PRIME BY PRIME: A REFERENCE FOR EMERGENT MODULARITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks can break a task into subtasks and learn to solve each independently, in a different part of the model. But what determines whether this modular organization emerges during learning remains poorly understood. One obstacle is that the relevant subtasks and their internal representations are often unknown. We address this by studying transformers trained to compute $(a + b) \bmod n$ and $(a \times b) \bmod n$, where $n = p_1 \cdots p_k$ is a product of distinct primes. These tasks admit a simple decomposition: represent the inputs through their remainders $a \bmod p_i$ and $b \bmod p_i$, perform arithmetic separately for each prime, and reconstruct the unique answer modulo n consistent with the results, as guaranteed by the Chinese remainder theorem. We ask whether transformers discover this decomposition through training. We find that models trained jointly on addition and multiplication (i.e., learning the ring $\mathbf{Z}_n$) consistently do so, with evidence at three levels: (i) errors occur approximately independently across primes, (ii) each prime's computation occupies its own representational subspace, and (iii) these computations largely rely on distinct groups of neurons. This modular organization critically depends on the training task: addition alone produces entangled or partially modular solutions that vary across seeds, including fused subtasks and asymmetric dependencies between primes. Why training on the ring induces this modular solution, while addition alone does not, remains an open theoretical question and an interesting direction for future work.

1 INTRODUCTION

A neural network is said to be *modular* over a set of subtasks if each subtask is computed independently within the model. Modularity is thought to aid compositional generalization and reduce interference between tasks (Csordás et al., 2021; Mittal et al., 2022). It also benefits model interpretation and safety interventions: when capabilities are separate (Räuker et al., 2023), it is easier to locate their source, and to unlearn one capability without collateral damage to others (Wybitul et al., 2026; Nguyen et al., 2024).

Despite modularity being a well-documented property of biological neural networks (Wagner et al., 2007; Clune et al., 2013), artificial neural networks are more frequently entangled than modular (Elhage et al., 2022; Liu et al., 2024). However, when and why model capabilities are modular or entangled is poorly understood, in part because most real-world tasks have unclear subtask decompositions, which makes finding where each subtask lives in the model difficult.

We study transformers trained to compute $(a + b) \bmod n$ and $(a \times b) \bmod n$, where $n = p_1 \cdots p_k$ is a product of distinct primes. The Chinese remainder theorem (CRT) gives a simple subtask decomposition: compute $(a + b) \bmod p_i$ and $(a \times b) \bmod p_i$ for each prime p_i independently, and reconstruct the final answer from these results.

This setting avoids two main difficulties in testing for modularity. The CRT gives the subtask decomposition, and because models use Fourier features to compute modular arithmetic (Nanda et al., 2023; Zhong et al., 2023), we know where per-prime computations would occur in representation space if they exist. This lets us test for per-prime modules without searching for them.

We find that models trained jointly on addition and multiplication (i.e., learning the ring $\mathbf{Z}_n$) consistently use this factorization, with evidence at three levels:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- **Behavioural Modularity:** errors occur approximately independently across primes.
- **Representational Modularity:** each prime's computation largely occupies its own subspace.
- **Neural Modularity:** each prime uses largely distinct sets of neurons.

In contrast, models trained on addition alone range from fully entangled to partially modular depending on the seed.

1.1 Contributions

We make the following contributions:

- We show that models trained jointly on addition and multiplication $\mod n$ implement the Chinese remainder theorem, computing arithmetic modulo each prime divisor of n independently.
- We introduce definitions of modularity at three levels with tests for each, and show that models implementing the CRT satisfy each of them.
- We show that there exists a large range of structures between modular and entangled models, which include fused subtasks, and subtasks which depend on each other asymmetrically.

2 Related works

Modularity Modularity in neural networks has been studied through neuron clustering (Filan et al., 2021), structural compositionality of subtasks (Lepori et al., 2023), and probing and ablating architecturally separate modules (Béna & Goodman, 2025). Csordás et al. (2021) learn weight masks to test if functions like multiplication and addition use separate weights. Mittal et al. (2022) discuss the benefits of modular neural networks, and the difficulties in testing for modularity due to a lack of evaluation metrics and small modular models. Since modularity rarely emerges on its own, several works propose methods to encourage it (Liu et al., 2024; 2023). These works fix modules architecturally or must discover subtasks. In our setting, subtasks and their representational subspaces are known in advance, subtasks are split by prime rather than by operation, and modularity emerges consistently without incentives.

Modular addition and groups Modular addition is one of the most studied tasks in mechanistic interpretability, much of it focused on grokking (Power et al., 2022; Nanda et al., 2023; Gromov, 2023; Zhong et al., 2023). The Fourier "clock" algorithm described in these papers is the basis of our representational analysis. This work has been extended to other groups (Stander et al., 2024; Chughtai et al., 2023). Higgins et al. (2018) relate entanglement to group theory, focusing on equivariance and world representations rather than modularity.

Multiplication and the CRT McCracken et al. (2025) describe an *approximate* Chinese remainder theorem for modular addition, but find no consistent and complete per-prime decomposition, consistent with our addition models. Hwang et al. (2026) note that standard embeddings do not naturally expose per-prime structure when trained on modular addition, and propose embeddings that do. Multiplication and rings are less studied. Nguyen (2026) looks at multiplication and how it creates a clock under the discrete log. Chen et al. (2026) expand the algorithm in Chughtai et al. (2023) to multiplicative monoids (multiplication $\mod n$), but do not see a CRT decomposition, which we find only occurs consistently in rings. Xu (2026) studies grokking on models learning addition and multiplication, but asks which *operation* groks first, whereas we focus on whether each prime is learned independently.

3 Behavioural modularity

3.1 Definitions

The first and simplest notion of modularity is *behavioural modularity*.

Let X be a set of model inputs, $\mathcal{C}$ class labels assigned to X, and $f : X \to \mathcal{C}$ a model trained to classify X. Suppose that $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ are label sets of X such that it suffices to know all the labels $\mathcal{C}_i$ to classify X in $\mathcal{C}$. We may refer to the $\mathcal{C}_i$ as subtasks.

