Mechanistic Interpretability on Irreducible Integer Identifiers

Noah Syrkis

January 8, 2025

1 | Mechanistic Interpretability (MI)

2 | Modular Arithmetic

3 | Grokking on $\mathcal{T}_{\text{miiii}}$

4 | Embeddings

5 | Neurons

6 | The ω-Spike

"This disgusting pile of matrices is actually just an astoundingly poorly written, elegant and consice algorithm" – Neel Nanda[1]

[1]Not verbatim, but the gist of it

1 | Mechanistic Interpretability (MI)

- Sub-symbolic nature of deep learning obscures model mechanisms

1 | Mechanistic Interpretability (MI)

- Sub-symbolic nature of deep learning obscures model mechanisms
- No obvious mapping from the weights of a trained model to math notation

1 | Mechanistic Interpretability (MI)

- Sub-symbolic nature of deep learning obscures model mechanisms
- No obvious mapping from the weights of a trained model to math notation
- MI is about reverse engineering these models, and looking closely at them

1 | Mechanistic Interpretability (MI)

- Sub-symbolic nature of deep learning obscures model mechanisms
- No obvious mapping from the weights of a trained model to math notation
- MI is about reverse engineering these models, and looking closely at them
- How does a given model work? How can we train it faster? Is it safe?

1 | Mechanistic Interpretability (MI)

- Early MI work focus on modular addition[1]

1 | Mechanistic Interpretability (MI)

- Early MI work focus on modular addition[1]
- $\mathcal{T}_{\text{nanda}}$ focus on a model mapping $(x_0, x_1) \to y$

1 | Mechanistic Interpretability (MI)

- Early MI work focus on modular addition[1]
- $\mathcal{T}_{\text{nanda}}$ focus on a model mapping $(x_0, x_1) \rightarrow y$
- True mapping given by $y = x_0 + x_1 \bmod p$

1 | Mechanistic Interpretability (MI)

(0,0)	(1,0)	(2,0)
(0,1)	(1,1)	(2,1)
(0,2)	(1,2)	(2,2)

Table 1: Table of (x_0, x_1)-tuples for $p = 3$

- Early MI work focus on modular addition[1]
- $\mathcal{T}_{\text{nanda}}$ focus on a model mapping $(x_0, x_1) \to y$
- True mapping given by $y = x_0 + x_1 \bmod p$

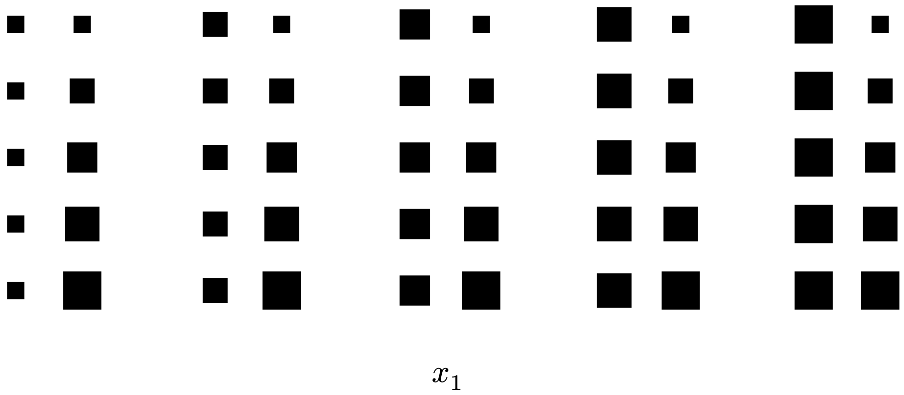

Figure 1: esch of (x_0, x_1)-tuples for $p = 5$

- on y from $\mathcal{T}_{\text{nanda}}$

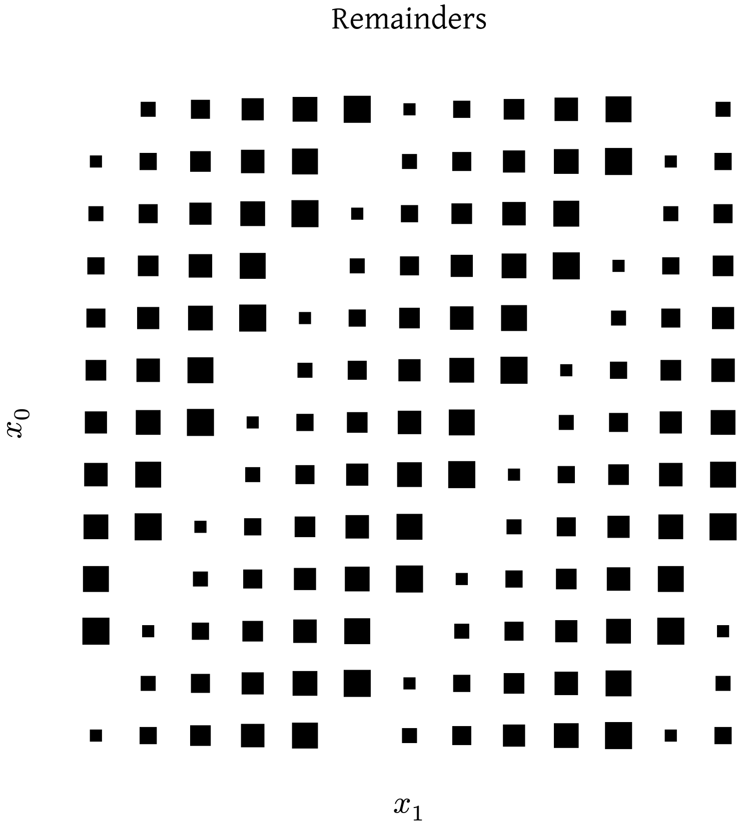

Figure 2: esch diagram of y from $\mathcal{T}_{\text{nanda}}$

1 | Mechanistic Interpretability (MI)

