Grokking

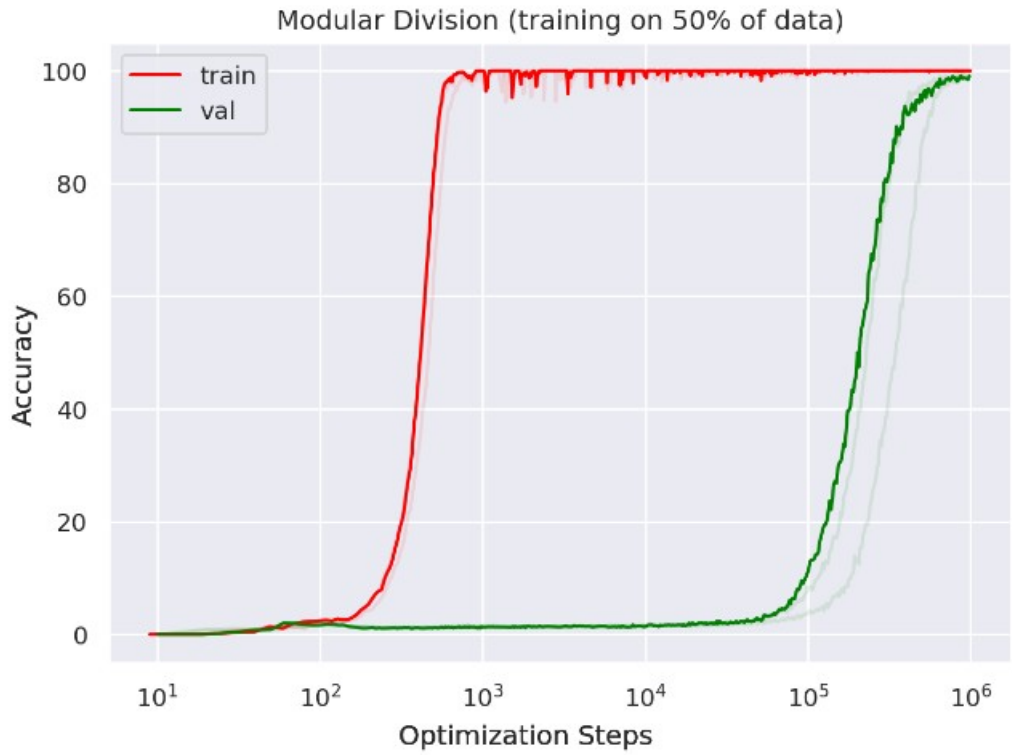
2023.11.10

Contents

- Grokking: Generalization beyond Overfitting on Small Algorithmic Datasets (2021, iclr workshop)
- Progress Measures for Grokking via Mechanistic Interpretability (2023, iclr spotlight)
- The Clock and the Pizza: Two Stories in Mechanistic Explanation of Neural Networks (2023, neurips oral)

What is grokking(顿悟)?

- long after severely overfitting,
- validation accuracy sometimes suddenly begins to increase from chance level toward perfect generalization.
- We call this phenomenon ***"grokking"***.



-- (OpenAI) Grokking: Generalization beyond Overfitting on Small Algorithmic Datasets

Why do we study grokking?

- Do neural networks memorize or generalize?
- Can we predict grokking through some smoother metrics?

Progress Measures for Grokking via Mechanistic Interpretability

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, Jacob Steinhardt

set up and background

task:

- train transformer to perform modular addition: a + b = c (mod P)
- input: "a b =", where a and b are encoded as P-dimensional 1-hot vector

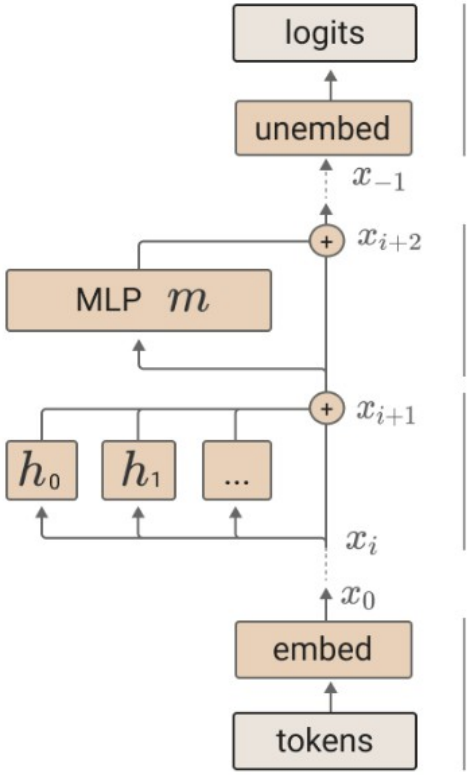
dataset:

- $P = 113$
- 30% of all possible inputs $(30\% \cdot 113 \cdot 113)$ as training set

training setting:

- full batch gradient descent
- AdamW optimizer with learning rate $\gamma = 0.001$ and weight decay parameter $\lambda = 1$
- 40,000 epochs of training
- evaluate test loss and accuracy on all pairs of inputs not used for training

set up and background - model



$$Logits = W_U x^{(2)}$$

$$MLP = ReLU(W_{in} x^{(1)})$$

$$x^{(2)} = W_{out} MLP + x^{(1)}$$

$$A^j = softmax({x^{(0)}}^T {W_K^j}^T W_Q^j x_2^{(0)})$$

$$x^{(1)} = \left[\sum_j W_O^j W_V^j \left(x^{(0)} \cdot A^j\right)\right] + x_2^{(0)}$$

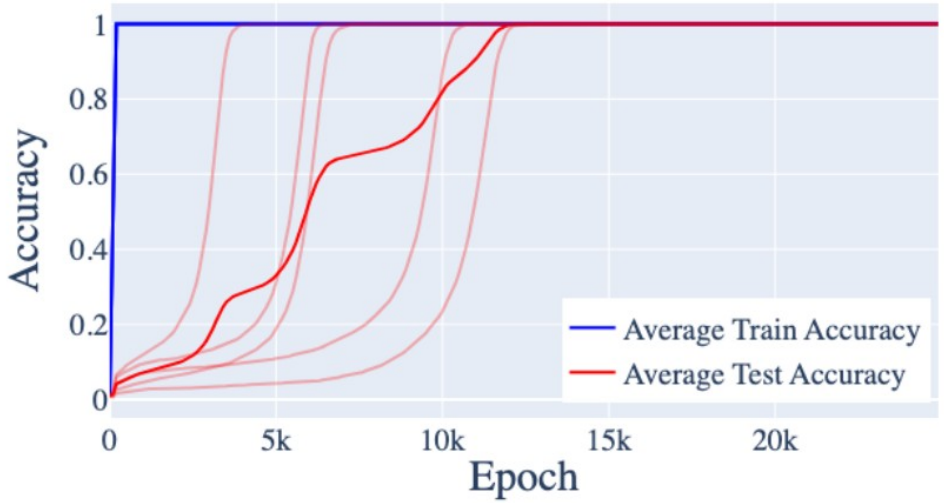
- $$x_i^{(0)} = W_E t_i + p_i$$

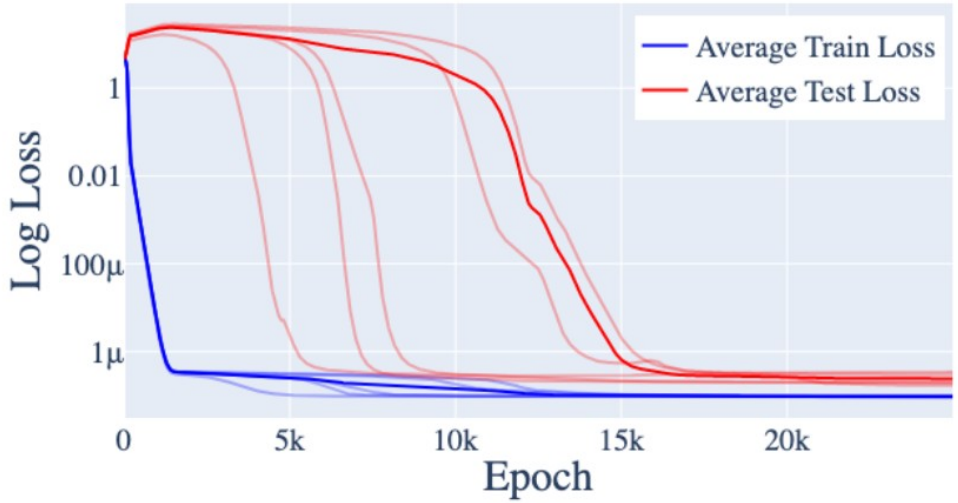
one-layer ReLU transformer:

- token embedding: $d = 128$
- learned positional embedding
- 4 attention heads of $d_{head} = 32$
- 512 hidden units in MLP

- $x_2^{(0)}$: token embedding of "="
- $W_L = W_U W_{out}$

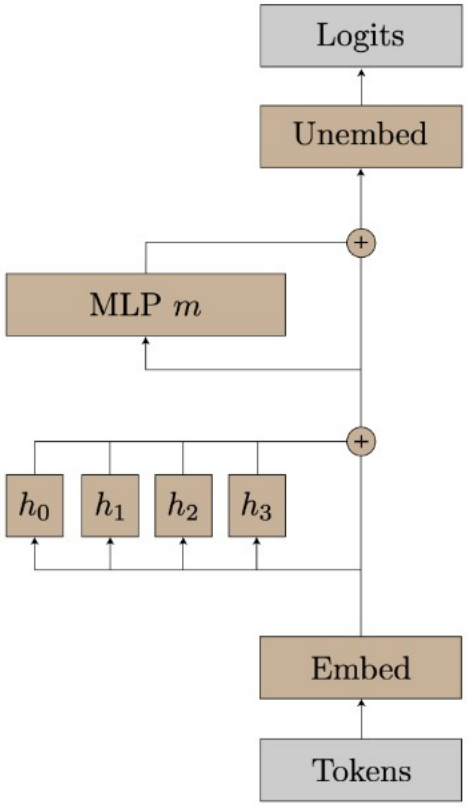
Networks trained on this task consistently exhibit grokking!





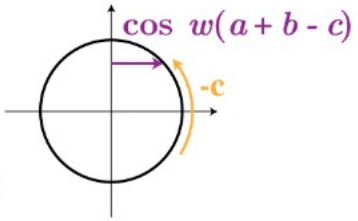
- **Stage 1:** The networks first **overfit** the training set: train accuracy quickly converges to 100% and the train loss quickly declines, while the test accuracy remains low and the test loss remains high.
- **Stage 2:** After around 10, 000 epochs, the network **generalizes** and test accuracy increases to near 100%.