Let A_i be the event that f's label in $\mathcal{C}_i$ is correct. The probability of A_i is the accuracy of f on $\mathcal{C}_i$.

For example, we will be looking at arithmetic $\mod 1001$, which factors as $1001 = 7 \cdot 11 \cdot 13$. Our class labels $\mathcal{C}$ are the output classes $\mod 1001$, and our subclass labels are the output classes $\mod 7$, 11, and 13. Then A_7 would be the event that the model is correct $\mod 7$ on an input, so $f(a, b) \equiv (a + b) \mod 7$. Importantly, it may be the case that the model is *incorrect* $\mod 1001$, but *correct* $\mod 7$.

Definition 3.1 (Behavioural modularity, during training)**.** f is said to be behaviourally modular over the subtasks $\mathcal{C}_1, \ldots, \mathcal{C}_k$ if the events A_i are mutually independent.

If the model has perfect accuracy everywhere, then each A_i has probability 1, so they are automatically independent. Because of this, we will test something stronger post-training:

Definition 3.2 (Behavioural modularity, post-training)**.** f is said to be behaviourally modular post-training over the subtasks $\mathcal{C}_1, \ldots, \mathcal{C}_k$ if its predictive distribution (obtained from the softmax probabilities) factorizes across them: $p(c|x) = \prod_i p_i(c_i|x)$ for some distributions p_i over $\mathcal{C}_i$, and each $x \in X$.

If Definition 3.2 holds, then across the training data, the model's *second* most likely output (from looking at its softmax probabilities) should differ from its top prediction by only one subtask. This we can test easily.

3.2 Experimental setup

We use the following setup throughout the paper, unless stated otherwise.

Dataset Our dataset consists of inputs of the form "$a + b =$ " and "$a \times b =$ ", with $a, b \in \{0, 1, \ldots, n-1\}$, the correct answers being the solution $\mod n$, with $n = 1001$. The addition only model is trained exclusively on addition examples. We use a training split of 50%.

Model We train a one-layer pre-LayerNorm transformer (Xiong et al., 2020; Vaswani et al., 2017) ($d_{\text{model}} = 256$, 4 heads of dimension 256, GELU MLP (Hendrycks & Gimpel, 2016) of width 1024, untied unembedding), no dropout. We use AdamW (Loshchilov & Hutter, 2019) (learning rate 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.98$, weight decay 0.1, batch size 512). Models are trained to at least 80k steps, with every model reaching at least 99.7% test accuracy.

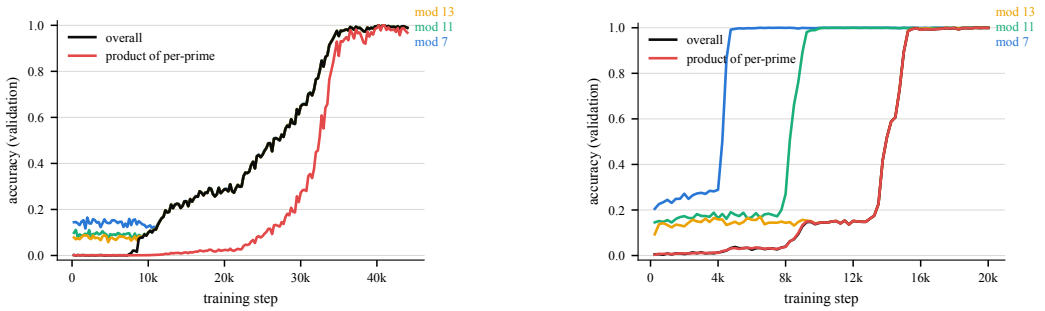
Most work on modular arithmetic studies grokking, which needs higher weight decay (≥ 0.5). We use a lower value to keep the setup more typical.

3.3 Results

Figure 1 shows test accuracy for the addition as well as addition and multiplication (ring) models during training, as well as the per-prime accuracies, and the product of the per-prime accuracies. In the ring, we see that each prime is acquired at distinct steps, and the product of the per-prime accuracies closely follows the overall accuracy, evidence for independence as in Definition 3.1. Because there are three random variables, we also need to confirm that each pair is independent, which we do in Table 8. In contrast, in addition we see that the per-prime accuracies and the overall accuracy are equal throughout training, so the model is not behaviourally modular.

In Table 1 we can confirm post-training modularity, as in Definition 3.2. We see that in the ring, in almost every single one of the 20,000 examples seen, the second likeliest answer differs in exactly one prime.

3.4 Discussion



(a) Addition,

(b) Addition and multiplication (ring)

Figure 1: **Behavioural modularity during training.** Model jointly trained on addition and multiplication shows modularity as in Definition 3.1, while model trained on addition alone does not.

Table 1: **Next likeliest answers.** On the same model, this reports the number of primes which differ between the model's top answer, and its second highest answer in its softmax probabilities. This confirms post-training behavioural modularity, as in Definition 3.2, in the ring.

Model	mean primes off	off by 1	off by 2	off by 3
Addition	2.875	0.0%	12.51%	87.49%
Ring (both operations)	1.001	99.95%	0.04%	0.01%
Random wrong answer	2.692	2.8%	25.2%	72.0%

The ring exhibits behavioural modularity, while addition alone does not. One surprise in Figure 1 is that the per-prime accuracies are all equal during training. Intuitively one would assume that because the chance level accuracy $\mod 7$ is higher than $\mod 13$, its accuracy would stay higher throughout. This is due to how the model constructs the "clock" algorithm for modular addition, as in Nanda et al. (2023). The difference between two numbers that appear beside each other on the clock must share no divisors with n (this is because any generator of the integers $\mod n$ cannot share divisors with n). This means that they differ modulo every prime dividing n. Once the model forms the clock, here around step 8,000, the likeliest errors for the model to make will be beside the correct answer on the clock, which is then an error in each prime. In the ring, each prime is computed independently, so errors do not correlate.

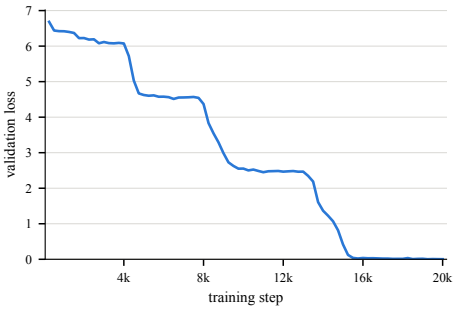


Figure 2: Validation loss of the ring, which falls in one step per prime.