- Array

7	2	11	4	9	1	8	2	10	6	3	10	5
3	8	1	7	12	5	2	9	11	4	0	6	10
11	4	9	2	6	0	7	3	8	2	1	11	10
5	10	3	8	1	12	4	7	2	9	1	10	6
12	6	0	11	4	8	1	5	10	3	7	2	9
2	9	7	0	11	3	12	6	4	8	10	1	5
8	1	12	5	10	7	0	11	9	2	6	4	3
4	11	6	9	3	2	10	1	7	0	12	8	5
10	5	2	12	7	9	3	0	6	1	8	11	4
6	12	8	3	0	11	5	4	1	10	2	9	7
1	7	4	10	8	6	9	2	12	5	11	3	0
9	3	10	6	2	4	11	8	5	7	0	12	1
0	8	5	1	11	10	6	12	3	9	4	7	2

1 | Mechanistic Interpretability (MI)

- Are hard to see

1 | Mechanistic Interpretability (MI)

1. Make a task
2. Solve the task
3. Inspect the solution

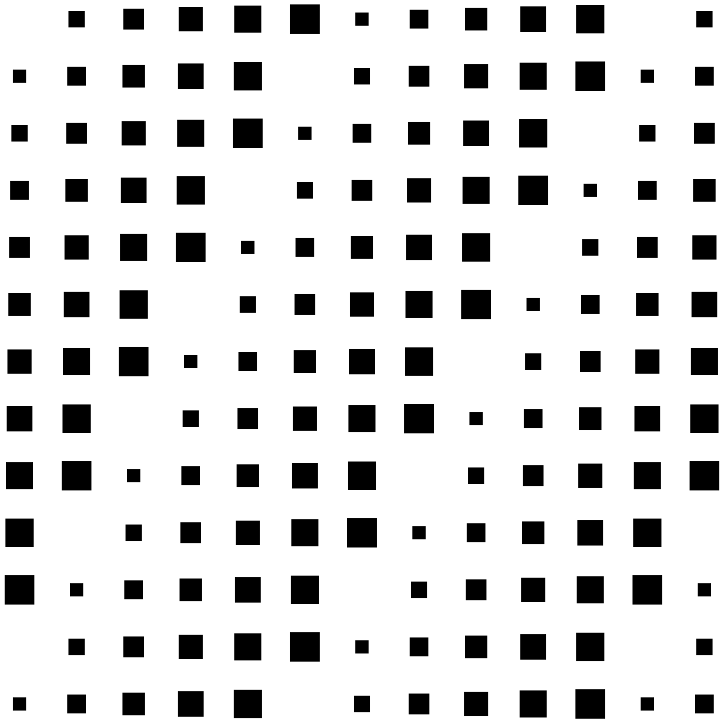

Figure 3: Target y for as x_0 and x_1 move from 0 to $p - 1$ for the task $x_0 + x_1 \bmod p = y$

1 | Mechanistic Interpretability (MI)

1. Make a task
2. Solve the task
3. Inspect the solution

- Think artificial neuroscience

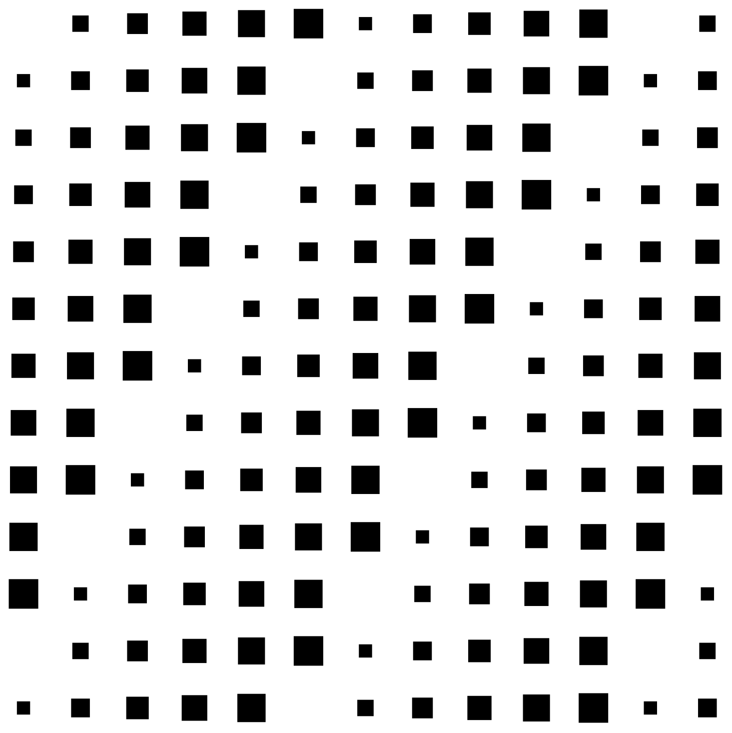

Figure 3: Target y for as x_0 and x_1 move from 0 to $p - 1$ for the task $x_0 + x_1 \bmod p = y$

1.1 | Grokking [1]