Hypothesis: fourier multiplication algorithm



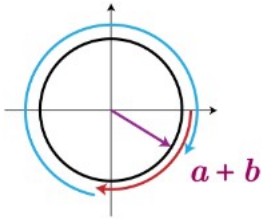
3. Computes logits using further trig identities:

$$\text{Logit}(c) \propto \cos(w(a+b-c))$$
$$= \cos(w(a+b))\cos(wc) + \sin(w(a+b))\sin(wc)$$



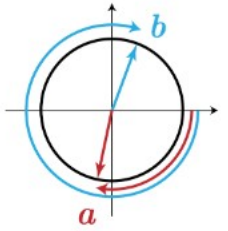
2. Calculates sine and cosine of $a+b$ using trig identities:

$$\sin(w(a+b)) = \sin(wa)\cos(wb) + \cos(wa)\sin(wb)$$
$$\cos(w(a+b)) = \cos(wa)\cos(wb) - \sin(wa)\sin(wb)$$



1. Translates one-hot a, b to Fourier basis:

$$a \to \sin(wa), \cos(wa)$$
$$b \to \sin(wb), \cos(wb)$$



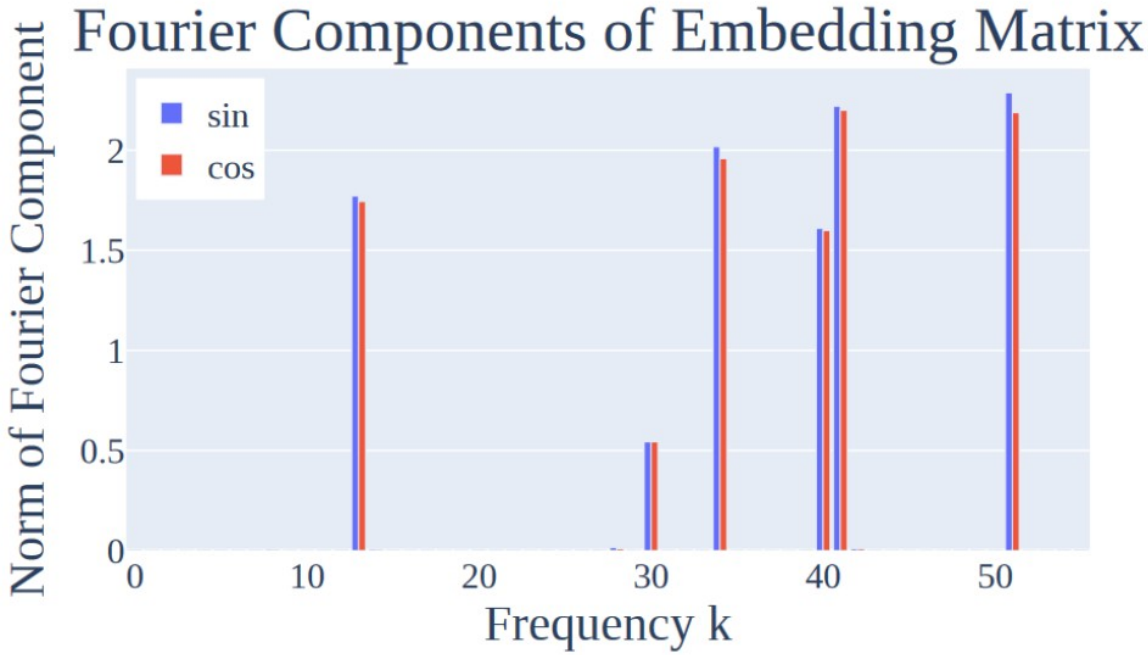
Reverse engineering a one-layer transformer

- Evidence 1: periodic structure in weights and activations
- Evidence 2: actual mechanisms implemented in the model weights
- Evidence 3: zoom in to individual neurons
- Evidence 4: use ablations to confirm that the interpretation is faithful
  - replace various components of the model by the components of the Fourier multiplication algorithm and find that doing so consistently does not harm and sometimes even improves model performance

Evidence 1: periodicity

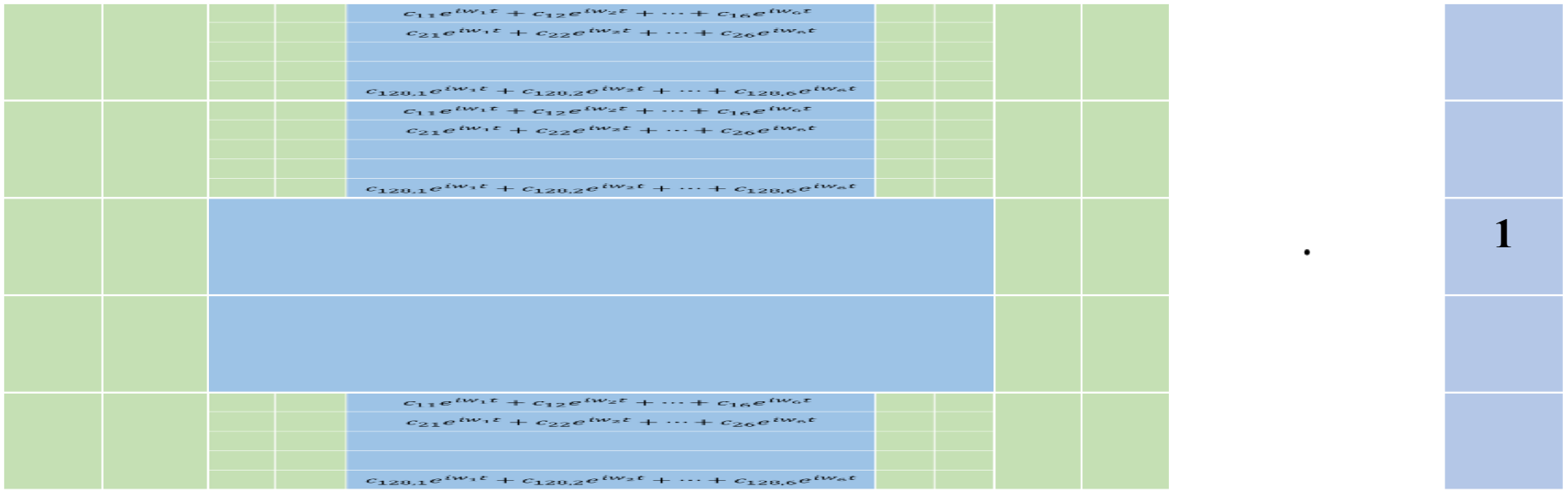
Periodicity: the output of every part of the network is periodic as a function of the input tokens. i.e. $\sim e^{i\omega_k a}$ or $e^{i\omega_k b}$

- **periodicity in the embeddings**
- We apply a Fourier transform along the input dimension of the embedding matrix W_E then compute the $\ell 2$-norm along the other dimension.
- The embedding matrix W_E is sparse in the Fourier basis–it only has significant nonnegligible norm at 6 frequencies. Of these frequencies, only 5 appear to be used significantly in later parts of the model (corresponding to $k \in \{14, 35, 41, 42, 52\}$). We dub these the key frequencies of the model.
- $\omega_k = \frac{2\pi k}{p}$



Evidence 1: periodicity

- **periodicity in the embeddings**



$W_E \in \mathbb{R}^{128\times113}$

$t_a \in \mathbb{R}^{113}$

Evidence 1: periodicity

- **periodicity in attention heads and MLP neuron activations**
- **periodicity in logits**

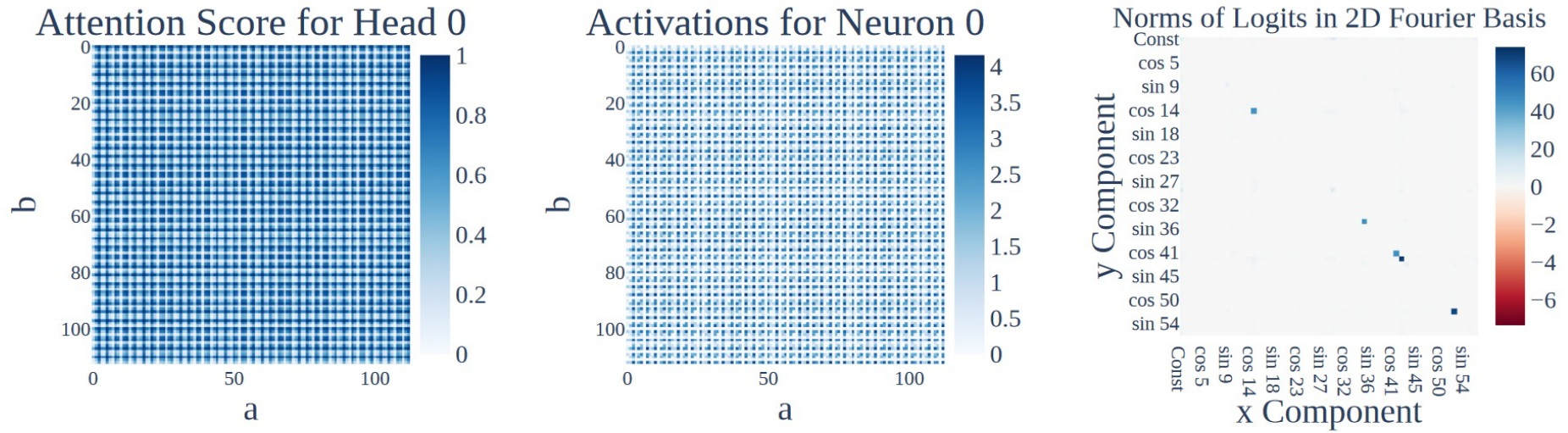


Figure 4: *(Left)* The attention score for head 0 from the token '=' to 'a', as a function of inputs a, b. *(Center)* The activations of MLP neuron 0 given inputs a, b. Both the attention scores and the neuron activations are periodic (Section 4.1). *(Right)* The norm of the Fourier components of the logits (2D Fourier transform is taken over the inputs a, b, and then norm is taken over the logit axis). There are 20 significant components corresponding to the 5 key frequencies (Section 4.1).