Looking at the ring in Figure 1, each prime being acquired successively creates a clean example of the quantization model of neural scaling (Michaud et al., 2023), which posits that model capabilities are learned in discrete steps. This phenomenon can also be seen in the model's loss curve, which resembles a staircase, a phenomenon also indicative of saddle-to-saddle dynamics (Jacot et al., 2022; Pesme & Flammarion, 2023). Whenever the prime p is acquired, the cross entropy loss drops from $\log(n)$ to $\log(n/p) = \log(n) - \log(p)$, so each new prime drops the loss by $\approx \log(p)$. We also note that primes are acquired in order $7 \to 11 \to 13$, which we find to hold across models we test.

4 Representational Modularity

For many tasks, the most interpretable computations occur in representation space. Therefore, it makes sense to introduce representational modularity, which can be studied on the readout, or some layer of the residual stream.

4.1 FORMAL DEFINITION

Definition 4.1 (Representational modularity)**.** Fix a representation map $h : X \to \mathbb{R}^d$ (e.g., residual stream at a given layer, or the readout). Let $\varepsilon, \delta > 0$. Given subtask decomposition $\mathcal{C}_1, \ldots, \mathcal{C}_k$, f is (ε, δ)-representationally modular at h if there exist subspaces $V_1, V_2, \ldots, V_k$ such that, for each i,

- Projecting h onto $V_i^{\perp}$ reduces the accuracy of $\mathcal{C}_i$ to within δ of chance, and keeps the accuracy of each other task to within ε of baseline.
- Projecting h onto V_i keeps the accuracy of $\mathcal{C}_i$ to within ε of baseline, and reduces the accuracy of each other task to within δ of chance.

In other words, each task approximately occupies its own subspace.

4.2 RESULTS

We use the same training setup described in Subsection 3.2. We will look at the residual stream, then the readout.

Residual stream We look at 3000 sample inputs, take class means for each of the $\mod p$ classes, and centre them. The span of the centred class means form a $(p-1)$-dimensional vector subspace, U_p. In Figure 3, we see that these subspaces have significant clustering for joint addition and multiplication, but not for addition alone.

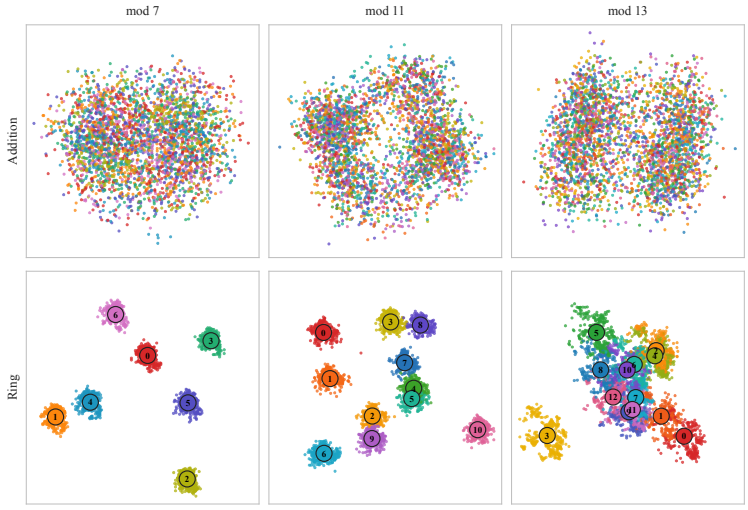


Figure 3: **Residual stream subspaces.** This shows the residual streams of 3,000 inputs projected onto the per-prime subspaces U_p, then onto their top two principal components. Both use the same seed, the inputs of the ring are only addition examples for consistency. This shows significant clustering in the ring, and none in addition. This can be seen as a per-prime neural collapse (Papyan et al., 2020).

We can then test Definition 4.1 on these subspaces:

- We "kill" the prime p by projecting the residual stream onto $U_p^{\perp}$ and calculating the per-prime accuracies.
- We "keep" the prime p by projecting the residual stream onto U_p and calculating the per-prime accuracies.

In Table 2 we see that "killing" p drops its accuracy to chance while sparing the others, and "keeping" p keeps its accuracy high, while the others fall to chance, confirming Definition 4.1.

Readout Models compute modular arithmetic with Fourier features (Nanda et al., 2023), so we know in advance which directions in logit space carry information about each prime. These are the $\bmod n$ Fourier basis vectors, $\psi_k(c) = e^{2\pi ikc/n}$, where $0 \le k \le n-1$ is the frequency, and $0 \le c \le n-1$ the output $\bmod n$.

If p is a factor of n, and $k = 0 \bmod p$, then $\psi_k(c)$ is only a function of $c \bmod n/p$. In other words, ψ_k gives you no information $\bmod p$.

- We "kill" the prime p by projecting the readout onto the Fourier features ψ_k which carry no information $\bmod p$: the frequencies $k = 0 \bmod p$.
- We "keep" the prime p by projecting the readout onto Fourier features ψ_k which *do* carry information $\bmod p$: the frequencies $k \neq 0 \bmod p$.

Specifically, we expand the unembedding W in the Fourier basis as $W = \Phi\hat{W}$, where Φ is the matrix whose columns are the Fourier basis vectors, and we zero rows of $\hat{W}$ corresponding to the frequencies we want to ablate, obtaining $\hat{W}'$ and a new unembedding matrix $W' = \Phi\hat{W}'$. We then compute the logits via $W'h + b$, with the bias b left unchanged.

In Table 2, we see that, again, killing p drops its accuracy to chance while sparing the others, and keeping p keeps its accuracy high, while the others fall to chance. In Table 3, we test this on $143 = 11 \cdot 13$, $221 = 13 \cdot 17$, and $595 = 5 \cdot 7 \cdot 17$ as well, confirming Definition 4.1 in each case.