- Sudden generalization long after overfitting

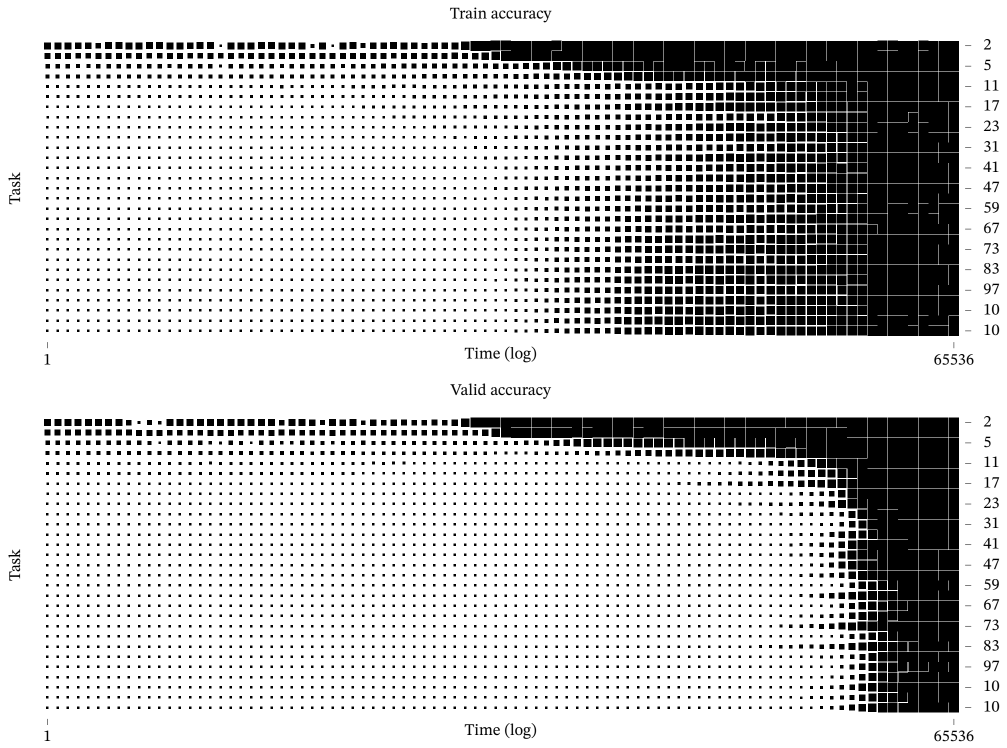

Figure 4: Example of the grokking

1.1 | Grokking [1]

- Sudden generalization long after overfitting
- MI (by definition) needs a mechanism

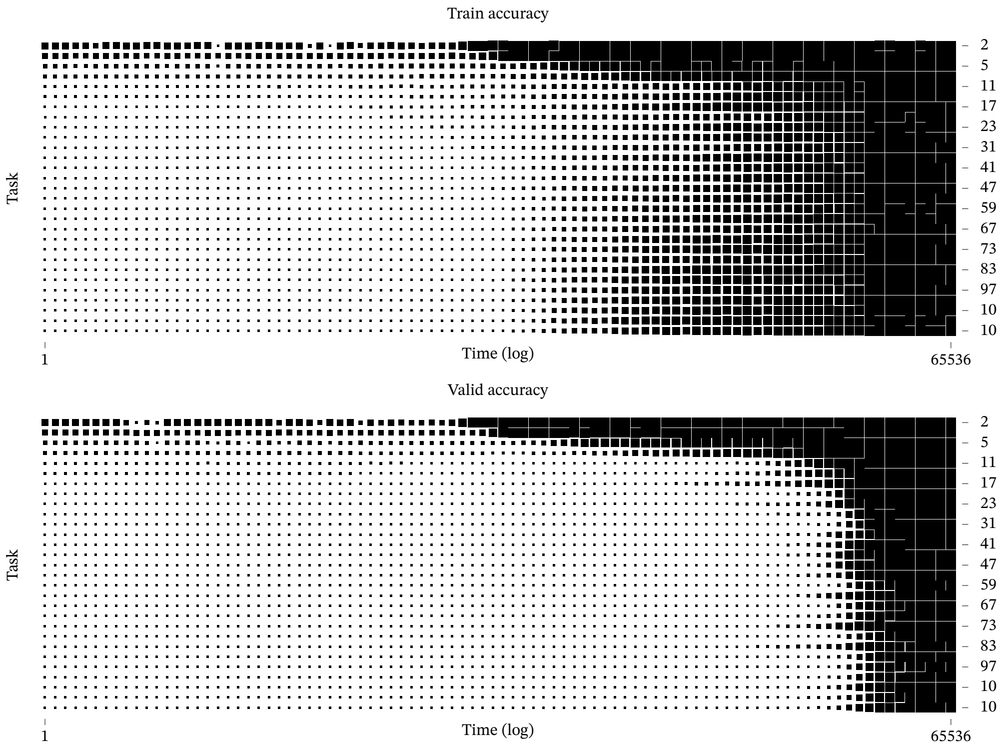

Figure 4: Example of the grokking

1.1 | Grokking [1]

- Sudden generalization long after overfitting
- MI (by definition) needs a mechanism
- Grokking is thus convenient for MI