Evidence 2: mechanism

- We now demonstrate that the model implements the trigonometric identity as follows:

1. the functions $\cos(\omega_k(a+b)), \sin(\omega_k(a+b))$ are linearly represented in the MLP activations,
2. and the unembed matrix reads these linear directions and multiplies them by $\cos(\omega_k c), \sin(\omega_k c)$ respectively

- We will do this in two steps:

1. W_L is approximately rank 10 and can be approximated as:

$$W_L = \sum_{k\in\{14,35,41,42,52\}} \cos(\omega_k)\, u_k^T + \sin(\omega_k)\, v_k^T$$

$\cos(\omega_k), \sin(\omega_k) \in \mathbb{R}^{113}$ are vectors whose cth entry is $\cos(\omega_k c)$ and $\sin(\omega_k c)$

2. then we have:

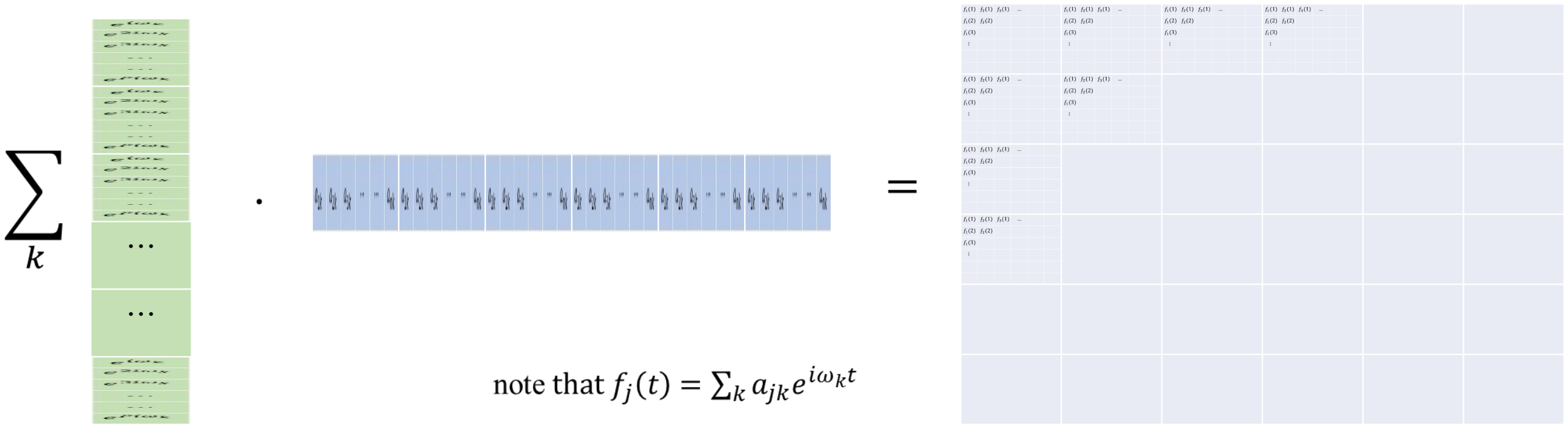
$$Logits(a,b) = W_L MLP(a,b) \approx \sum_k \cos(\omega_k)\, u_k^T MLP(a,b) + \sin(\omega_k)\, v_k^T MLP(a,b)$$

we check empirically that the terms $u_k^T MLP(a,b)$ and $v_k^T MLP(a,b)$ are approximate multiples of $\cos(\omega_k(a+b)), \sin(\omega_k(a+b))$

Evidence 2: mechanism

- **claim 1:** W_L is well approximated by $\cos(\omega_k)$ and $\sin(\omega_k)$

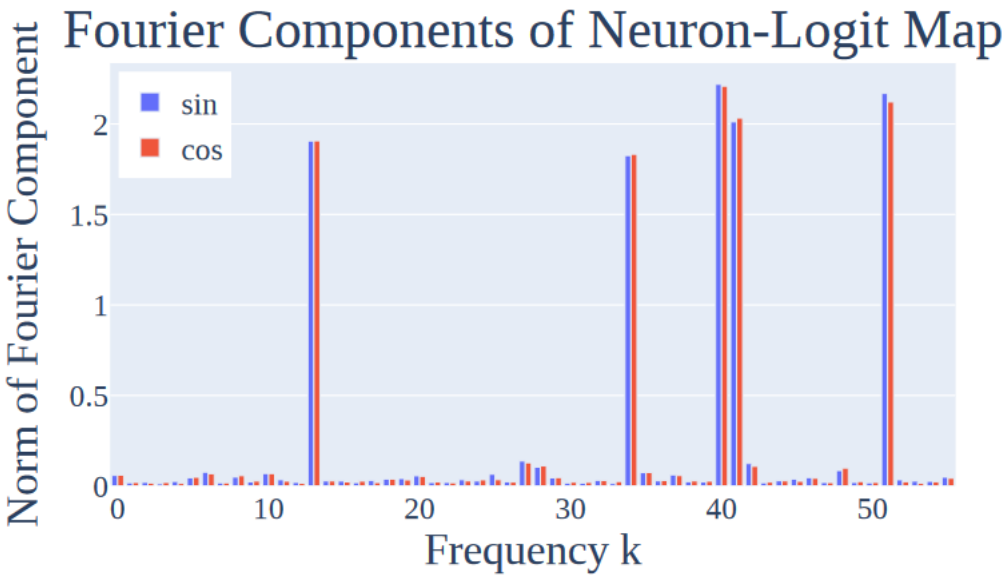
$$W_L = \sum_{k \in \{14,35,41,42,52\}} \cos(\omega_k)\, u_k^T + \sin(\omega_k)\, v_k^T$$

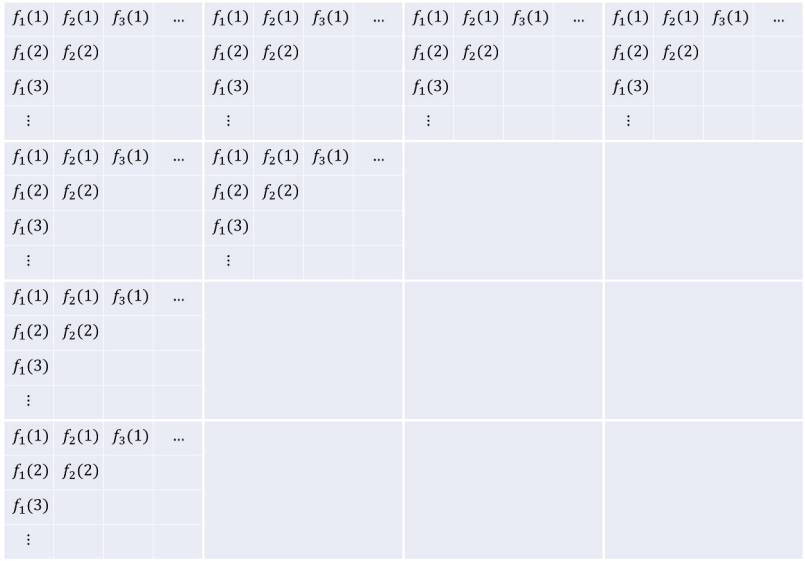


note that $f_j(t) = \sum_k a_{jk} e^{i\omega_k t}$

Evidence 2: mechanism

- **claim 1:** W_L is well approximated by $\cos(\omega_k c)$ and $\sin(\omega_k c)$
- Note that $f_j(t) = \sum_i a_{ij} e^{i\omega_i t}$. We perform a discrete Fourier transform on the logit axis of W_L and look at the 10 directions u_k, v_k corresponding to $\cos(\omega_k)$ and $\sin(\omega_k)$.
- Then we can approximate W_L: $W_L = \sum_{k\in\{14,35,41,42,52\}} \cos(\omega_k)\, u_k^T + \sin(\omega_k)\, v_k^T$.
- The residual has Frobenius norm that is under 0.55% of the norm of W_L.
- → This shows that WL is well approximated by the 10 directions corresponding to $\cos(\omega_k)$ and $\sin(\omega_k)$ for each of the five key frequencies.





Evidence 2: mechanism

- **claim 2:** The unembedding matrix "reads off" terms of the form $\cos(\omega_k(a + b))$ and $\sin(\omega_k(a + b))$ from the MLP neurons.

W_L Component	Fourier components of $u_k^T \text{MLP}(a, b)$ or $v_k^T \text{MLP}(a, b)$	FVE
$\cos(w_{14}c)$	$44.6\cos(w_{14}a)\cos(w_{14}b) - 43.6\sin(w_{14}a)\sin(w_{14}b) \approx 44.1\cos(w_{14}(a+b))$	93.2%
$\sin(w_{14}c)$	$44.1\sin(w_{14}a)\cos(w_{14}b) + 44.1\cos(w_{14}a)\sin(w_{14}b) \approx 44.1\sin(w_{14}(a+b))$	93.5%
$\cos(w_{35}c)$	$40.7\cos(w_{35}a)\cos(w_{35}b) - 43.6\sin(w_{35}a)\sin(w_{35}b) \approx 42.2\cos(w_{35}(a+b))$	96.8%
$\sin(w_{35}c)$	$41.8\sin(w_{35}a)\cos(w_{35}b) + 41.8\cos(w_{35}a)\sin(w_{35}b) \approx 41.8\sin(w_{35}(a+b))$	96.5%
$\cos(w_{41}c)$	$44.8\cos(w_{41}a)\cos(w_{41}b) - 44.8\sin(w_{41}a)\sin(w_{41}b) \approx 44.8\cos(w_{41}(a+b))$	97.0%
$\sin(w_{41}c)$	$44.5\sin(w_{41}a)\cos(w_{41}b) + 44.5\cos(w_{41}a)\sin(w_{41}b) \approx 44.5\sin(w_{41}(a+b))$	97.0%
$\cos(w_{42}c)$	$64.6\cos(w_{42}a)\cos(w_{42}b) - 68.5\sin(w_{42}a)\sin(w_{42}b) \approx 66.6\cos(w_{42}(a+b))$	96.4%
$\sin(w_{42}c)$	$67.8\sin(w_{42}a)\cos(w_{42}b) + 67.8\cos(w_{42}a)\sin(w_{42}b) \approx 67.8\sin(w_{42}(a+b))$	96.4%
$\cos(w_{52}c)$	$60.5\cos(w_{52}a)\cos(w_{52}b) - 65.5\sin(w_{52}a)\sin(w_{52}b) \approx 63.0\cos(w_{52}(a+b))$	97.4%
$\sin(w_{52}c)$	$64.5\sin(w_{52}a)\cos(w_{52}b) + 64.5\cos(w_{52}a)\sin(w_{52}b) \approx 64.5\sin(w_{52}(a+b))$	98.2%

Table 1: For each of the directions u_k or v_k (corresponding to the $\cos(w_k)$ and $\sin(w_k)$ components respectively) in the unembedding matrix, we take the dot product of the MLP activations with that direction, then perform a Fourier transform (middle column; only two largest coefficients shown). We then compute the fraction of variance explained (FVE) if we replace the projection with a single term proportional to $\cos(w_k(a+b))$ or $\sin(w_k(a+b))$, and find that it is consistently close to 1.

Evidence 2: mechanism

- **claim 3:** Logits are well approximated by a weighted sum of $\cos(\omega_k(a+b-c))$s.
- logits: $113 \cdot 113 \cdot 113$
- We approximate the output logits as $\sum_k \alpha_k \cos(\omega_k(a+b-c))$ for $k \in \{14, 35, 41, 42, 52\}$ and fit the coefficients α_k via ordinary least squares.
- This approximation explains 95% of the variance in the original logits.