Table 2: **Subspace ablations.** Ablate the per-prime subspaces in the residual stream and readout. Per-prime accuracies are scored by the model's argmax, reduced $\bmod p$. The table numbers are then the mean taken over 20 seeds. The ring satisfies Definition 4.1 with $\varepsilon \approx 0.03$ and $\delta \approx 0.10$ on the residual, and $\varepsilon \approx 0$ and $\delta \approx 0.05$ on the readout. For multiplication, 0 is an answer more often than other numbers, so multiplication can have a slightly higher chance accuracy.

			kill p			keep p		
Subspace	Op.	p	mod 7	mod 11	mod 13	mod 7	mod 11	mod 13
Readout blocks	$+$	7	**0.14**	1.00	1.00	**1.00**	0.14	0.12
		11	1.00	**0.09**	1.00	0.18	**1.00**	0.10
		13	1.00	1.00	**0.08**	0.18	0.12	**1.00**
	$\times$	7	**0.15**	1.00	1.00	**1.00**	0.20	0.17
		11	1.00	**0.09**	1.00	0.26	**1.00**	0.13
		13	1.00	1.00	**0.08**	0.24	0.15	**1.00**
Residual U_p	$+$	7	**0.15**	1.00	1.00	**1.00**	0.11	0.10
		11	1.00	**0.10**	1.00	0.21	**1.00**	0.10
		13	1.00	1.00	**0.08**	0.24	0.12	**1.00**
	$\times$	7	**0.24**	1.00	1.00	**1.00**	0.13	0.11
		11	1.00	**0.16**	1.00	0.27	**0.98**	0.12
		13	1.00	1.00	**0.13**	0.28	0.14	**0.97**

Table 3: **Readout ablations in other moduli.** Per-prime accuracies, mean values over 20 seeds each, $+$ operation. *Kill diag.*: accuracy modulo p after killing p, listed for each prime in order. *Keep off-diag.*: accuracy on the other primes after keeping only p. Chance is $1/p$.

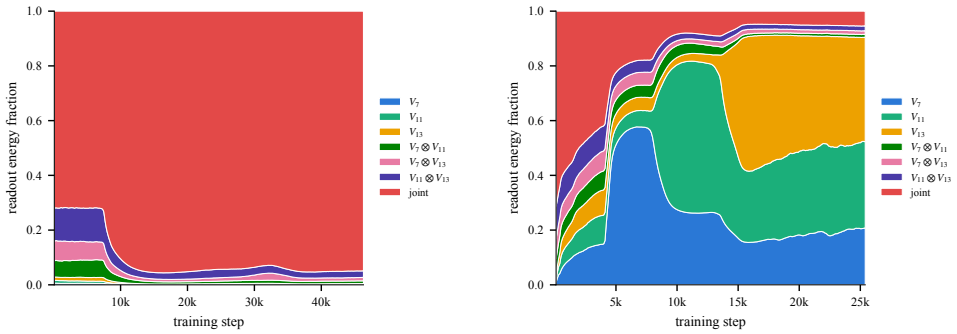
n	kill diag.	keep off-diag.
$143 = 11 \cdot 13$	0.09, 0.07	0.17, 0.19
$221 = 13 \cdot 17$	0.07, 0.06	0.13, 0.16
$595 = 5 \cdot 7 \cdot 17$	0.20, 0.14, 0.06	0.06–0.21

4.3 Discussion

Which frequencies the model learns influences how modular the model is. In our $\bmod 1001$ model, let V_7 be the subspace of the readout spanned by frequencies ψ_k which only depend on $\bmod 7$ (so

that $k = 0 \bmod 143$ because $143 = 11 \cdot 13$), and same for V_{11}, and V_{13}. Similarly, let $V_7 \otimes V_{11}$ be spanned by frequencies which depend on 7 and 11, but not 13, and so on. And last, let $V_7 \otimes V_{11} \otimes V_{13}$ be spanned by the frequencies carrying information in every prime.

The *energy* of each subspace is the fraction of the unembedding's squared norm it holds, computed from the DFT over the output classes. This tells us how much of each subspace the model is learning. Comparing Figure 4 and Figure 1, each prime is learned exactly when the model puts energy into a subspace carrying it.



(a) Addition (b) Ring

Figure 4: **Subspace decomposition of unembedding matrix** The ring uses pure-prime subspaces, addition uses the joint subspace. Here "joint" means $V_7 \otimes V_{11} \otimes V_{13}$, and "mixed" the subspaces $V_7 \otimes V_{11}$, $V_7 \otimes V_{13}$, and $V_{11} \otimes V_{13}$.

So when the model is modular it learns the "pure" per-prime frequencies, and when the model is fully entangled it learns the fully joint frequencies.

5 Neural Modularity

5.1 Definitions

Our definition of neural modularity is similar to representational modularity, with subspaces replaced by neuron sets.

Definition 5.1. Let $\varepsilon, \delta > 0$. Given subtask decomposition $\mathcal{C}_1, \ldots, \mathcal{C}_k$, f is said to be (ε, δ)-neurally modular if there exist neuron sets $\mathcal{N}_1, \ldots, \mathcal{N}_k$ such that

- Setting every neuron $j \in \mathcal{N}_i$ to 0 reduces the accuracy of $\mathcal{C}_i$ to within δ of chance, and keeps the accuracy of each other task within ε of baseline.
- Setting every neuron $j \notin \mathcal{N}_i$ to 0 keeps the accuracy of $\mathcal{C}_i$ within ε of baseline, and reduces the accuracy of each other task to within δ of chance.

In other words, each task approximately uses its own set of neurons.