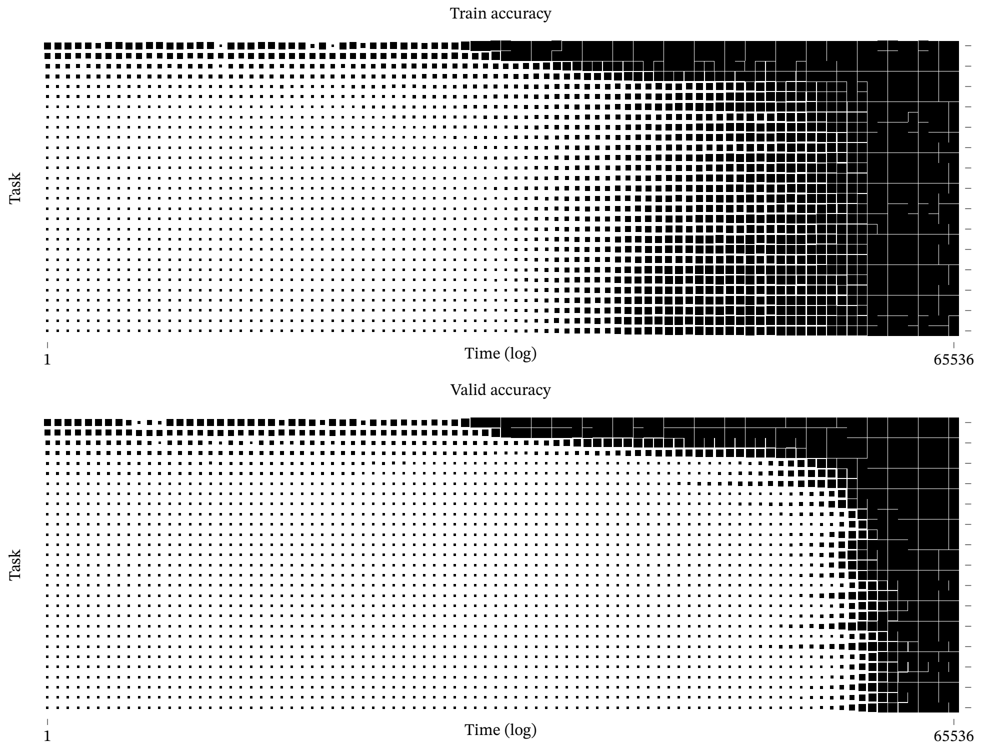

Figure 4: Example of the grokking

2 | Modular Arithmetic

- "Seminal" MI paper by Nanda et al. (2023) focuses on modular addition ($\mathcal{T}_{\text{nanda}}$)
- Their final setup trains on $p = 113$
- They train a one-layer transformer
- We call their task $\mathcal{T}_{\text{nanda}}$

$$\mathcal{T}_{\text{nanda}} = (x_0 + x_1) \bmod p, \forall x_0, x_1 \quad (1.1)$$

$$\mathcal{T}_{\text{miiii}} = \left(x_0 p^0 + x_1 p^1\right) \bmod q, \forall q < p \quad (1.2)$$

2 | Modular Arithmetic

- "Seminal" MI paper by Nanda et al. (2023) focuses on modular addition ($\mathcal{T}_{\text{nanda}}$)
- Their final setup trains on $p = 113$
- They train a one-layer transformer
- We call their task $\mathcal{T}_{\text{nanda}}$
- And ours we call $\mathcal{T}_{\text{miiii}}$

$$\mathcal{T}_{\text{nanda}} = (x_0 + x_1) \bmod p, \forall x_0, x_1 \quad (1.1)$$

$$\mathcal{T}_{\text{miiii}} = \left(x_0 p^0 + x_1 p^1\right) \bmod q, \forall q < p \quad (1.2)$$

2 | Modular Arithmetic

- $\mathcal{T}_{\text{miiii}}$ is non-commutative …
- … and multi-task: q ranges from 2 to 109[1]
- $\mathcal{T}_{\text{nanda}}$ use a single layer transformer
- Note that these tasks are synthetic and trivial to solve with conventional programming
- They are used in the MI literature to turn black boxes opaque

[1]Largest prime less than $p = 113$

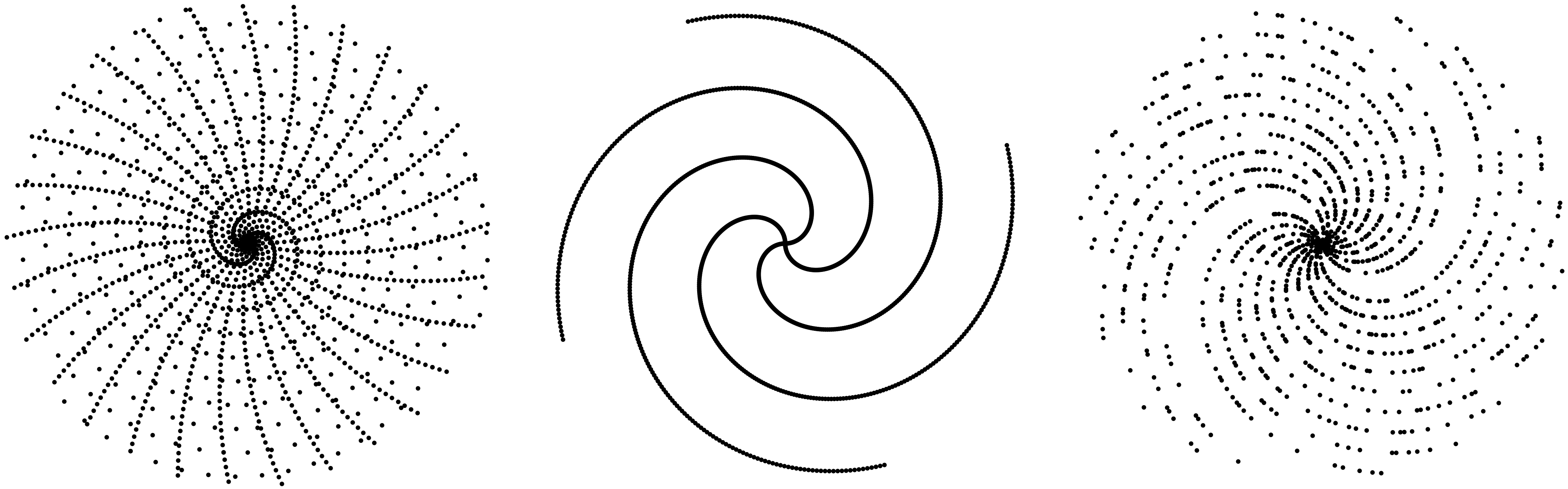

Figure 8: $\mathbb{N} < p^2$ multiples of 13 or 27 (left) 11 (mid.) or primes (right)