Now we have 3 claims:

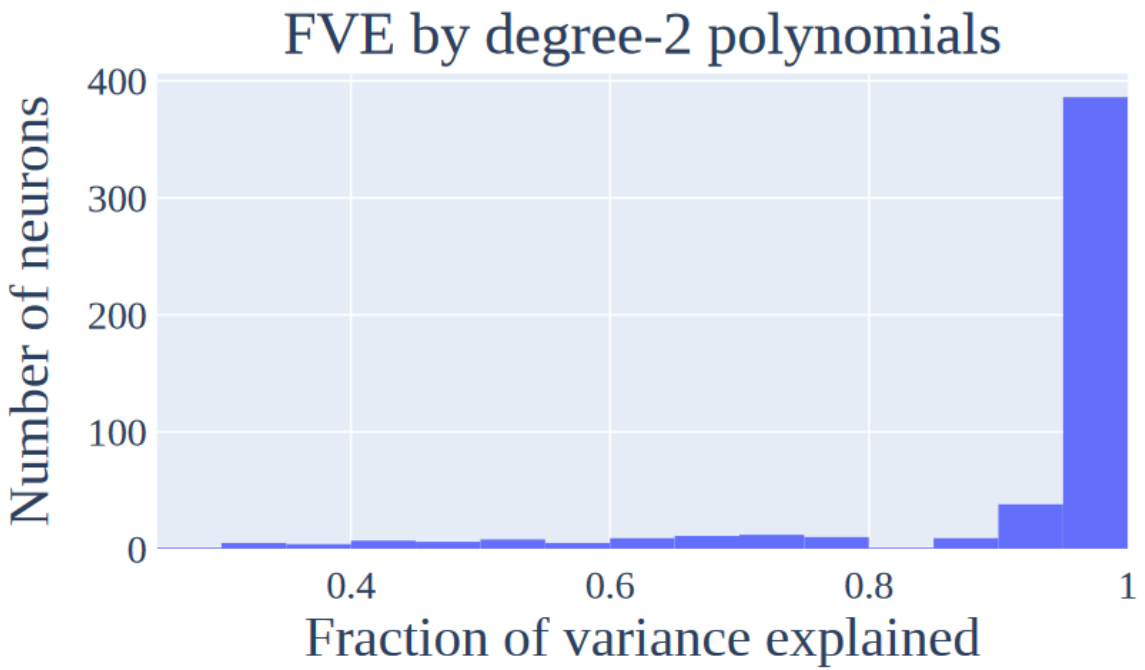
- **claim 1:** W_L is well approximated by $\cos(\omega_k c)$ and $\sin(\omega_k c)$.
- **claim 2:** The unembedding matrix "reads off" terms of the form $\cos\big(\omega_k(a+b)\big)$ and $\sin\big(\omega_k(a+b)\big)$ from the MLP neurons.
- **claim 3:** Logits are well approximated by a weighted sum of $\cos(\omega_k(a+b-c))$s.

→ Taken together, these results confirm that the model computes sums of terms of the form:

$$\boldsymbol{\cos(\omega_k(a+b-c)) = \cos(\omega_k(a+b))\cos(\omega_k c) + \sin(\omega_k(a+b))\sin(\omega_k c).}$$

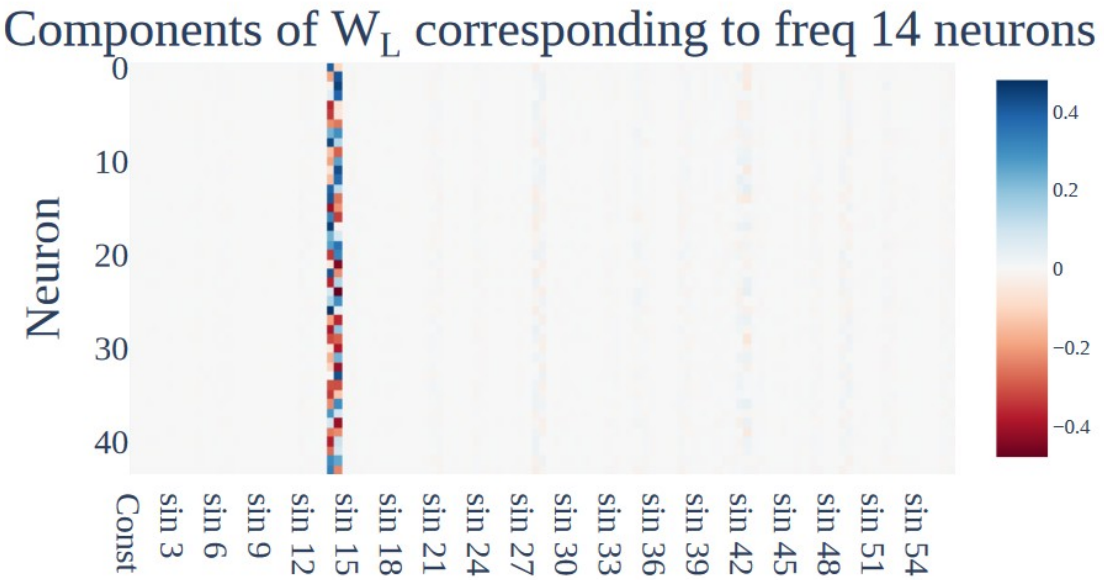
Evidence 3: individual neurons

- Most MLP neurons approximately compute a degree-2 polynomial of a single frequency.
- Out of 512 total neurons, 433 (84.6%) have over 85% of their variance explained with a single frequency.

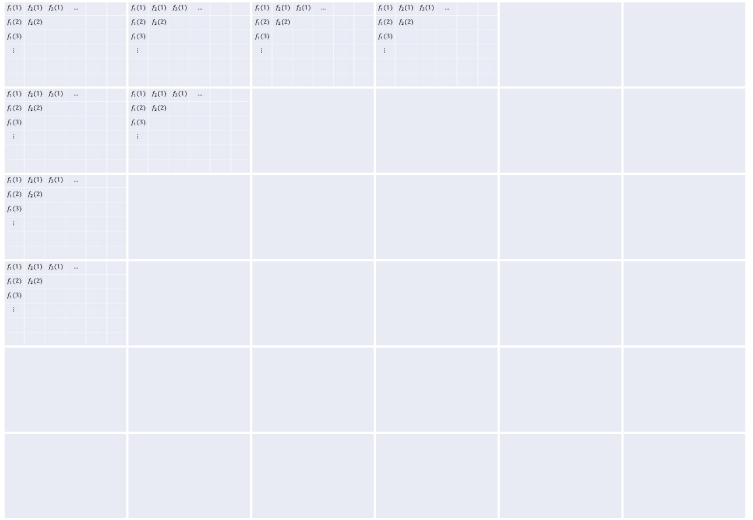


Evidence 3: individual neurons

- Maps to the logits are localized by frequency.
- We partition these 433 neurons by the frequencies with the highest variance explained.
- For each resulting subset, W_L has only two non-trivial components, corresponding to $\sin(\omega_k t)$ and $\cos(\omega_k t)$ at that frequency.

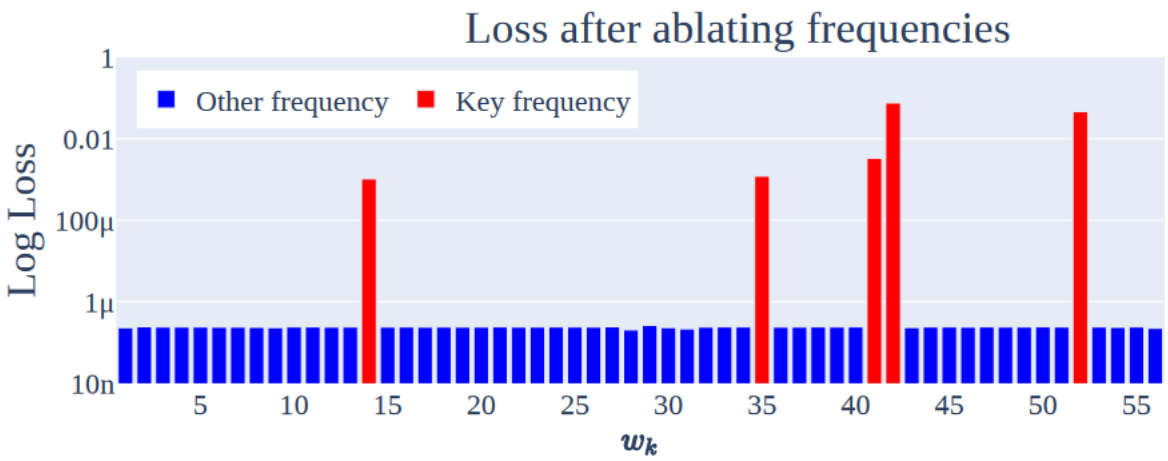


note that $f_j(t) = \sum_k a_{jk} e^{i\omega_k t}$



Evidence 4: ablations

- We verify that the approximations are faithful to the model's functionality, by replacing each component with its approximation. This generally does not hurt the performance.
- **MLP neurons**
  - In *Evidence 3*, we identified 433 neurons that were well-approximated by a degree-2 polynomial.
  - → replace each of these neurons' activation value by the corresponding polynomial
  - → This increases loss by only 3% in relative terms (from $2.41 \cdot 10^{-7}$ to $2.48 \cdot 10^{-7}$) and has no effect on accuracy.
- **Logit frequencies**
  - We take a 2D DFT on the $113 \cdot 113 \cdot 113$ logit matrix over all $113 \cdot 113$ pairs of inputs to get the logits in the Fourier basis, then set various frequencies in this basis to 0.
  - Ablating any key frequency causes a significant increase in loss. In contrast, ablating other frequencies does not hurt the model at all.



Understanding Grokking Behavior via Progress Measures

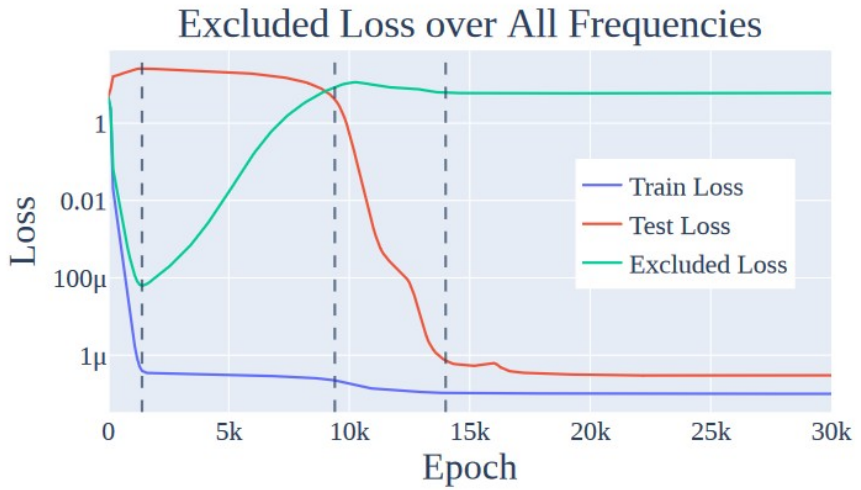
- progress measures
  - **restricted loss**
    - Since the final network uses a sparse set of frequencies ω_k, it makes sense to check how well intermediate versions of the model can do using only those frequencies.
    - perform a 2D DFT on the logits
    - set all terms besides the constant term and the terms corresponding to $\cos\big(\omega_k(a+b)\big)$ and $\sin(\omega_k(a+b))$ for the five key frequencies to 0
    - measure the loss of the ablated network
  - **excluded loss**
    - Instead of keeping the important frequencies, we next remove only those key frequencies from the logits but keep the rest.
    - We measure this on the **training data** to track how much of the performance comes from **Fourier multiplication versus memorization**.
    - **IDEA:** the memorizing solution should be spread out in the Fourier domain, so that ablating a few directions will leave it mostly unaffected, while the generalizing solution will be hurt significantly.