A natural question following this definition is how to assign a subtask to each neuron. First, consider a single post-activation MLP neuron. We want to assign the neuron a score for each of the subtasks $\mathcal{C}_i$ indicating how much it distinguishes between the classes in $\mathcal{C}_i$. For each input from the input set $x \in X$, let $z(x)$ be the neuron's activation, and $\overline{z}$ its mean activation. For each $c_i \in \mathcal{C}_i$, let μ_{c_i} be the mean activation over all instances of that class, and write $c_i(x)$ for the label of x in $\mathcal{C}_i$. Then

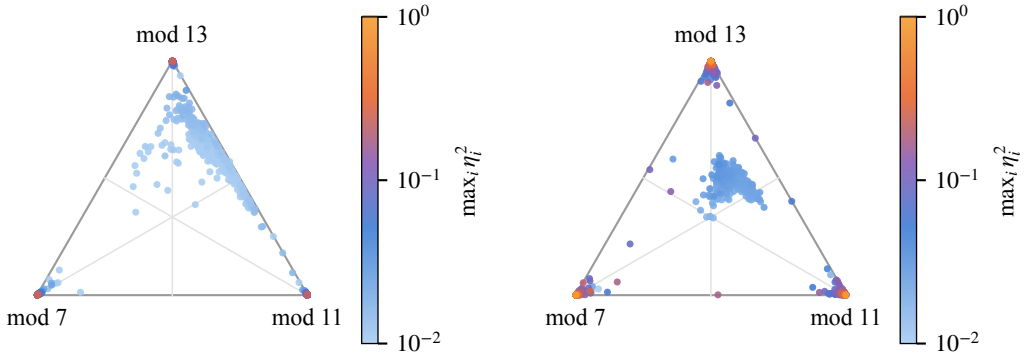
$$\eta_i^2 = \frac{\sum_x (\mu_{c_i(x)} - \overline{z})^2}{\sum_x (z(x) - \overline{z})^2}$$

is the between-class variance over the total variance. For a specific neuron j, we may write $\eta_i^2(j)$. We then assign each neuron to its most selective subtask: $j \in \mathcal{N}_i$ if $\eta_i^2(j) = \max_{i'} \eta_{i'}^2(j)$.

We can also give each neuron a *selectivity share* $s_i = \eta_i^2 / \sum_{i'} \eta_{i'}^2$, the proportion of its label-explained variance attributable to $\mathcal{C}_i$.

5.2 Results

When looking at addition in the ring, the η_i^2 scores are generally low, the highest is 0.213, but the selectivity shares still distinguish the primes clearly. We plot selectivity and the max η_i^2 scores in Figure 5.



(a) Ring ($+$) (b) Ring ($\times$)

Figure 5: **Neural modularity.** This shows the per-neuron selectivity scores s_i, and colour indicates the highest η^2 score attained by that neuron. We calculate η^2 scores using $200{,}000$ training examples. Corner means selective for one prime. The colour indicates that neuron's highest η_i^2 score. For addition, $689/1024$ neurons had a selectivity score > 0.8, for multiplication it was $676/1024$. The max η_i^2 score for each operation was 0.213 for addition and 0.937 for multiplication. Neurons, for addition in particular, have a slight bias towards larger primes.

Table 4 ablates these sets. The addition operation shows perfect modularity. Under $\times$ the scores are slightly worse. This is unlike behavioural and representational modularity, so neural modularity is slightly weaker.

Table 4: **Neuron set ablations**. Per-prime accuracies for $n = 1001$, mean over 20 seeds. Under $\times$, killing $\mathcal{N}_p$ leaves $\mod p$ slightly above the best information-free accuracy (in this case, these are 0.27, 0.17, and 0.15). The ring satisfies Definition 5.1 with $\varepsilon \approx 0$, and $\delta \approx 0.1$.

		kill $\mathcal{N}_p$			keep $\mathcal{N}_p$		
Op.	p	mod 7	mod 11	mod 13	mod 7	mod 11	mod 13
$+$	7	**0.13**	1.00	1.00	**1.00**	0.10	0.09
	11	1.00	**0.10**	1.00	0.13	**1.00**	0.09
	13	1.00	1.00	**0.09**	0.13	0.10	**1.00**
$\times$	7	**0.37**	1.00	1.00	**1.00**	0.22	0.18
	11	1.00	**0.23**	1.00	0.36	**1.00**	0.19
	13	1.00	1.00	**0.19**	0.36	0.22	**1.00**

6 Partial Modularity

So far, models either compute subtasks independently or jointly. Models trained on addition show two further relationships between subtasks:

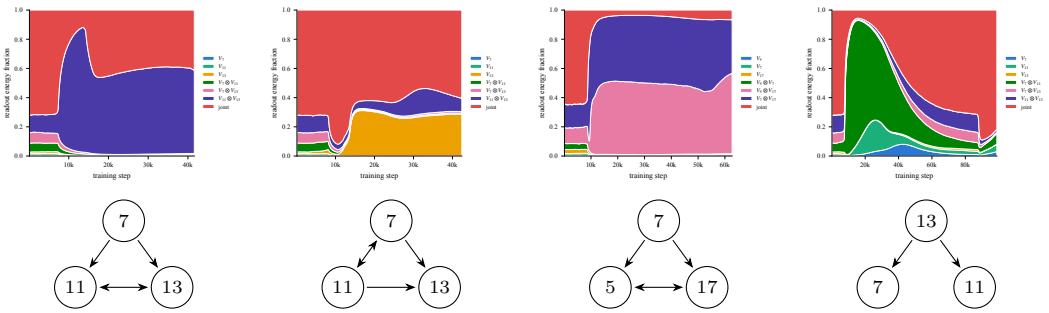
- **Asymmetry**: ablating $\mathcal{B}$ destroys $\mathcal{A}$, but ablating $\mathcal{A}$ leaves $\mathcal{B}$ intact.

- **Fusion**: two subtasks depend on each other, so the model may be modular only at a coarser decomposition.

Figure 4 showed that the ring puts its readout energy into the pure subspaces, while addition models mostly use the joint subspace $V_7 \otimes V_{11} \otimes V_{13}$. Some addition models use mixed subspaces as well. Figure 6 shows four seeds, including one for $n = 595$, which exhibit partial modularity. In each case, the model uses subspaces other than the pure and joint ones. Subspace ablations for $n = 1001$, seed 10 in Figure 6 are in Table 5. The model uses $V_{11} \otimes V_{13}$ together with the joint subspace. Keeping the joint subspace alone raises no prime above chance, and keeping $V_{11} \otimes V_{13}$ alone raises 11 and 13 but leaves 7 at chance. Keeping both recovers everything. So 11 and 13 are fused, and 7 depends on both subspaces, even though $V_{11} \otimes V_{13}$ carries no information $\mod 7$. We can visualize this, and the dependencies for other models, in a subtask dependency graph in Figure 6. Ablation tables for the other seeds are in Appendix C.