- For two-token samples, plot them varying one on each axis (Figure 9)
- When a matrix is periodic use Fourier
- Singular value decomposition

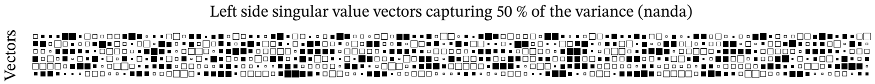

Figure 9: Top singular vectors of $\mathbf{U}_{W_{E_{\mathcal{T}_{\mathrm{nanda}}}}}$ (top), varying x_0 and x_1 in sample (left) and freq. (right) space in $W_{\mathrm{out}_{\mathcal{T}_{\mathrm{miiii}}}}$

3 | Grokking on $\mathcal{T}_{\text{miiii}}$

- The model groks on $\mathcal{T}_{\text{miiii}}$ (Figure 10)
- Needed GrokFast [2] on compute budget
- Final hyperparams are seen in Table 6

rate	λ	wd	d	lr	heads
$\frac{1}{10}$	$\frac{1}{2}$	$\frac{1}{3}$	256	$\frac{3}{10^4}$	4

Table 6: Hyperparams for $\mathcal{T}_{\text{miiii}}$

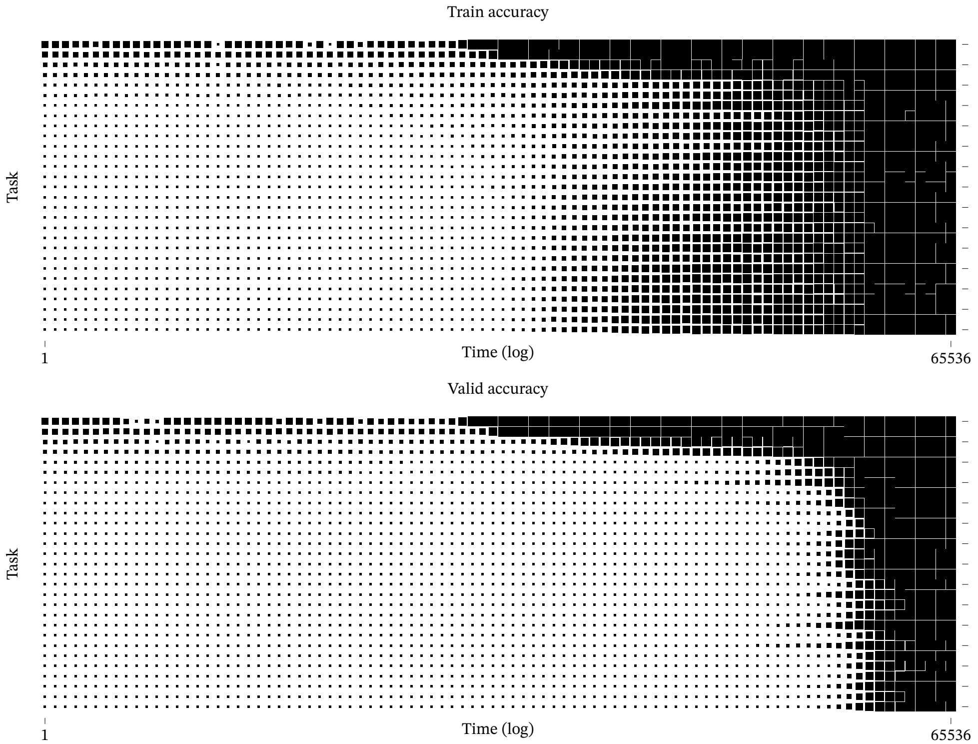

Figure 10: Training (top) and validation (bottom) accuracy during training on $\mathcal{T}_{\text{miiii}}$

4 | Embeddings

How the embedding layer deals with the difference between $\mathcal{T}_{\text{nanda}}$ and $\mathcal{T}_{\text{miiii}}$

- The position embs. of Figure 12 reflects that $\mathcal{T}_{\text{nanda}}$ is commutative and $\mathcal{T}_{\text{miiii}}$ is not

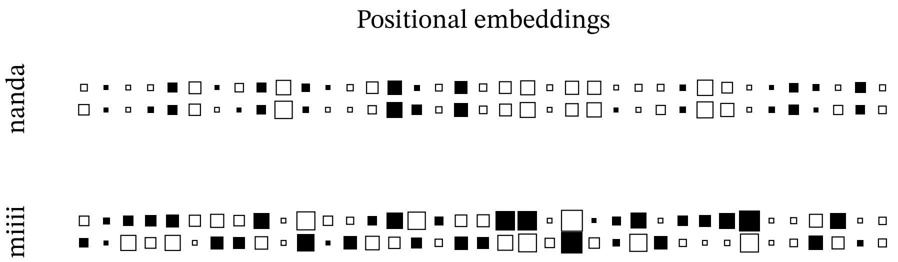

Figure 11: Positional embeddings for $\mathcal{T}_{\text{nanda}}$ (top) and $\mathcal{T}_{\text{miiii}}$ (bottom).