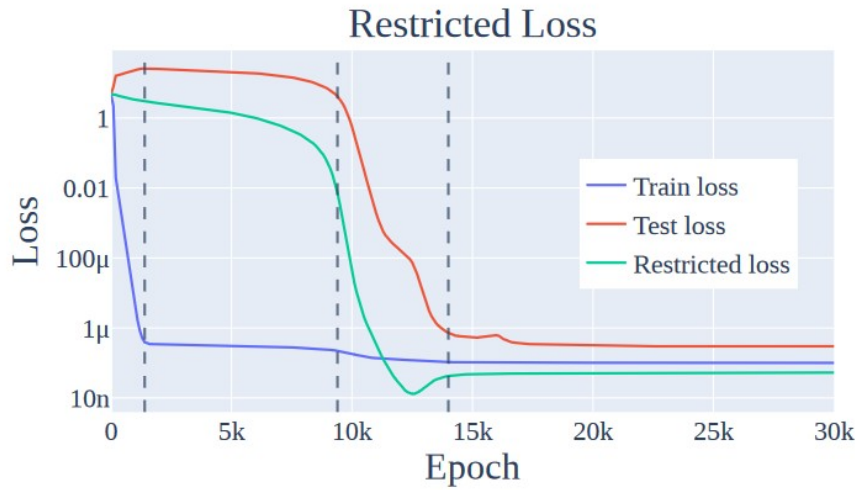
Understanding Grokking Behavior via Progress Measures

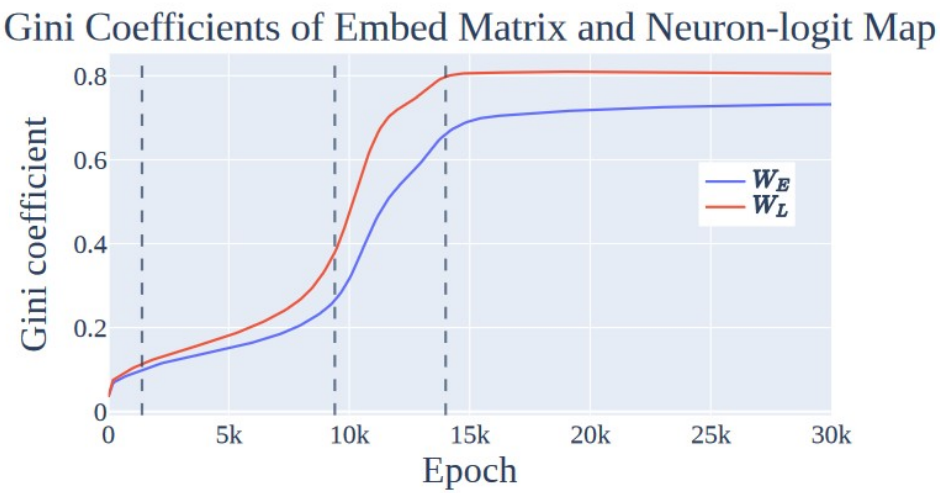
- progress measures
  - **the Gini coefficient**
    - measures the sparsity of W_E and W_L in the Fourier basis
  - **the ℓ2-norm of the weights**

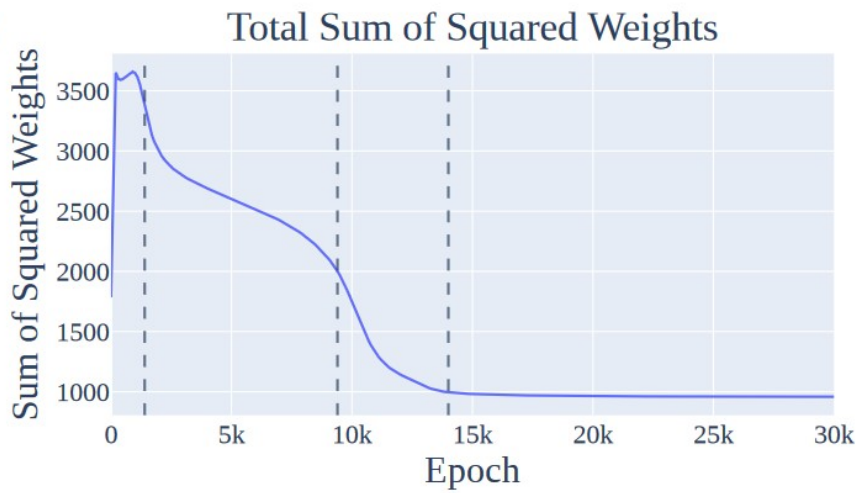
Phase of Grokking

- memorization
  - (epochs 0-1.4k)
  - excluded loss ↓
  - train loss ↓
  - restricted loss →
  - gini coefficient →
- circuit formation
  - (epochs 1.4k-9.4k)
  - excluded loss ↑
  - train, test loss →
  - restricted loss ↓
  - sum of squared weights ↓
- cleanup
  - (epochs 9.4k-14k)
  - excluded loss →
  - restricted loss ↓
  - test loss ↓↓
  - sum of squared weights ↓
  - gini coefficient ↑↑









Grokking and Weight Decay

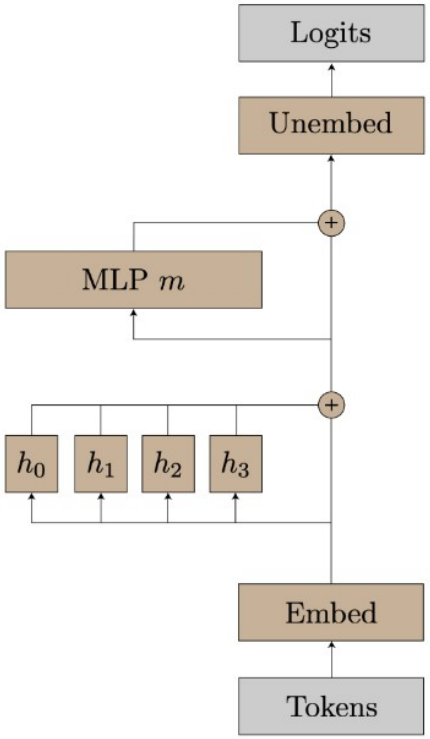
- **smaller amounts of weight decay** causes the network to take significantly **longer to grok**
- our networks **do not grok** on the modular arithmetic task **without weight decay** or some other form of regularization
- the amount of data affects grokking
  - when networks are provided with **enough data**, there is **no longer a gap** between the train and test losses

→ both regularization and limited data are necessary for grokking

The Clock and the Pizza: Two Stories in Mechanistic Explanation of Neural Networks

Ziqian Zhong, Ziming Liu, Max Tegmark, Jacob Andreas

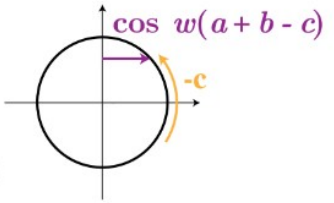
Is the way mentioned in the last paper the unique way to implement modular addition in neural networks?



the only solution?

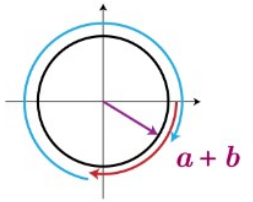
Computes logits using further trig identities:

$$\text{Logit}(c) \propto \cos(w(a+b-c))$$
$$= \cos(w(a+b))\cos(wc) + \sin(w(a+b))\sin(wc)$$



Calculates sine and cosine of $a+b$ using trig identities:

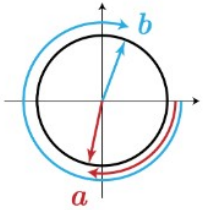
$$\sin(w(a+b)) = \sin(wa)\cos(wb) + \cos(wa)\sin(wb)$$
$$\cos(w(a+b)) = \cos(wa)\cos(wb) - \sin(wa)\sin(wb)$$



Translates one-hot a, b to Fourier basis:

$a \to \sin(wa), \cos(wa)$

$b \to \sin(wb), \cos(wb)$



Modular Arithmetic and the Clock Algorithm

- setup
  - train neural networks to perform modular addition $a + b = c \ (mod\ p)$
  - $p = 59$
  - every integer $t \rightarrow \boldsymbol{E}_t \in \mathbb{R}^d$; model input: $[\boldsymbol{E}_a, \boldsymbol{E}_b] \in \mathbb{R}^{2d}$
  - two different network architectures
    - Model A: one-layer ReLU transformer with constant attention (can be understood as a ReLU MLP)
    - Model B: a standard one-layer ReLU transformer