Table 5: Subspace ablations for addition, seed 10. Bold: intact (> 0.8).

	subspace	mod 7	mod 11	mod 13
keep	$V_{11} \otimes V_{13}$	0.14	**0.89**	**0.89**
keep	$V_7 \otimes V_{11} \otimes V_{13}$	0.14	0.09	0.07
keep	both	**1.00**	**1.00**	**1.00**
kill	$V_7 \otimes V_{11} \otimes V_{13}$	0.14	**0.89**	**0.89**
kill	$V_{11} \otimes V_{13}$	0.14	0.09	0.08



(a) $n = 1001$, seed 10 (b) $n = 1001$, seed 7 (c) n=595, seed 3 (d) $n = 1001$, seed 14

Figure 6: **Partial modularity and dependency graphs.** This shows the readout block energy over training, and the resulting subtask dependency graphs for four models trained on modular addition. An arrow between subtasks $\mathcal{A} \to \mathcal{B}$ means "$\mathcal{A}$ depends on $\mathcal{B}$," in the sense that ablating the subspace carrying $\mathcal{B}$ degrades $\mathcal{A}$. A double-headed arrow marks a fused pair.

7 Conclusion

Models jointly trained on addition and multiplication $\mod n$ implement the Chinese remainder theorem: errors factor across primes, each prime is computed in its own subspace, and each prime largely uses its own set of neurons. Models trained on addition alone are instead entangled or partially modular, and subtasks in partially modular models can be fused or depend asymmetrically on others. This gives a reference for what complete, emergent modularity looks like.

Future work The natural next steps are to expand to deeper transformers, where some layers may be more modular than others, and also to look at models trained on real-world data, which may have more complicated subtask decompositions and dependencies. This also may have applications to unlearning, and understanding why unlearning may degrade other capabilities in asymmetrical ways. It would also be very interesting to understand why the ring specifically is modular while addition is not. We discuss one possible explanation in Appendix D, but it would be valuable to have a complete mechanistic reason, and the ability to turn on modularity by adding the multiplication operation may be able to give insight into why modularity occurs.

8 AI USE STATEMENT

In this work, we used generative AI tools to propose or refine hypotheses, design and provide feedback on research methodology and experiments, create code, interpret results, aid with formatting figures and tables, summarize existing literature, and search for related papers. It also helped propose titles, and suggest writing improvements and phrasing. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including all texts, claims, and figures produced with the help of generative AI.

REFERENCES

Gabriel Béna and Dan F. M. Goodman. Dynamics of specialization in neural modules under resource constraints. *Nature Communications*, 16(1):187, 2025. doi: 10.1038/s41467-024-55188-9. URL https://doi.org/10.1038/s41467-024-55188-9.

Zitong Andrew Chen, Junaid Hasan, Akhil Srinivasan, Hemkesh Bandi, and Jarod Alper. Multiplication beyond groups: Stratified Fourier mechanisms in transformer circuits, 2026. URL https://arxiv.org/abs/2607.07066.

Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pp. 6243–6267, 2023.

Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. The evolutionary origins of modularity. *Proceedings of the Royal Society B: Biological Sciences*, 280(1755):20122863, 2013. doi: 10.1098/rspb.2012.2863. URL https://doi.org/10.1098/rspb.2012.2863.

Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Are neural nets modular? Inspecting functional modularity through differentiable weight masks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=7uVcpu-gMD.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition, 2022. URL https://arxiv.org/abs/2209.10652.

Daniel Filan, Stephen Casper, Shlomi Hod, Cody Wild, Andrew Critch, and Stuart Russell. Clusterability in neural networks, 2021. URL https://arxiv.org/abs/2103.03386.

Andrey Gromov. Grokking modular arithmetic, 2023. URL https://arxiv.org/abs/2301.02679.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs), 2016. URL https://arxiv.org/abs/1606.08415.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations, 2018. URL https://arxiv.org/abs/1812.02230.

Hyunsang Hwang, SuHyun Bae, and Donghun Lee. Prime Fourier embeddings: A principled basis for modular arithmetic. In *3rd AI for Math Workshop: Toward Self-Evolving Scientific Agents*, 2026. URL https://openreview.net/forum?id=SmSv0S3bUW.

Arthur Jacot, François Ged, Berfin Şimşek, Clément Hongler, and Franck Gabriel. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity, 2022. URL https://arxiv.org/abs/2106.15933.

Michael A. Lepori, Thomas Serre, and Ellie Pavlick. Break it down: Evidence for structural compositionality in neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=rwbzMiuFQl.

Ziming Liu, Mikail Khona, Ila R. Fiete, and Max Tegmark. Growing brains: Co-emergence of anatomical and functional modularity in recurrent neural networks, 2023. URL https://arxiv.org/abs/2310.07711.

Ziming Liu, Eric Gan, and Max Tegmark. Seeing is believing: Brain-inspired modular training for mechanistic interpretability. *Entropy*, 26(1), 2024. ISSN 1099-4300. doi: 10.3390/e26010041. URL https://www.mdpi.com/1099-4300/26/1/41.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Gavin McCracken, Gabriela Moisescu-Pareja, Vincent Létourneau, Doina Precup, and Jonathan Love. Uncovering a universal abstract algorithm for modular addition in neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=zuHs6RHQwT.

Eric J Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=3tbTw2ga8K.

Sarthak Mittal, Yoshua Bengio, and Guillaume Lajoie. Is a modular architecture enough? In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=3-3XMModtrx.

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.

Huu Danh Nguyen. The discrete-log clock: How a transformer learns modular multiplication. In *Mechanistic Interpretability Workshop at ICML 2026*, 2026. URL https://openreview.net/forum?id=DOuOKdrxE1.

Thanh Tam Nguyen, Thanh Trung Huynh, Zhao Ren, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning, 2024. URL https://arxiv.org/abs/2209.02299.

Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40): 24652–24663, 2020. doi: 10.1073/pnas.2015509117. URL https://www.pnas.org/doi/abs/10.1073/pnas.2015509117.

