Mechanistic Interpretability on (multi-task) Irreducible Integer Identifiers

Noah Syrkis

January 20, 2025

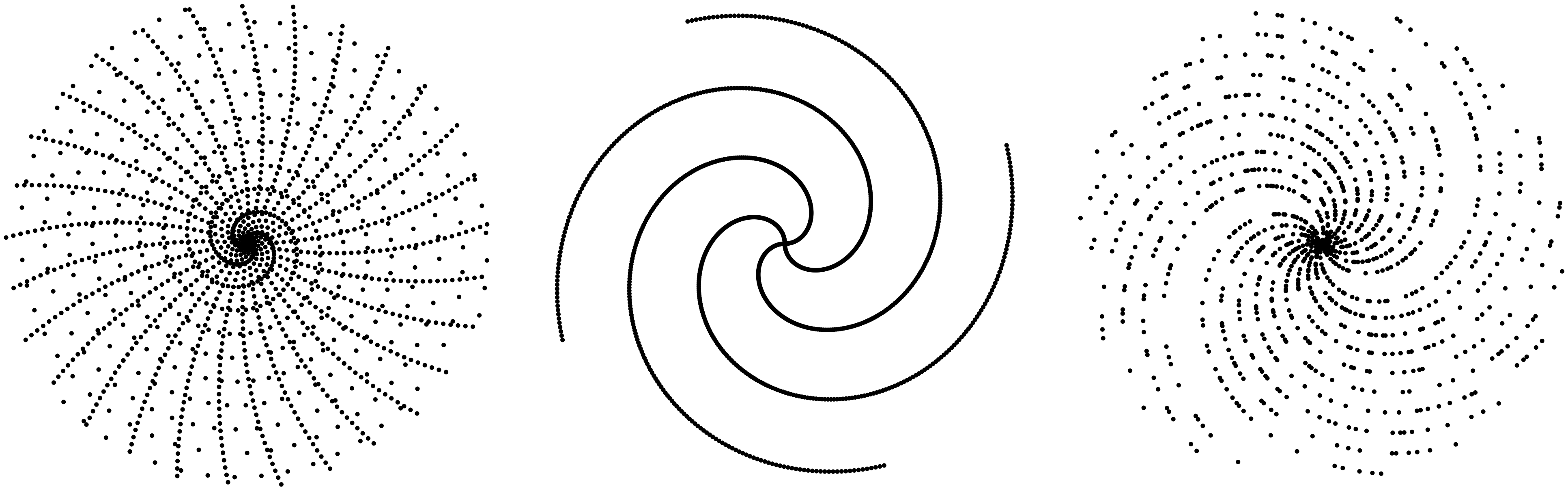

Figure 1: $\mathbb{N} < p^2$ *multiples of 13 or 27 (left) 11 (mid.) or primes (right)*

“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
- Many low hanging fruits / practical botany phase of the science

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
- Many low hanging fruits / practical botany phase of the science
- How does a given model work? How can we train it faster? Is it safe?

1.1 | Grokking

- Grokking [1] is "sudden generalization"

$$h(t) = h(t-1)\alpha + g(t)(1-\alpha) \tag{1.1}$$

$$\hat{g}(t) = g(t) + \lambda h(t) \tag{1.2}$$

1.1 | Grokking

- Grokking [1] is "sudden generalization"
- MI (often) needs a mechanism

$$h(t) = h(t-1)\alpha + g(t)(1-\alpha) \tag{1.1}$$

$$\hat{g}(t) = g(t) + \lambda h(t) \tag{1.2}$$

1.1 | Grokking

- Grokking [1] is “sudden generalization”
- MI (often) needs a mechanism
- Grokking is thus convenient for MI

$$h(t) = h(t-1)\alpha + g(t)(1-\alpha) \tag{1.1}$$

$$\hat{g}(t) = g(t) + \lambda h(t) \tag{1.2}$$

1.1 | Grokking

- Grokking [1] is “sudden generalization”
- MI (often) needs a mechanism
- Grokking is thus convenient for MI
- Lee et al. (2024) speeds up grokking by boosting slow gradients as per Eq. 1
- For more see Appendix A

$$h(t) = h(t-1)\alpha + g(t)(1-\alpha) \tag{1.1}$$

$$\hat{g}(t) = g(t) + \lambda h(t) \tag{1.2}$$

1.2 | Visualizing

- MI needs creativity ...