4.1 | Correcting for non-commutativity

- The position embs. of Figure 12 reflects that $\mathcal{T}_{\text{nanda}}$ is commutative and $\mathcal{T}_{\text{miiii}}$ is not
- Maybe: this corrects non-comm. of $\mathcal{T}_{\text{miiii}}$?
- Corr. is 0.95 for $\mathcal{T}_{\text{nanda}}$ and -0.64 for $\mathcal{T}_{\text{miiii}}$

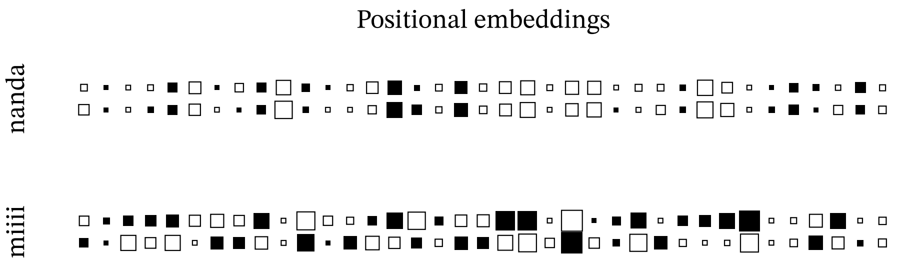

Figure 12: Positional embeddings for $\mathcal{T}_{\text{nanda}}$ (top) and $\mathcal{T}_{\text{miiii}}$ (bottom).

4.2 | Correcting for multi-tasking

- For $\mathcal{T}_{\text{nanda}}$ token embs. are essentially linear combinations of 5 frequencies (ω)
- For $\mathcal{T}_{\text{miiii}}$ more frequencies are in play
- Each $\mathcal{T}_{\text{miiii}}$ subtask targets unique prime
- Possibility: One basis per prime task

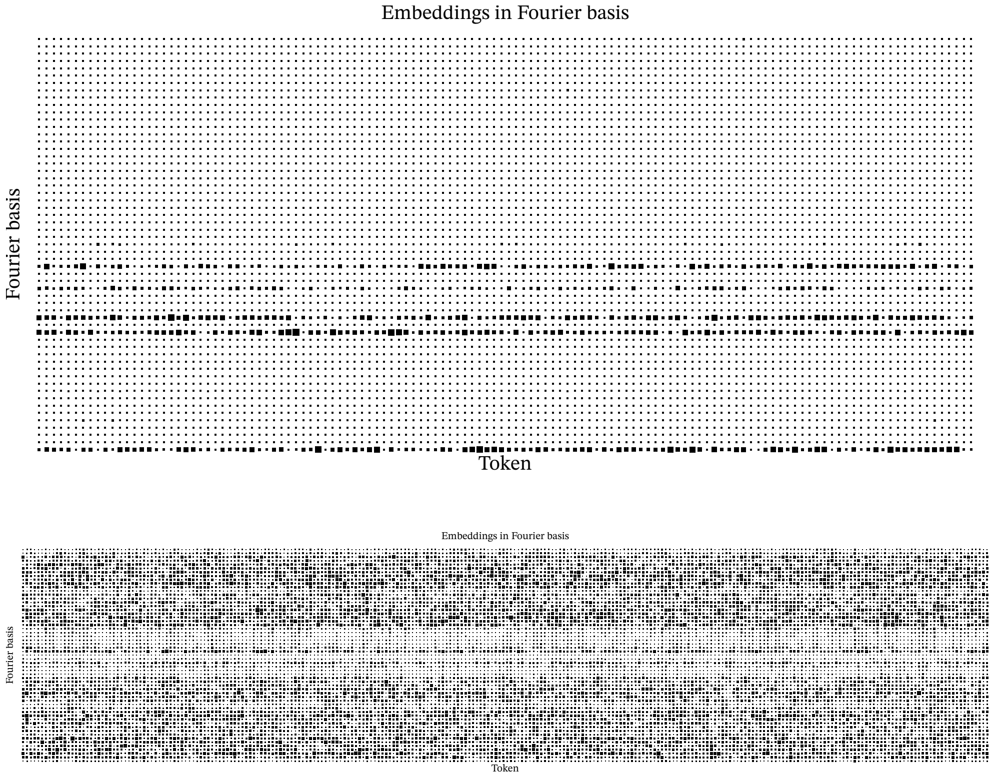

Figure 13: $\mathcal{T}_{\text{nanda}}$ (top) and $\mathcal{T}_{\text{miiii}}$ (bottom) token embeddings in Fourier basis

4.3 | Sanity-check and task-mask

- Masking $q \in \{2, 3, 5, 7\}$ yields we see a slight decrease in token emb. freqs.
- Sanity check: $\mathcal{T}_{\text{baseline}}$ has no periodicity
- The tok. embs. encode a basis per subtask?

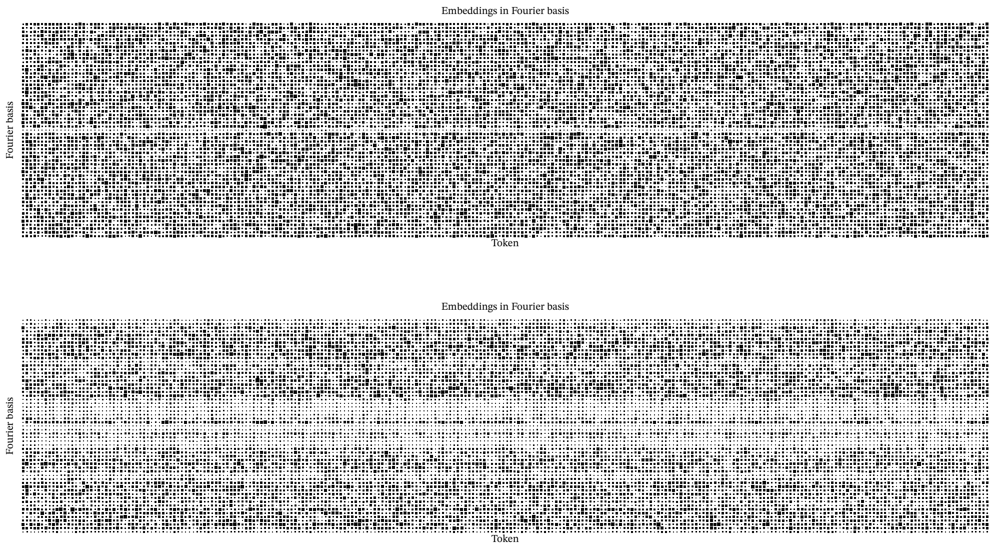

Figure 14: $\mathcal{T}_{\text{baseline}}$ (top), $\mathcal{T}_{\text{miiii}}$ (middle) and $\mathcal{T}_{\text{masked}}$ (bottom) token embeddings in Fourier basis