Scott Pesme and Nicolas Flammarion. Saddle-to-saddle dynamics in diagonal linear networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 7475–7505. Curran Associates, Inc., 2023. doi: 10.52202/075280-0329. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/17a9ab4190289f0e1504bbb98d1d111a-Paper-Conference.pdf.

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.

Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. Toward Transparent AI: A Survey on Interpreting the Inner Structures of Deep Neural Networks . In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 464–483, Los Alamitos, CA, USA, February 2023. IEEE Computer Society. doi: 10.1109/SaTML54575.2023.00039. URL https://doi.ieeecomputersociety.org/10.1109/SaTML54575.2023.00039.

Dashiell Stander, Qinan Yu, Honglu Fan, and Stella Biderman. Grokking group multiplication with cosets, 2024. URL https://arxiv.org/abs/2312.06581.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Günter P. Wagner, Mihaela Pavlicev, and James M. Cheverud. The road to modularity. *Nature Reviews Genetics*, 8(12):921–931, 2007. doi: 10.1038/nrg2267. URL https://doi.org/10.1038/nrg2267.

Evžen Wybitul, Tim G. J. Rudner, and Christian Schroeder de Witt. Entangled representations amplify collateral damage in unlearning, 2026. URL https://arxiv.org/abs/2609.02285.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 10524–10533, 2020.

Yongzhong Xu. The geometry of multi-task grokking: Transverse instability, superposition, and weight decay phase structure, 2026. URL https://arxiv.org/abs/2602.18523.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: two stories in mechanistic explanation of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 27223–27250, 2023.

A MLP

We train a two-layer MLP (embedding 256, concatenated inputs, two GELU layers of width 1024, linear readout) with otherwise identical settings. The hidden-subspace and neuron tests act on the same layer, the 1024 neurons feeding the readout.

Table 6: MLP ring, seed 0: kill diagonal (off-diagonal kills are all 1.00) and range of keep off-diagonals. Readout scored on 20,000 validation examples.

Op.	Level	kill mod 7 / 11 / 13	keep off-diag.
$+$	Readout blocks	0.15, 0.09, 0.08	0.03–0.08
	Hidden U_p	0.14, 0.10, 0.08	0.07–0.19
	Neurons $\mathcal{N}_p$	0.18, 0.13, 0.13	0.12–0.26
$\times$	Readout blocks	0.14, 0.09, 0.08	0.10–0.19
	Hidden U_p	0.27, 0.15, 0.15	0.12–0.27
	Neurons $\mathcal{N}_p$	0.34, 0.26, 0.22	0.21–0.38

Table 7: MLP addition, seed 0: readout family kill. Mod 11 survives killing either other family, so it is carried by V_{11} alone; 7 and 13 are fused.

	mod 7	mod 11	mod 13
kill 7	**0.15**	0.98	0.10
kill 11	0.14	**0.03**	0.07
kill 13	0.16	0.97	**0.07**

This gives us a dependency graph we have not seen before:

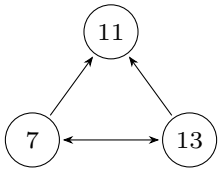


Figure 7: **Dependency graph for the MLP trained on addition (seed 0).** Mod 7 and mod 13 are fused, and both depend on 11 information, while mod 11 is independent of the others.

B Additional behavioural modularity results

B.1 Independence of accuracies

Quantity	Gap $= P(\text{joint}) - \prod P(\text{marginal})$
$P(A_7A_{11}) - P(A_7)P(A_{11})$	-0.0000 ± 0.0001
$P(A_7A_{13}) - P(A_7)P(A_{13})$	-0.0000 ± 0.0001
$P(A_{11}A_{13}) - P(A_{11})P(A_{13})$	-0.0002 ± 0.0072

Table 8: Independence of per-prime correctness on the ring (n=1001), averaged over 13 (seed, checkpoint) training points. All gaps ≈ 0: the mod-p correctness events are mutually independent throughout training.

B.2 Sequential grokking

By increasing the weight decay to 0.3 and training fraction to 0.25% we can induce grokking (delayed generalization following overfitting). Here each prime groks independently of each other, creating a sequence of grokking events:

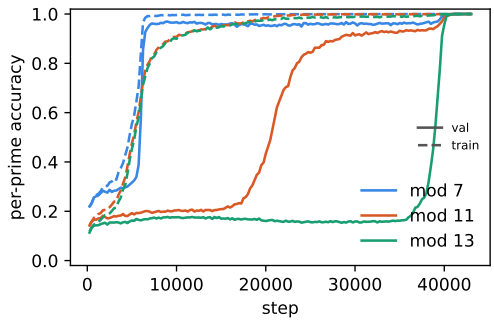


Figure 8: Per-prime training and validation accuracies. Each prime experiences delayed generalization following overfitting, indicative of grokking.

C Other partial tables

These are the ablation tables to reconstruct the dependency graphs in Figure 6.

Table 9: Ablations for $n = 1001$, seed 7. Bold = intact (> 0.8).

	Subspace	mod 7	mod 11	mod 13
keep	V_{13}	0.14	0.09	**1.00**
keep	$V_7 \otimes V_{11} \otimes V_{13}$	0.25	0.25	0.25
keep	$\{V_{13},\, V_7 \otimes V_{11} \otimes V_{13}\}$	**1.00**	**1.00**	**1.00**
kill	V_{13}	0.23	0.23	0.23
kill	$V_7 \otimes V_{11} \otimes V_{13}$	0.14	0.55	**1.00**

Table 10: Ablations for $n = 595$, seed 3. Bold = intact (> 0.8).

	Subspace	mod 5	mod 7	mod 17
keep	$V_5 \otimes V_{17}$	**0.87**	0.13	**0.87**
keep	$V_7 \otimes V_{17}$	0.21	0.64	0.64
keep	$\{V_5 \otimes V_{17},\, V_7 \otimes V_{17}\}$	**1.00**	**1.00**	**1.00**
kill	$V_5 \otimes V_{17}$	0.20	0.57	0.57
kill	$V_7 \otimes V_{17}$	**0.83**	0.16	**0.83**
kill	$\{V_5 \otimes V_{17},\, V_7 \otimes V_{17}\}$	0.19	0.14	0.06

Table 11: Ablations for $n = 1001$, seed 14. Bold = intact (> 0.8).