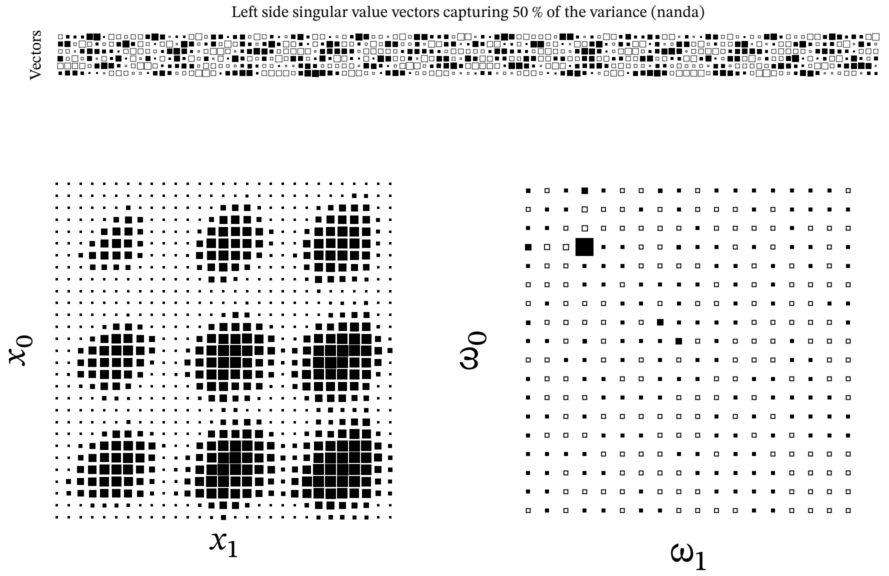

Figure 2: Top singular vectors of $\mathbf{U}_{W_{E_{\mathcal{T}_{\text{nanda}}}}}$ *(top), varying* x_0 *and* x_1 *in sample (left) and freq. (right) space in* $W_{\text{out}_{\mathcal{T}_{\text{miiii}}}}$

1.2 | Visualizing

- MI needs creativity ... but there are tricks:

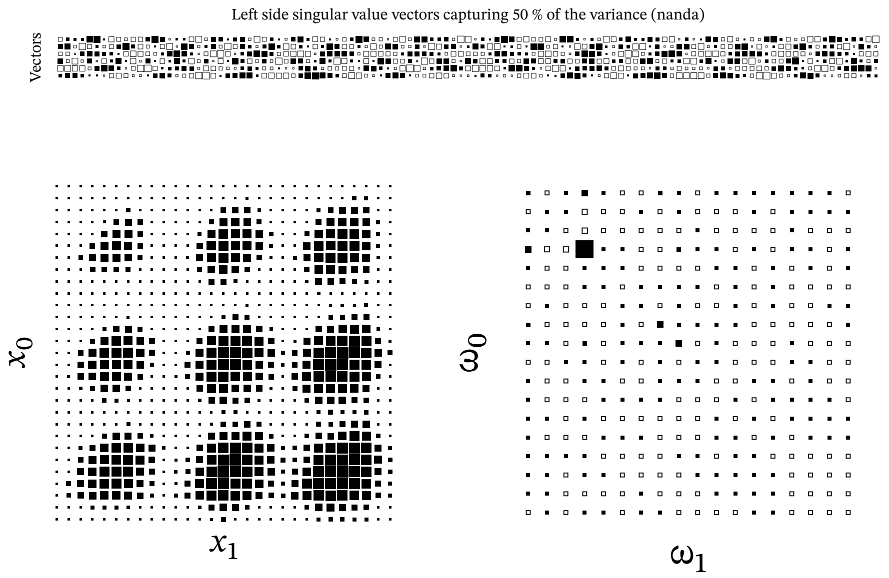

Figure 2: Top singular vectors of $\mathbf{U}_{W_{E_{\mathcal{T}_{\text{nanda}}}}}$ *(top), varying* x_0 *and* x_1 *in sample (left) and freq. (right) space in* $W_{\text{out}_{\mathcal{T}_{\text{miiii}}}}$

1.2 | Visualizing

- MI needs creativity … but there are tricks:
  - For two-token samples, plot them varying one on each axis (Figure 2)
  - When a matrix is periodic use Fourier
  - Singular value decomp. (Appendix C).
  - Take away: get commfy with `esch`-plots

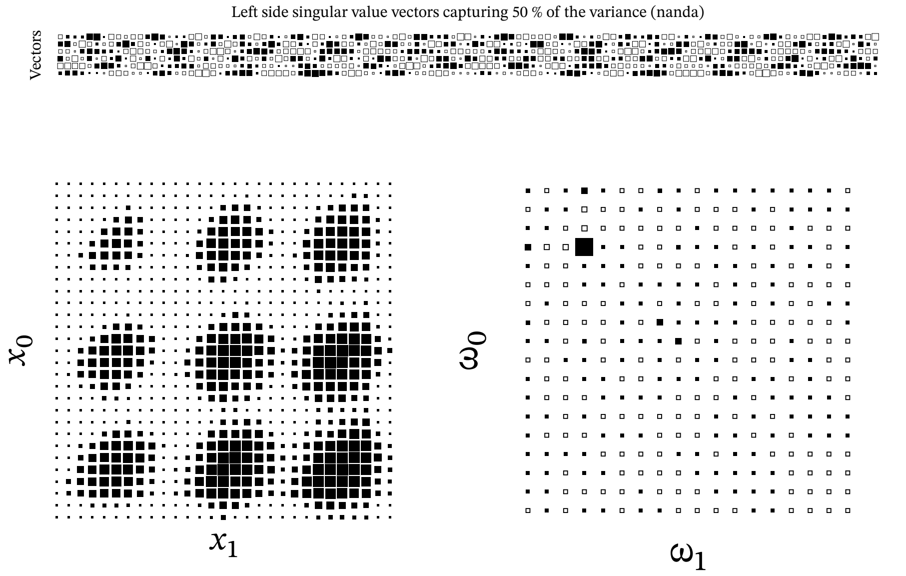

Figure 2: Top singular vectors of $\mathbf{U}_{W_{E_{\mathcal{T}_{\text{nanda}}}}}$ (top), varying x_0 and x_1 in sample (left) and freq. (right) space in $W_{\text{out}_{\mathcal{T}_{\text{miiii}}}}$

2 | Modular Arithmetic

- “Seminal” MI paper by Nanda et al. (2023) focuses on modular additon (Eq. 2)
- Their final setup trains on $p = 113$
- They train a one-layer transformer
- We call their task $\mathcal{T}_{\text{nanda}}$

$$(x_0 + x_1) \bmod p, \quad \forall x_0, x_1 \tag{2}$$

2 | Modular Arithmetic

- "Seminal" MI paper by Nanda et al. (2023)

  focuses on modular additon (Eq. 2)
- Their final setup trains on $p = 113$
- They train a one-layer transformer
- We call their task $\mathcal{T}_{\text{nanda}}$
- And ours, seen in Eq. 3, we call $\mathcal{T}_{\text{miiii}}$

$$(x_0 + x_1) \bmod p, \quad \forall x_0, x_1 \tag{2}$$

$$(x_0 p^0 + x_1 p^1) \bmod q, \quad \forall q < p \tag{3}$$

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 $p = 113$

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

- The model groks on $\mathcal{T}_{\text{miiii}}$ (Figure 3)
- Needed GrokFast [2] on compute budget
- Final hyperparams are seen in Table 1