Review of the Clock Algorithm

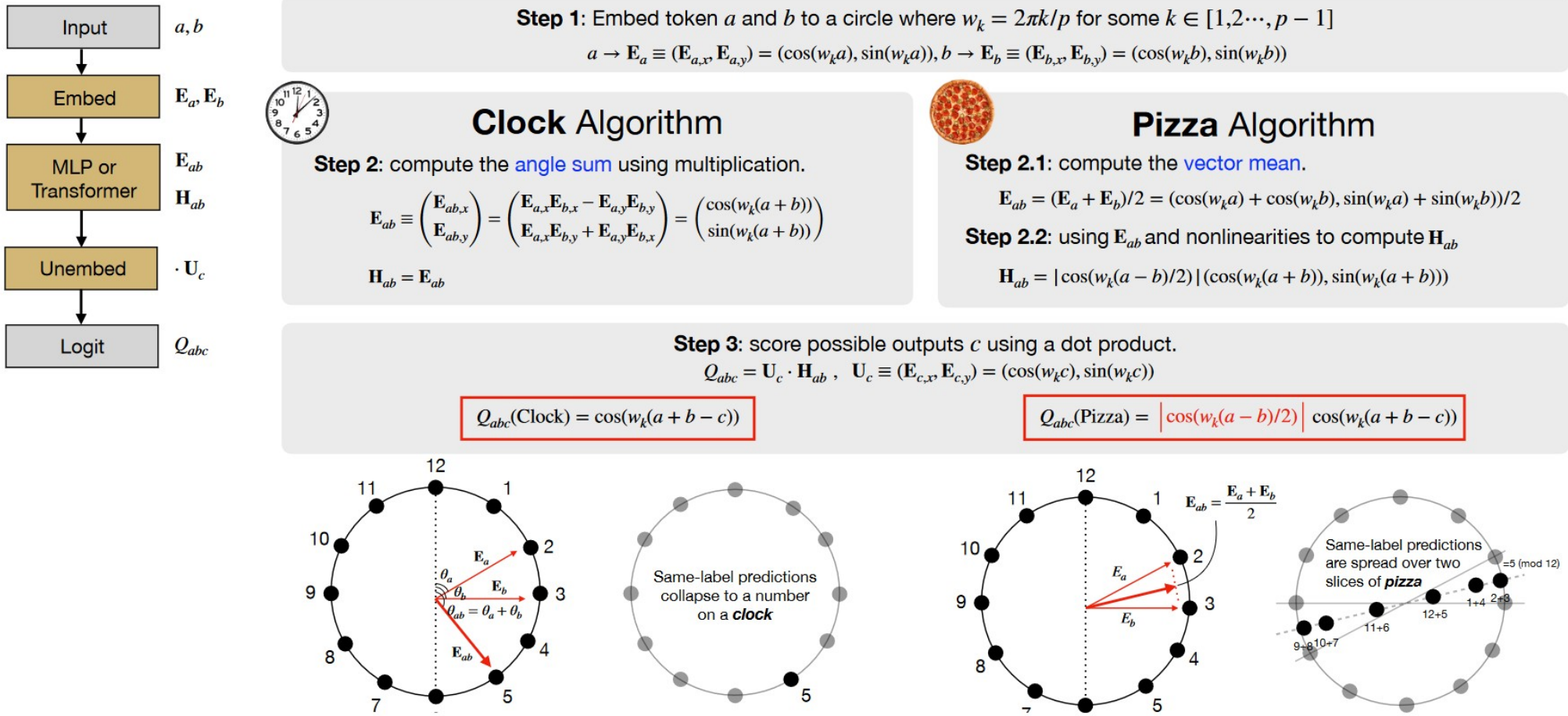


Figure 1: Illustration of the *Clock* and the *Pizza* Algorithm.

Review of the Clock Algorithm

- Crucial to this algorithm is the fact that the attention mechanism can be leveraged to perform multiplication.

→ What happens in model variants when the attention mechanism is absent, as in Model A?

- Maybe a different algorithm…

First Evidence for *Clock* Violation: Gradient Symmetricity

Since

$$Q_{abc}^{Clock} = \cos(\omega_k(a+b-c))$$

$$Q_{abc}^{Clock} = \left(E_{a,x}E_{b,x} - E_{a,y}E_{b,y}\right)E_{c,x} + \left(E_{a,x}E_{b,y} + E_{a,y}E_{b,x}\right)E_{c,y},$$

we have:

$$\nabla_{E_a} Q_{abc} \neq \nabla_{E_b} Q_{abc}$$

To check whether the equation holds true:

- We compute the 6 largest principal components of the input embedding vectors.
- We then compute the gradients of output logits with respect to the input embeddings, and then project them onto these 6 principal components.

First Evidence for *Clock* Violation: Gradient Symmetricity

Clock algorithm requires that:

$$\nabla_{E_a} Q_{abc} \neq \nabla_{E_b} Q_{abc}$$

While Model B demonstrates asymmetry in general, Model A exhibits gradient symmetry.

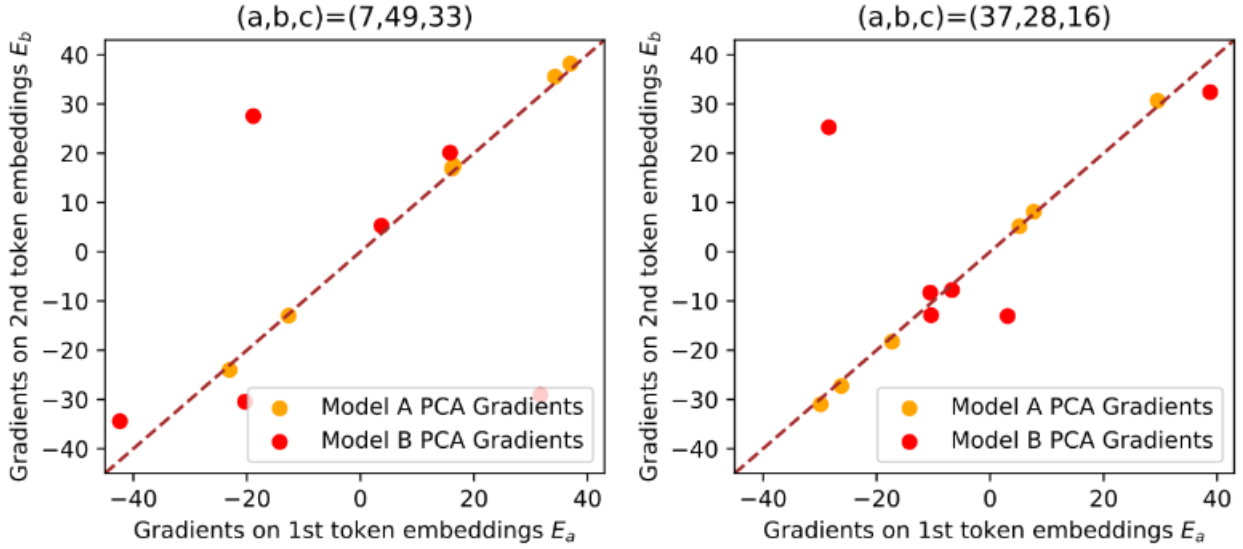


Figure 2: Gradients on first six principal components of input embeddings. (a, b, c) in the title stands for taking gradients on the output logit c for input (a, b). x and y axes represent the gradients for embeddings of the first and the second token. The dashed line $y = x$ signals a symmetric gradient.

Second Evidence for *Clock* Violation: Logit Patterns

- The correct logits of **Model A** have a clear dependency on $a - b$. Within each row, the correct logits are roughly the same.
- This pattern is not observed in **Model B**.

→ This suggests that Models A and B are implementing different algorithms.

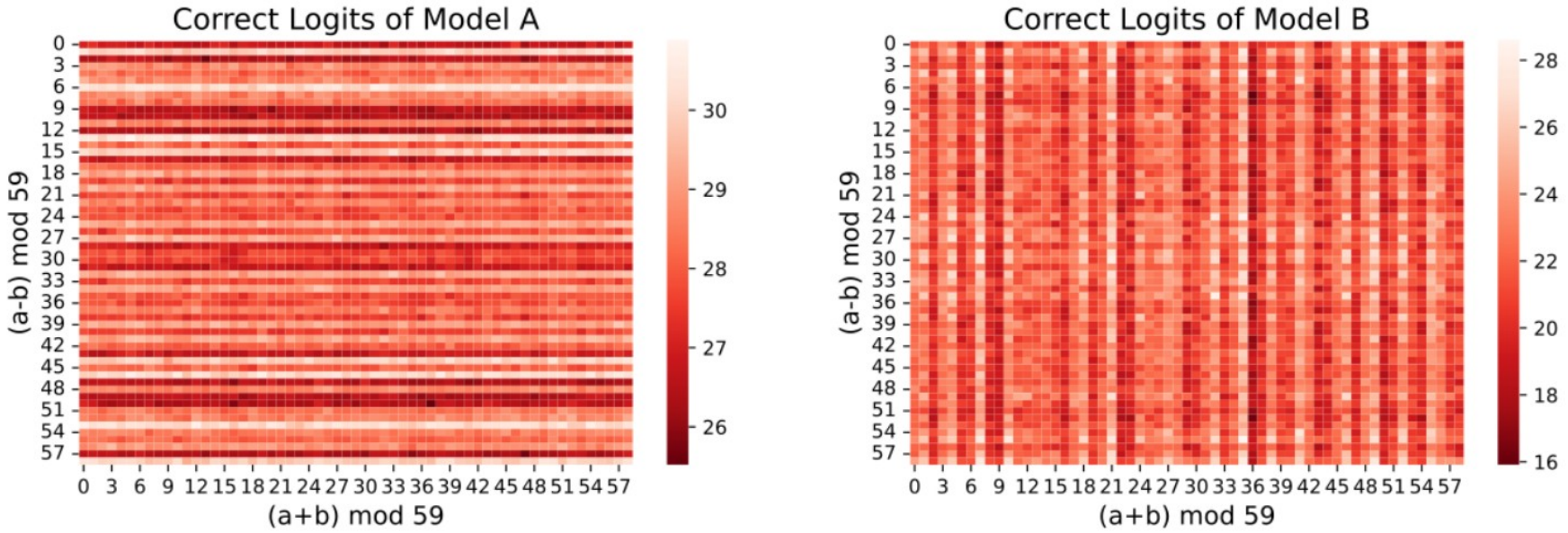


Figure 3: Correct Logits of Model A & Model B. The correct logits of Model A (left) have a clear dependence on $a - b$, while those of Model B (right) do not.