	Subspace	mod 7	mod 11	mod 13
keep	V_7	**0.99**	0.09	0.08
keep	V_{11}	0.15	**1.00**	0.07
keep	$V_7 \otimes V_{11}$	**0.96**	**0.96**	0.08
keep	$\{V_7 \otimes V_{11},\, V_7 \otimes V_{11} \otimes V_{13}\}$	**0.91**	**0.90**	**0.90**
kill	$\{V_7,\, V_7 \otimes V_{11}\}$	0.31	**0.99**	0.31
kill	$\{V_{11},\, V_7 \otimes V_{11}\}$	**0.86**	0.16	0.16
kill	$V_7 \otimes V_{11} \otimes V_{13}$	**1.00**	**1.00**	0.08

We can summarize the $n = 1001$ results for 20 seeds in the following table:

Table 12: Readout family kills for all 20 addition seeds ($n = 1001$). Bold: intact (> 0.8).

	kill 7			kill 11			kill 13			
Seed	7	11	13	7	11	13	7	11	13	Outcome
0	0.14	0.08	0.08	0.15	0.09	0.08	0.14	0.09	0.07	entangled
1	0.15	0.09	0.07	0.14	0.09	0.08	0.15	0.10	0.08	entangled
2	0.15	0.10	0.07	0.14	0.09	0.08	0.14	0.09	0.07	entangled
3	0.15	**0.92**	**0.92**	0.14	0.09	0.08	0.14	0.09	0.08	partial
4	0.14	0.09	0.08	0.15	0.10	0.08	0.15	0.10	0.08	entangled
5	0.14	**0.93**	0.07	0.14	0.11	0.07	**1.00**	**1.00**	0.08	partial
6	0.13	0.10	0.08	0.14	0.10	0.08	0.14	0.10	0.07	entangled
7	0.12	0.53	**1.00**	0.15	0.08	**1.00**	0.14	0.09	0.06	partial
8	0.16	0.08	0.43	**1.00**	0.08	**1.00**	0.15	0.09	0.08	partial
9	0.14	0.16	0.16	0.14	0.08	0.07	0.14	0.09	0.07	entangled
10	0.15	**0.89**	**0.89**	0.14	0.09	0.08	0.14	0.09	0.07	partial
11	0.14	0.09	0.08	0.15	0.09	0.08	0.15	0.09	0.08	entangled
12	0.14	**1.00**	**1.00**	0.22	0.10	0.22	0.14	0.10	0.08	partial
13	0.15	0.12	0.12	0.14	0.10	0.07	0.14	0.09	0.08	entangled
14	0.15	**1.00**	0.08	**0.98**	0.08	0.07	**1.00**	**1.00**	0.08	partial
15	0.14	0.09	0.07	0.15	0.09	0.07	0.16	0.09	0.07	entangled
16	0.14	**1.00**	**1.00**	0.14	0.07	**0.87**	0.14	0.56	0.07	partial
17	0.13	0.10	0.51	**1.00**	0.09	**1.00**	0.78	0.09	0.07	partial
18	0.14	0.75	0.75	0.14	0.08	0.08	0.14	0.09	0.08	entangled
19	0.14	0.09	0.08	0.13	0.09	0.08	0.14	0.09	0.08	entangled

Partially modular: $9/20$, fully modular: $0/20$.

D THE UNIT GROUP

It is natural to assume that the ring is modular is because multiplication $\bmod n$ does not form a cyclic group. In particular, the unit group of the integers $\bmod n$ is not cyclic. To test if this is the reason that the ring is modular, we will train a model with the same setup as in Section 3.2 to learn just the unit group of multiplication $\bmod 1001$.

Across 8 seeds, the kill and keep results of ablating the readout (the same method as in Table 2), are summarized below in Table 13 and Table 14. Surprisingly, the model is *not* cleanly modular in any seed. This suggests that the reason the ring is consistently modular has something to do with the relationship between the addition and multiplication operations, rather than due to just multiplication itself.

Table 13: **Unit group kill matrix.** Seeds 0–7 ($n = 1001$). Bold > 0.8. Chance is $1/(p-1) = 0.17, 0.10, 0.08$. Models are *not* modular.

	kill 7			kill 11			kill 13		
Seed	7	11	13	7	11	13	7	11	13
0	0.17	**1.00**	0.33	0.51	0.11	0.09	0.48	0.50	0.08
1	0.18	**1.00**	0.25	0.51	0.10	0.09	0.50	0.25	0.09
2	0.17	**0.85**	0.28	0.17	0.12	0.09	0.17	0.21	0.08
3	0.15	0.30	0.30	0.80	0.09	0.80	0.35	0.10	0.10
4	0.17	0.17	0.35	0.16	0.10	0.33	0.17	0.10	0.07
5	0.16	**0.93**	0.71	0.44	0.11	0.21	0.48	**0.98**	0.09
6	0.18	0.25	0.13	0.16	0.09	0.08	0.22	0.22	0.09
7	0.16	0.10	0.09	0.50	0.10	0.09	0.49	0.11	0.08

Table 14: **Unit group keep matrix.** Seeds 0–7 ($n = 1001$). Bold > 0.8. Chance is $1/(p-1) = 0.17, 0.10, 0.08$. Models are *not* modular.

	keep 7			keep 11			keep 13		
Seed	7	11	13	7	11	13	7	11	13
0	0.76	0.49	0.36	**1.00**	**1.00**	**1.00**	0.71	0.63	0.61
1	0.62	0.31	0.28	0.64	**1.00**	0.64	**0.99**	**0.99**	**0.99**
2	0.80	0.79	0.79	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
3	**0.98**	**0.98**	**0.98**	0.48	0.48	0.50	**1.00**	**1.00**	**1.00**
4	**1.00**	**1.00**	**1.00**	**0.86**	**0.83**	**0.83**	**1.00**	**1.00**	**1.00**
5	0.46	0.45	0.23	**0.91**	**0.99**	**0.90**	**0.81**	**0.81**	**0.81**
6	**1.00**	**1.00**	**0.99**	**1.00**	**1.00**	**1.00**	**0.91**	**0.91**	**0.91**
7	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**