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

Table 1: Hyperparams for $\mathcal{T}_{\text{miiii}}$

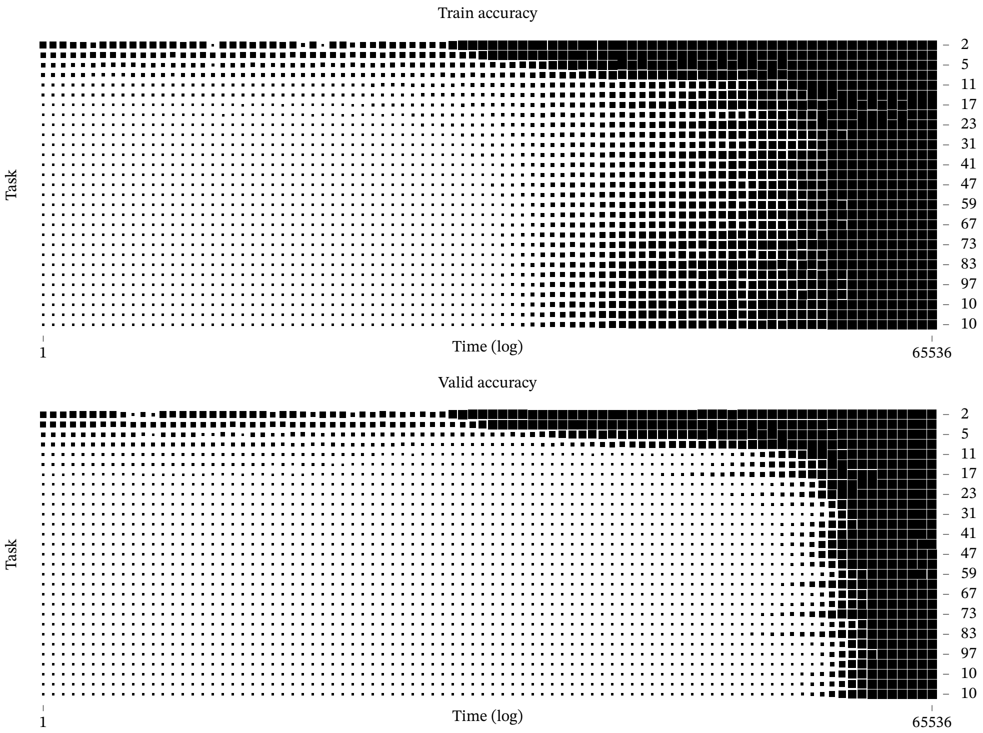

Figure 3: Training (top) and validation (bottom) accuracy during training on $\mathcal{T}_{\text{miiii}}$

4 | Embeddings

- The position embs. of Figure 4 reflects that $\mathcal{T}_{\text{nanda}}$ is commutative and $\mathcal{T}_{\text{miiii}}$ is not

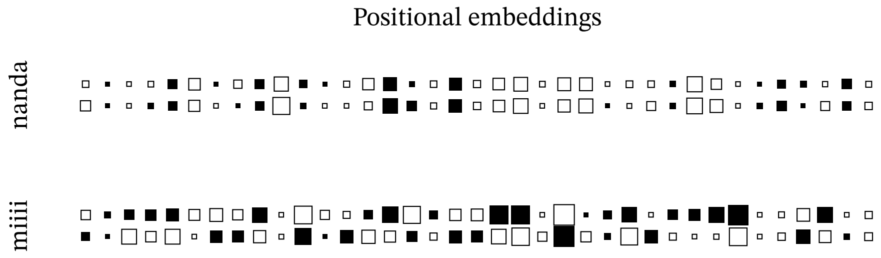

Figure 4: Positional embeddings for $\mathcal{T}_{\text{nanda}}$ *(top) and* $\mathcal{T}_{\text{miiii}}$ *(bottom).*