5 | Neurons

- Inspite of the dense Fourier basis of $W_{E_{\mathcal{T}_{\text{miiii}}}}$ the periodicity is clear

figs/neurs_113_miiii.svg

), caption: [Activations of first three neurons for $\mathcal{T}_{\text{nanda}}$ (top) and $\mathcal{T}_{\text{miiii}}$ (bottom)],)

5 | Neurons

- (Probably redundant) sanity check: Figure 16 confirms neurons are periodic
- See some freqs. ω rise into significance
- Lets log $|\omega > \mu_\omega + 2\sigma_\omega|$ while training

Figure 16: FFT of Activations of first three neurons for $\mathcal{T}_{\text{nanda}}$ (top) and $\mathcal{T}_{\text{miiii}}$ (bottom)

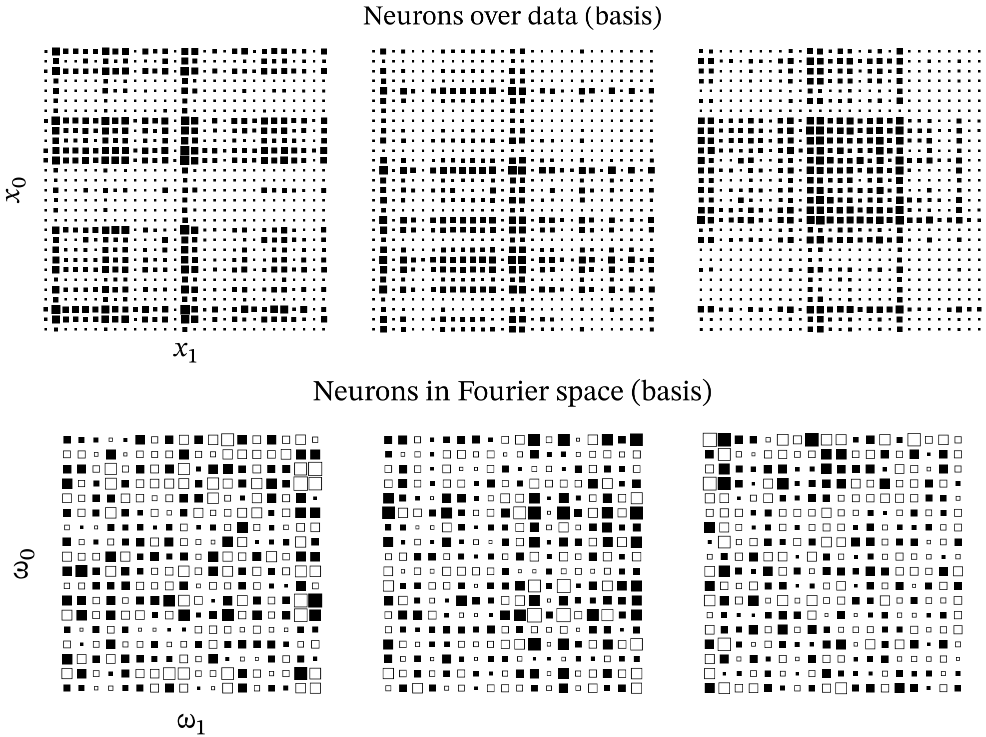

Figure 17: Neurons as archive for $\mathcal{T}_{\text{basline}}$

Figure 18: Neurons as algorithm $\mathcal{T}_{\text{miiii}}$

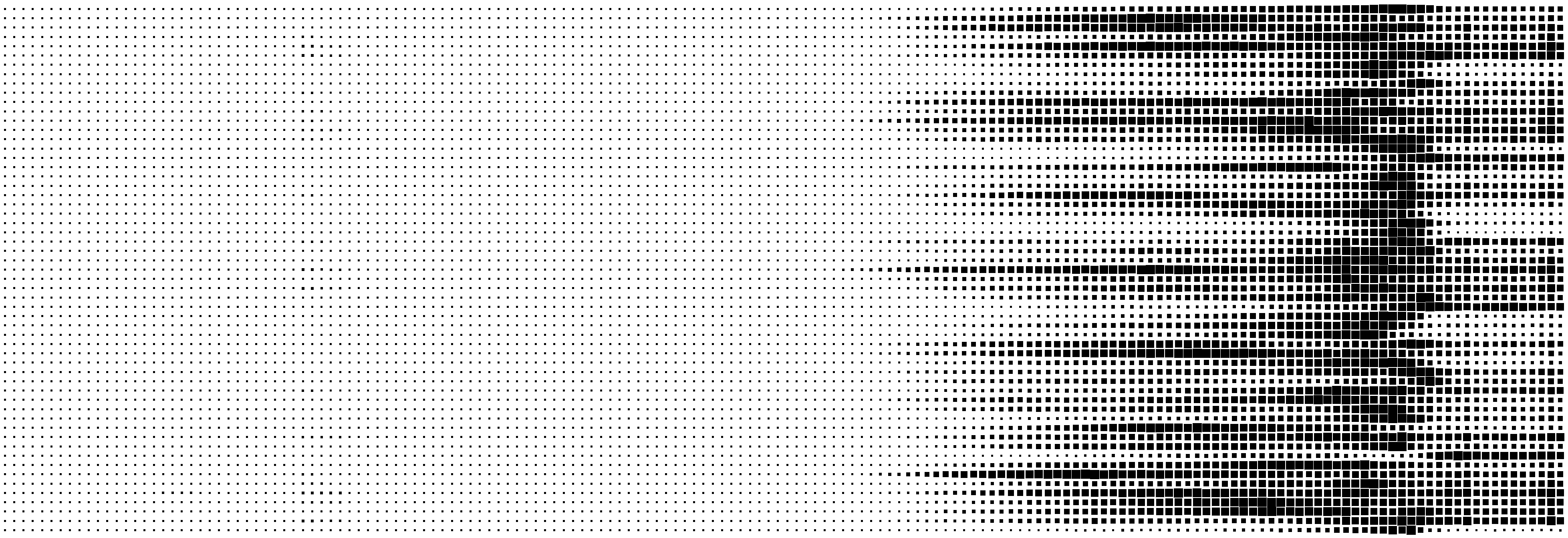

Figure 19: Number of neurons with frequency ω above the theshold $\mu_\omega + 2\sigma_\omega$

6 | The ω-Spike

- Neurs. periodic on solving $q \in \{2, 3, 5, 7\}$
- When we generalize to the reamining tasks, many frequencies activate (64-sample)
- Those ω's are not useful for memory and not useful after generalization

time	256	1024	4096	16384	65536
$\lvert\omega\rvert$	0	0	10	18	10

Table 7: active ω's through training

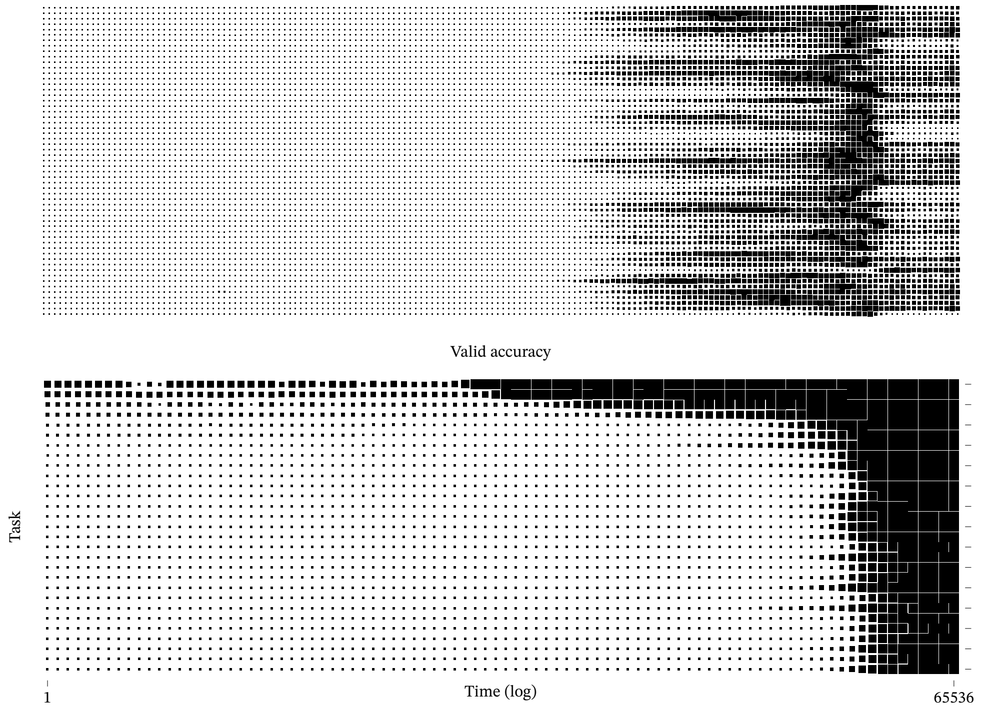

Figure 20: Figure 19 (top) and validation accuracy from Figure 10 (bottom)

6 | The ω-Spike

- GrokFast [2] shows time gradient sequences is (arguably) a stocastical signal with:
  - A fast varying overfitting component
  - A slow varying generealizing component
- My work confirms this to be true for $\mathcal{T}_{\text{miiii}}$ …
