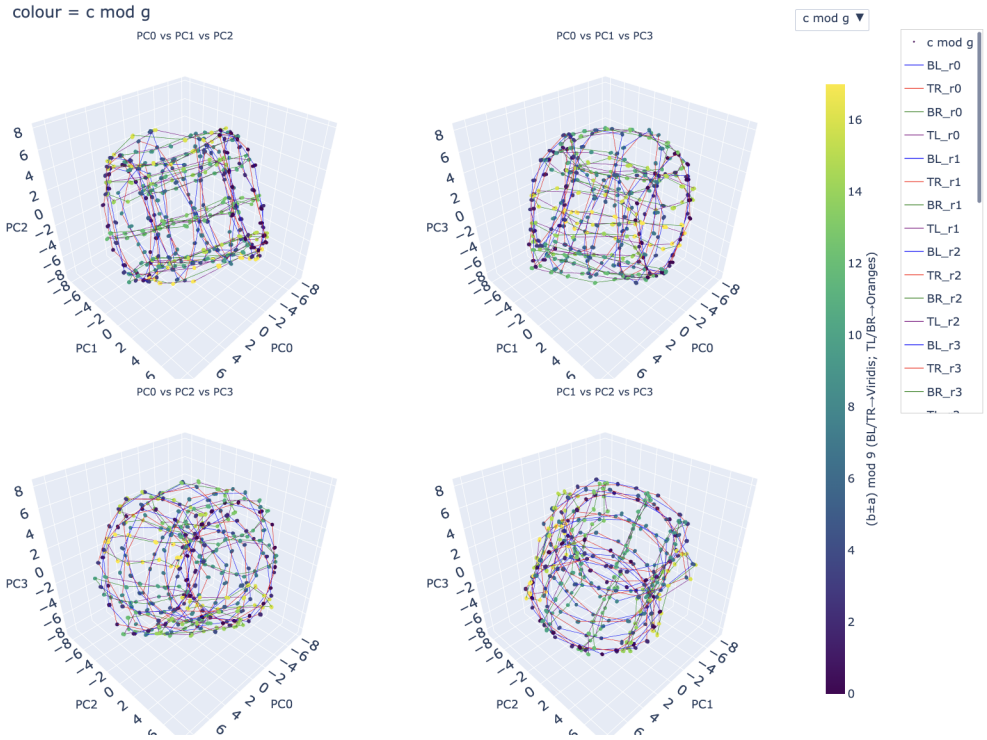

Figure 12: PCAs of a coset cluster's contributions to the logits, with frequency 8, reveal that the network stores information in the rotation class orthogonally to information in the reflection class. Here $g = 9$.

Appendix D. How were the plots made

We use an 80%, 20% train, test split, and train with Adam Kingma and Ba (2014), cross-entropy loss, with learning rate 0.001 and batch size 36 for 5000 epochs.

The remapping definition that is used to normalize sinusoidal functions with frequency f so that they can be plotted with frequency 1 is provided below.

Definition 1 (Step size) $d := (\frac{f}{\gcd(f,n)})^{-1}(\mathrm{mod}\frac{n}{\gcd(f,n)})$, *where the modular inverse is used.*

Definition 2 (Remapping: frequency normalization) *Consider the function* $h(x) = \cos(2\pi fx/n)$ *with frequency* f*. We define a new function* g*, allowing us to perform something analogous to a change of variables using the step size* d*:* $g(d \cdot x) = h(x) \iff g(x) = h(d^{-1} \cdot x)$*.*

Extended Abstract Track