4 | Embeddings

- The position embs. of Figure 4 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 $\mathcal{T}_{\text{miiii}}$

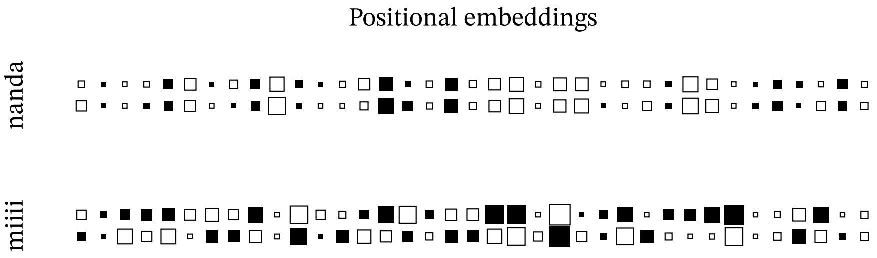

Figure 4: Positional embeddings for $\mathcal{T}_{\text{nanda}}$ (top) and $\mathcal{T}_{\text{miiii}}$ (bottom).

4 | Embeddings

- 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

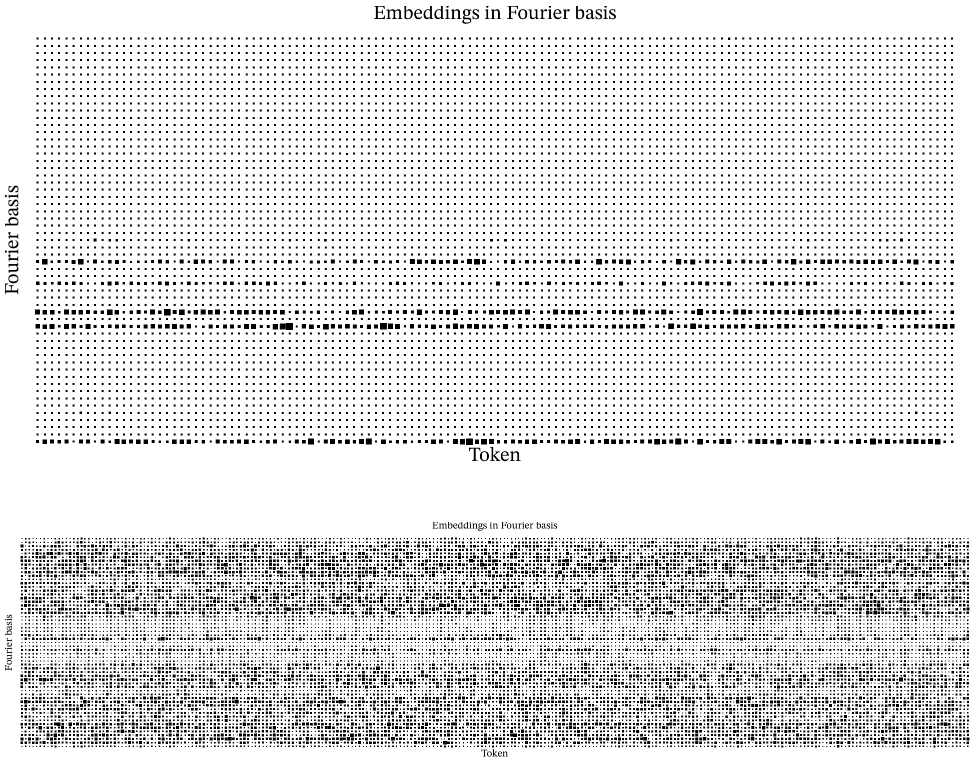

Figure 5: $\mathcal{T}_{\text{nanda}}$ (top) and $\mathcal{T}_{\text{miiii}}$ (bottom) token embeddings in Fourier basis

4 | Embeddings

- Masking $q \in \{2, 3, 5, 7\}$ yields we see a slight decrease in token emb. freqs.