- … and observes a strucutre that seems to fit neither of the two

6 | The ω-Spike

- Future work:
  - Modify GrokFast to assume a third stochastic component
  - Relate to signal processing literature
  - Can more depth make tok-embedding sparse?

References

[1] N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt, "Progress Measures for Grokking via Mechanistic Interpretability," no. arXiv:2301.05217. arXiv, Oct. 2023.

[2] J. Lee, B. G. Kang, K. Kim, and K. M. Lee, "Grokfast: Accelerated Grokking by Amplifying Slow Gradients," no. arXiv:2405.20233. Jun. 2024.

A | Stochastic Signal Processing

We denote the weights of a model as θ. The gradient of θ with respect to our loss function at time t we denote $g(t)$. As we train the model, $g(t)$ varies, going up and down. This can be thought of as a stocastic signal. We can represent this signal with a Fourier basis. GrokFast posits that the slow varying frequencies contribute to grokking. Higer frequencies are then muted, and grokking is indeed accelerated.

B | Discrete Fourier Transform

Function can be expressed as a linear combination of cosine and sine waves. A similar thing can be done for data / vectors.

C | Singular Value Decomposition

An $n \times m$ matrix M can be represented as a $U\Sigma V^*$, where U is an $m \times m$ complex unitary matrix, Σ a rectangular $m \times n$ diagonal matrix (padded with zeros), and V an $n \times n$ complex unitary matrix. Multiplying by M can thus be viewed as first rotating in the m-space with U, then scaling by Σ and then rotating by V in the n-space.