An Alternative Solution: the Pizza Algorithm

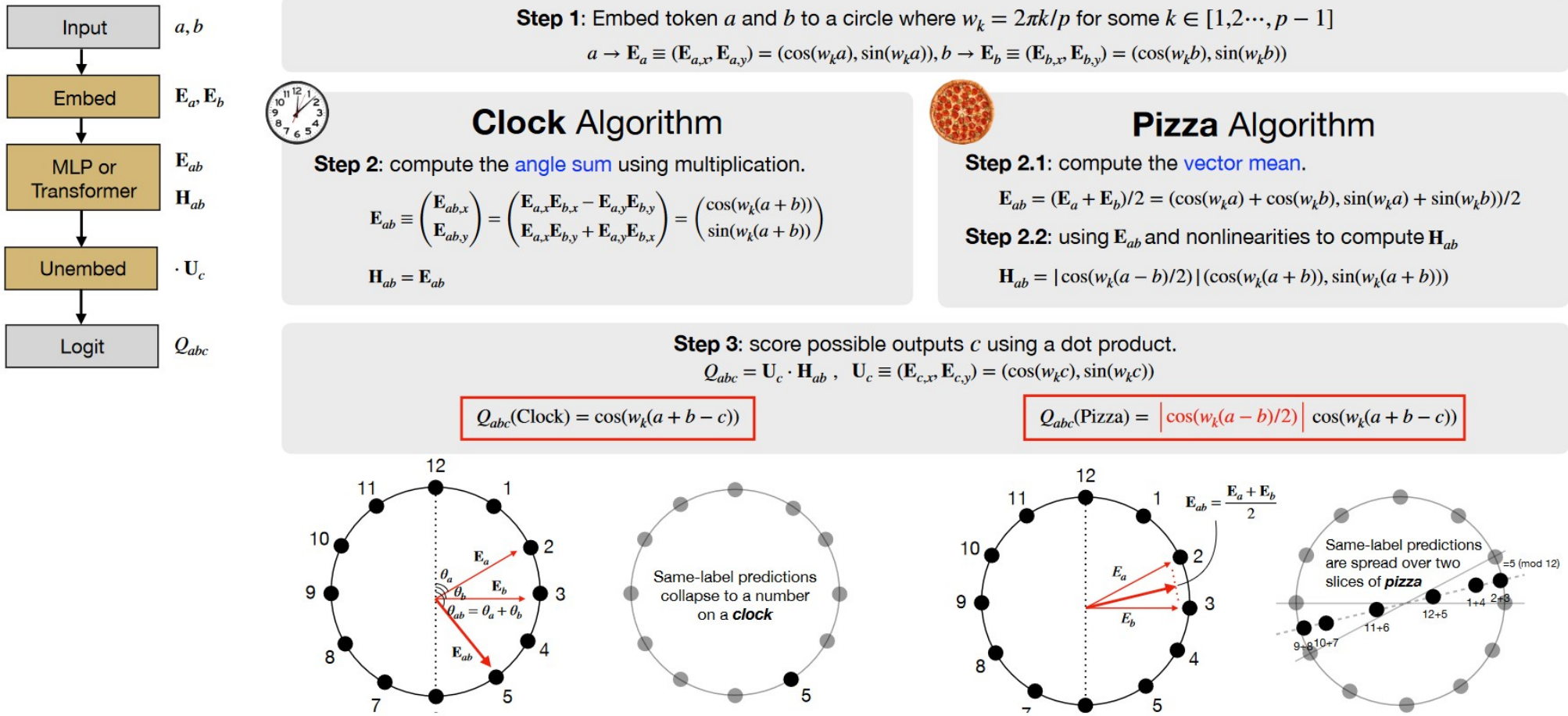


Figure 1: Illustration of the *Clock* and the *Pizza* Algorithm.

Algorithm: Pizza, Example

Step 1 Take w_k as of the accompanied pizza. On given input a and b, circularly embed them to two vectors on the circumference $(\cos(w_k a), \sin(w_k a))$ and $(\cos(w_k b), \sin(w_k b))$.

Step 2 Compute:

$$\begin{aligned}
\alpha &= |\cos(w_k a) + \cos(w_k b)| - |\sin(w_k a) + \sin(w_k b)| \\
&\approx |\cos(w_k(a-b)/2)| \cos(w_k(a+b)) \\
\beta &= |\cos(w_k a) + \cos(w_k b) + \sin(w_k a) + \sin(w_k b)|/\sqrt{2} \\
&\quad - |\cos(w_k a) + \cos(w_k b) - \sin(w_k a) - \sin(w_k b)|/\sqrt{2} \\
&= |\cos(w_k a - \pi/4) + \cos(w_k b - \pi/4)| - |\sin(w_k a - \pi/4) + \sin(w_k b - \pi/4)| \\
&\approx |\cos(w_k(a-b)/2)| \cos(w_k(a+b) - \pi/2) = |\cos(w_k(a-b)/2)| \sin(w_k(a+b))
\end{aligned}$$

Now notice $|\cos(t)| - |\sin(t)| \approx \cos(2t)$ for any $t \in \mathbb{R}$ (Figure 7), so the difference is approximately $|\cos(w_k(a-b)/2)| \cos(w_k(a+b-u))$.

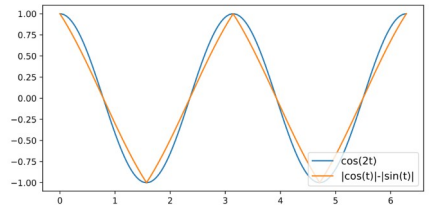


Figure 7: $|\cos(t)| - |\sin(t)|$ is approximately $\cos(2t)$ for any $t \in \mathbb{R}$

Step 3 Output of this pizza is computed as a dot product.

$$Q'_{abc} = \alpha \cos(w_k c) + \beta \sin(w_k c) \approx |\cos(w_k(a-b)/2)| \cos(w_k(a+b-c))$$

key difference

- key difference - what **non-linear operations** are required:
  - Clock requires multiplication of inputs in Step 2,
  - while Pizza requires only absolute value computation, which is easily implemented by the ReLU layers.

  → If neural networks lack inductive biases toward implementing multiplication, they may be more likely to implement Pizza rather than Clock.

First Evidence for *Pizza*: Logit Patterns

Step 3: score possible outputs c using a dot product.

$$Q_{abc} = \mathbf{U}_c \cdot \mathbf{H}_{ab}\ ,\quad \mathbf{U}_c \equiv (\mathbf{E}_{c,x}, \mathbf{E}_{c,y}) = (\cos(w_k c), \sin(w_k c))$$

$$Q_{abc}(\text{Clock}) = \cos(w_k(a+b-c))$$

$$Q_{abc}(\text{Pizza}) = \left|\cos(w_k(a-b)/2)\right| \cos(w_k(a+b-c))$$

- Q_{abc}^{Pizza} has an extra factor: $|\cos(\omega_k(a-b)/2)| \rightarrow Q_{abc}^{Pizza}$ is dependent on $a-b$
- Model A indeed exists a strong dependence on $a-b$

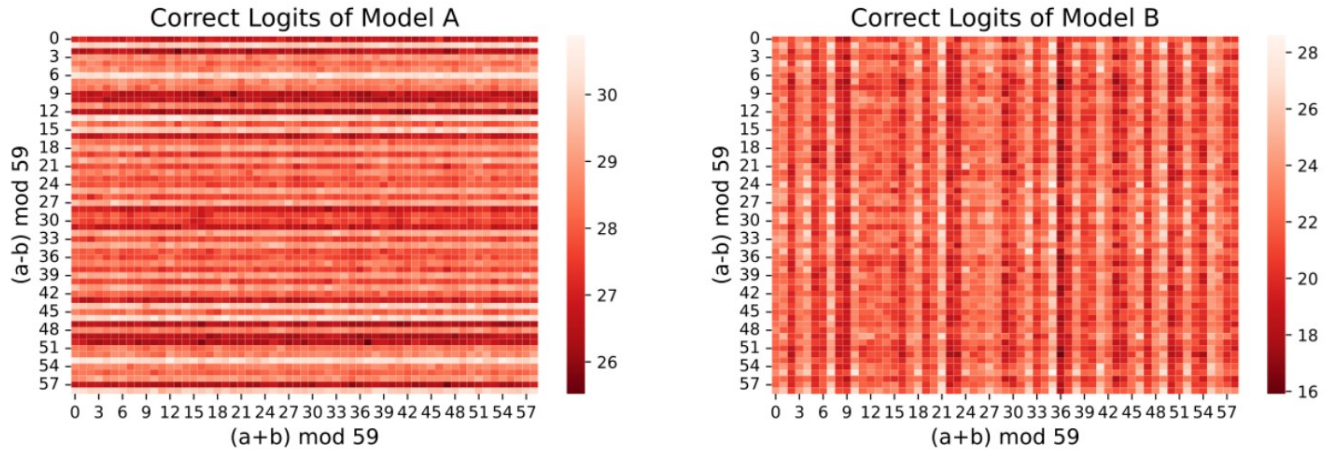
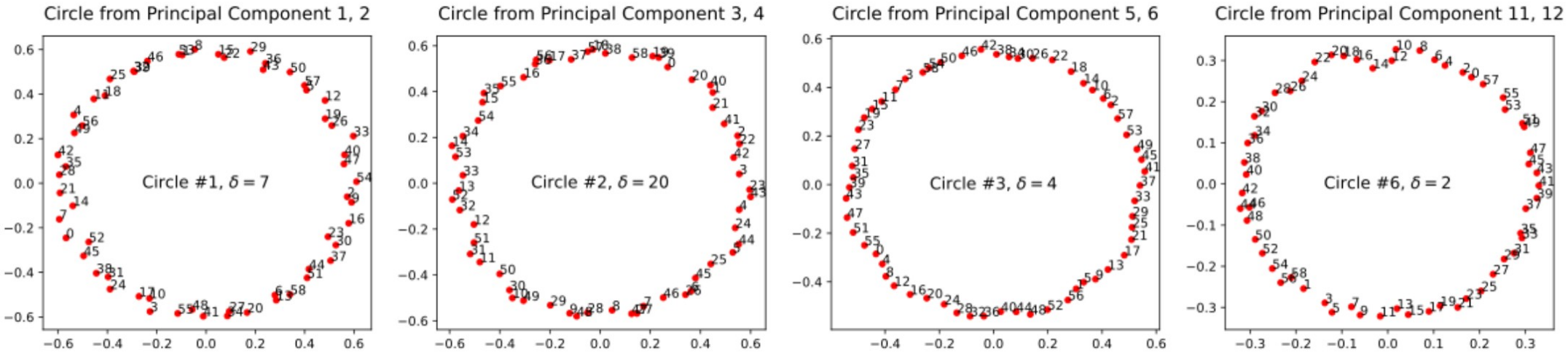


Figure 3: Correct Logits of Model A & Model B. The correct logits of Model A (left) have a clear dependence on $a-b$, while those of Model B (right) do not.

Second Evidence for *Pizza*: Clearer Logit Patterns via Circle Isolation

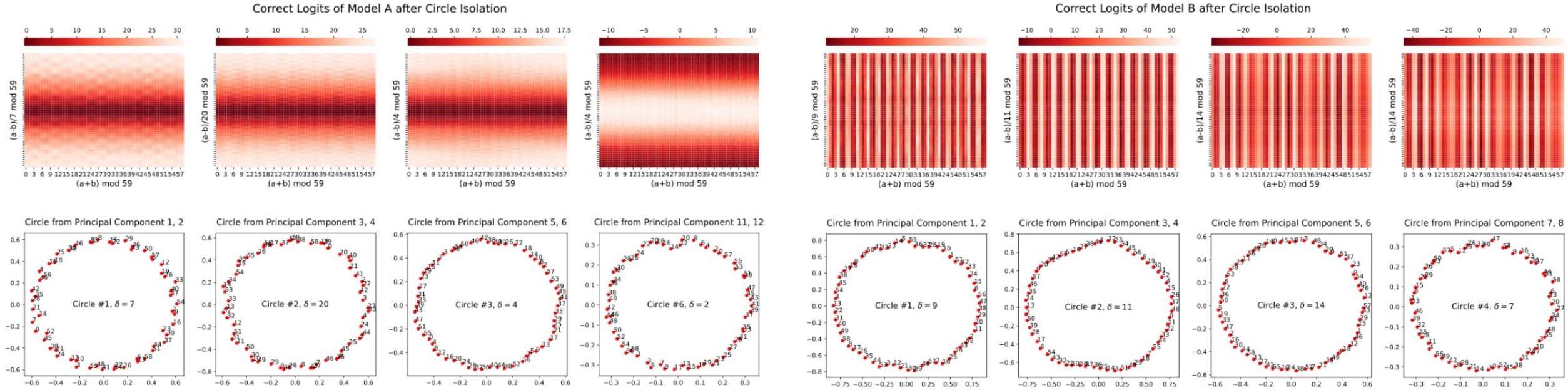
- we replace the embedding matrix $\boldsymbol{E}$ with a series of rank-2 approximations: using only the first and second principal components, or only the third and fourth, etc. For each such matrix, embeddings lie in a a two-dimensional subspace.
- For both Model A and Model B, we find that embeddings form a circle in this subspace.