4 | Embeddings

- 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?

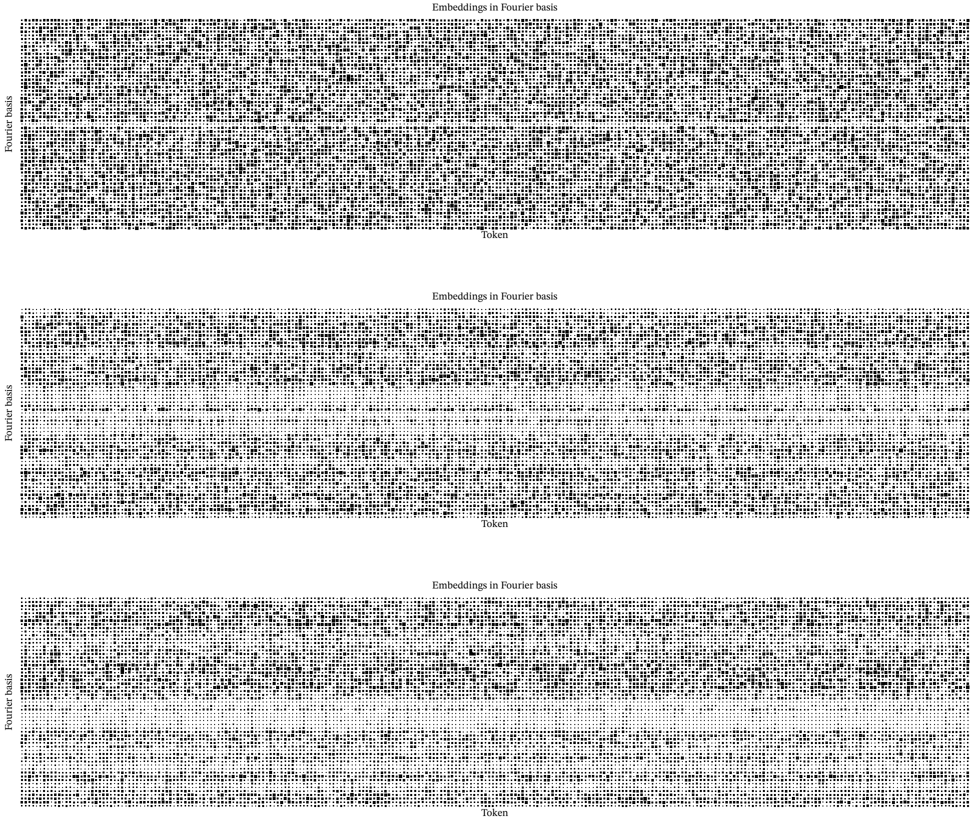

Figure 8: $\mathcal{T}_{\text{baseline}}$ (top), $\mathcal{T}_{\text{miiii}}$ (middle) and $\mathcal{T}_{\text{masked}}$ (bottom) token embeddings in Fourier basis

5 | Neurons

- Figure 9 shows transformer MLP neuron activations as x_0, x_1 vary on each axis
- Inspite of the dense Fourier basis of $W_{E_{\mathcal{T}_{\text{miiii}}}}$ the periodicity is clear

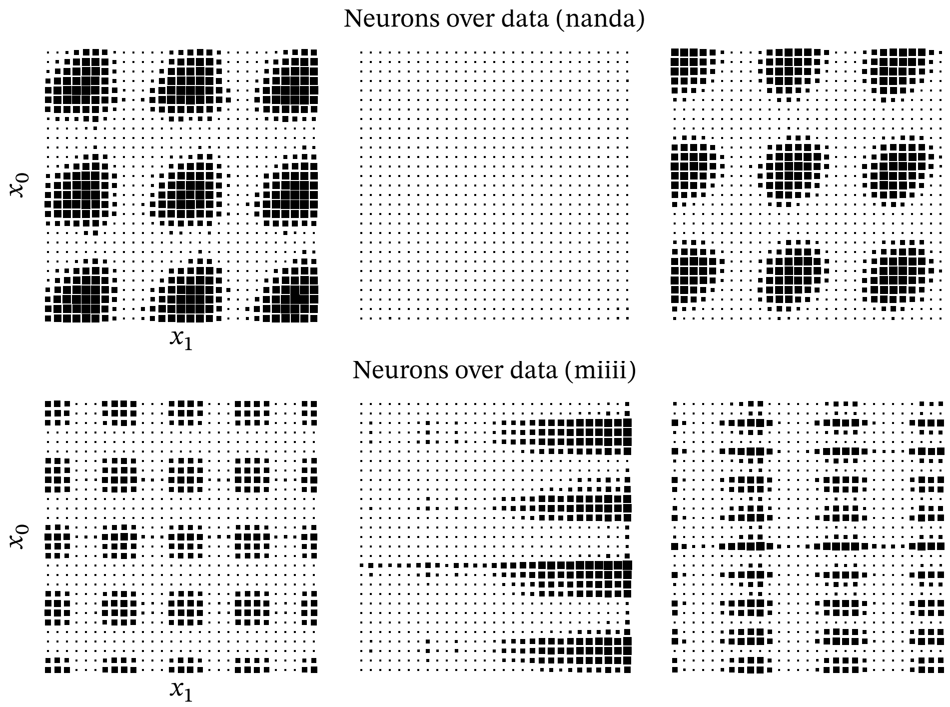

Figure 9: Activations of first three neurons for $\mathcal{T}_{\text{nanda}}$ (top) and $\mathcal{T}_{\text{miiii}}$ (bottom)

5 | Neurons

- (Probably redundant) sanity check:

  Figure 10 confirms neurons are periodic
- See some freqs. ω rise into significance
- Lets log $|\omega > \mu_\omega + 2\sigma_\omega|$ while training