"Circle Isolation"



Second Evidence for *Pizza*: Clearer Logit Patterns via Circle Isolation

- Both Model A and Model B **continue to behave in interpretable ways**: a subset of predictions remain highly accurate, **with this subset determined by the periodicity of the k of the isolated circle**.
- Model A's accuracy drops to zero at specific values of $a - b$, while Model B's accuracy is invariant in $a - b$.



Third Evidence for *Pizza:* Accompanied & Accompanying Pizza

- If two inputs (a, b) happen to lie antipodally, then their middle point will lie at the origin, where the correct "pizza slice" is difficult to identify.
- idea: adjacent distance from k → 2k

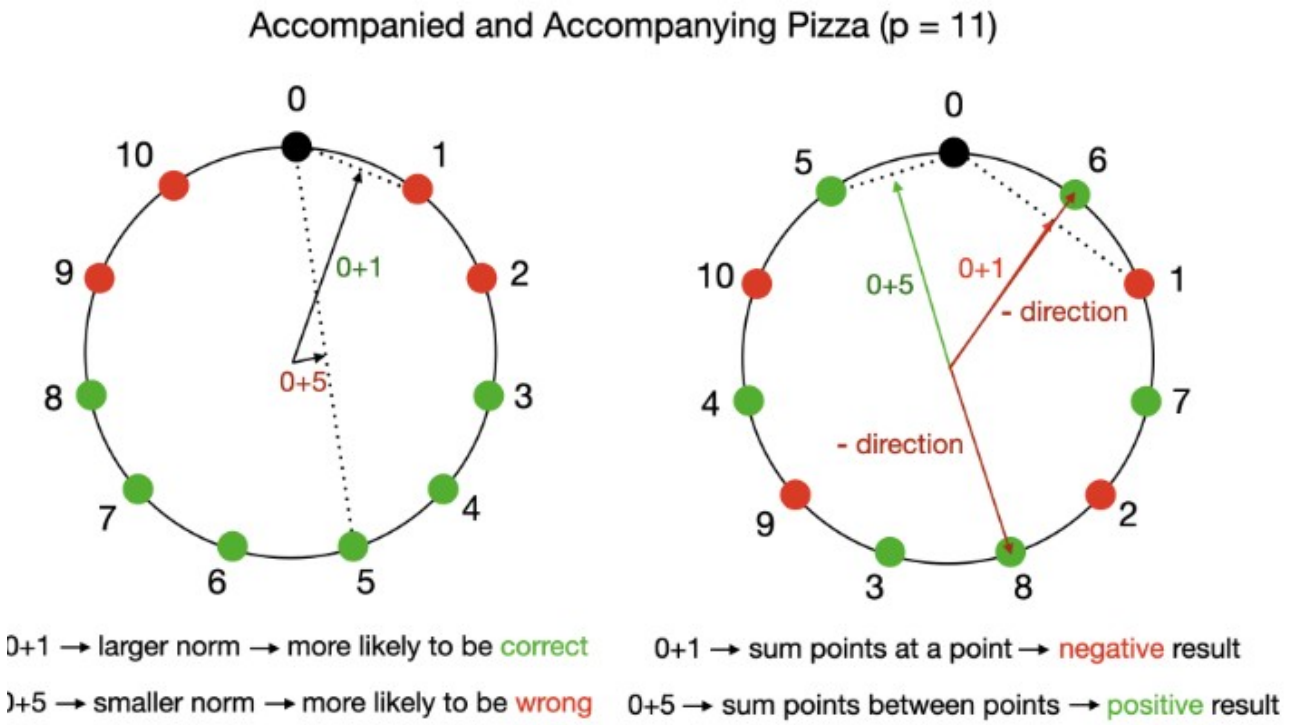
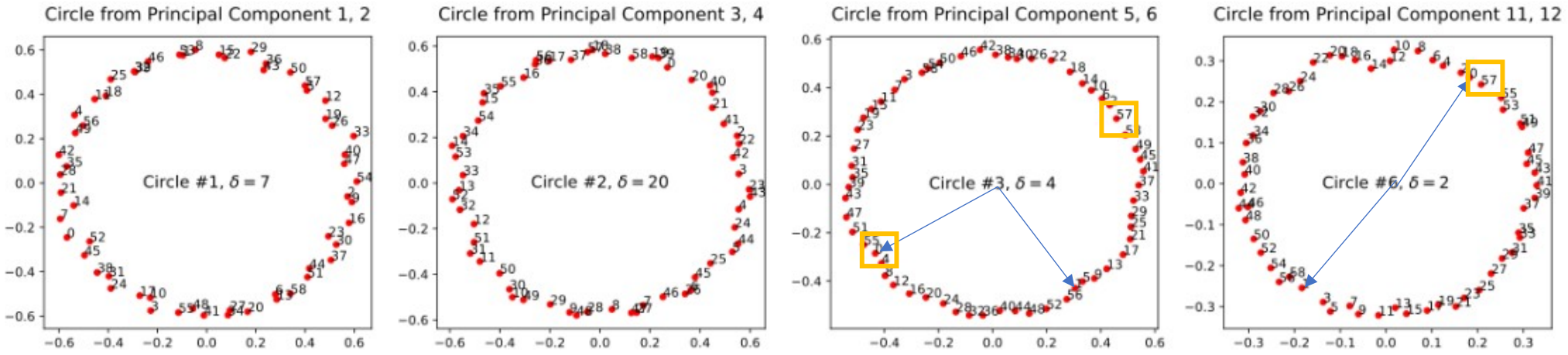


Figure 11: An Illustration on the Accompanying Pizza Algorithm

Third Evidence for *Pizza:* Accompanied & Accompanying Pizza

- In the experiment, we found that pizzas #1/#2/#3 all have accompanying pizzas, which we call pizzas #4/#5/#6.
- However, these accompanying pizzas do not play a significant role in final model predictions.
  - Accompanied pizzas #1/#2/#3 can achieve 99.7% accuracy, but accompanying pizzas #4/#5/#6 can only achieve 16.7% accuracy.



The Algorithmic Phase Space

How to distinguish Pizza from Clock?

- Metrics
  - Gradient Symmetricity:
    - clock requires $\nabla_{E_a} Q_{abc} \neq \nabla_{E_b} Q_{abc}$, pizza requires $\nabla_{E_a} Q_{abc} = \nabla_{E_b} Q_{abc}$; clock s_g small, pizza s_g big

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

  - Distance Irrelevance
    - measure the dependence of correct logits on differences between two inputs
    - clock does not depend on $b-a$, pizza depend on $b-a$; clock q big, pizza q small

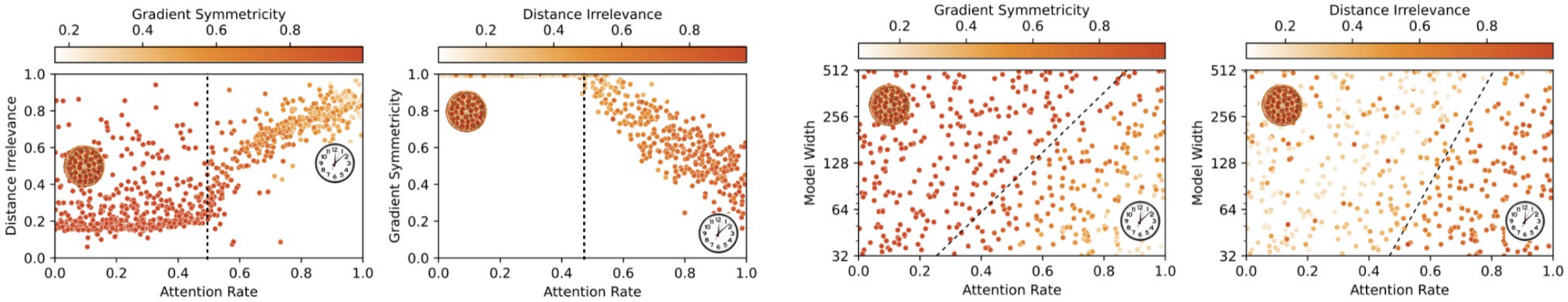
$$q \equiv \frac{\frac{1}{p}\sum_{d\in\mathbb{Z}_p} \mathrm{std}\left(L_{i,i+d} \mid i \in \mathbb{Z}_p^2\right)}{\mathrm{std}\left(L_{i,j} \mid i,j \in \mathbb{Z}_p^2\right)},$$

Phase Transition Results

- To interpolating between Model A (transformer without attention) and Model B (transformer with attention → attention rate α
- For a model with attention rate α, we modify the attention matrix M for each attention head to be $M' = M\alpha + I(1 - \alpha)$.

Phase Transition Results

- The Pizza and the Clock algorithms are the dominating algorithms with circular embeddings.
- Two-dimensional phase change observed for attention rate and layer width.
- Dominance of linear layers determines whether the Pizza or the Clock algorithm is preferred.
  - The **Clock** algorithm dominates when the **attention rate is higher** than the phase change point, and the **Pizza** algorithm dominates when the **attention rate is lower** than the point.
  - The **phase change point** gets **higher** when the **model width** increases.
  - **explanation**: When the model gets wider, the linear layers become more capable while the attention mechanism receive less benefit (attentions remain scalars while outputs from linear layers become wider vectors). The linear layer therefore gets more prominence with a wider model.



Conclusion

- Progress Measures for Grokking via Mechanistic Interpretability
  - how clock algorithm is implemented
  - **inspiration:** look for periodicity structure in model weights / activations for circular tasks
  - grokking comes with weight decay
- The Clock and the Pizza: Two Stories in Mechanistic Explanation of Neural Networks
  - clock algorithm is not the only solution
  - neural networks can shift from one solution to another as hyperparameters change

Future directions

- more realistic settings?
  - tasks – beyond modular addition
  - transformer with tokenization
  - transformer with multi-layer architecture