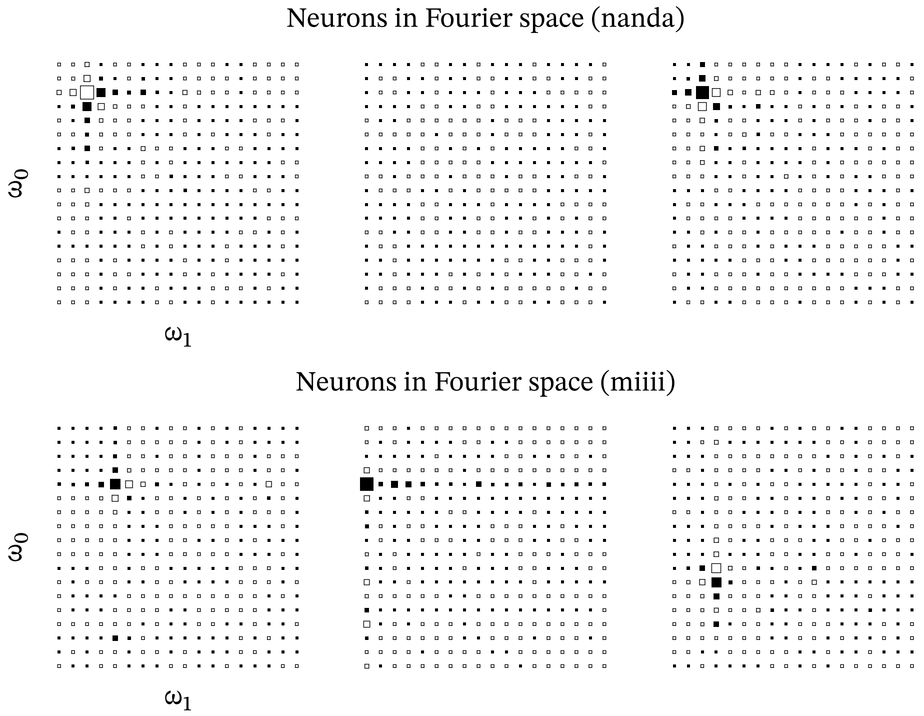

Figure 10: FFT of Activations of first three neurons for $\mathcal{T}_{\text{nanda}}$ (top) and $\mathcal{T}_{\text{miiii}}$ (bottom)

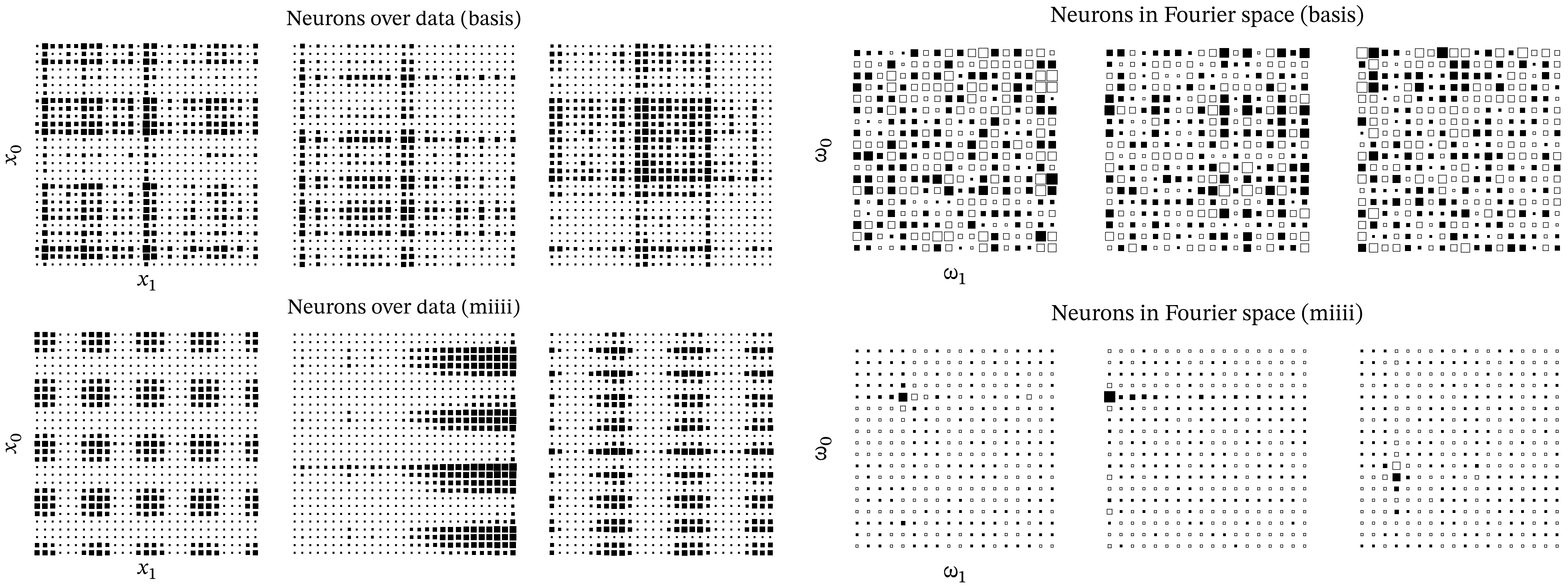

Figure 11: Neurons as archive and algorithm. $\mathcal{T}_{\text{basline}}$ on top, FFT on right.

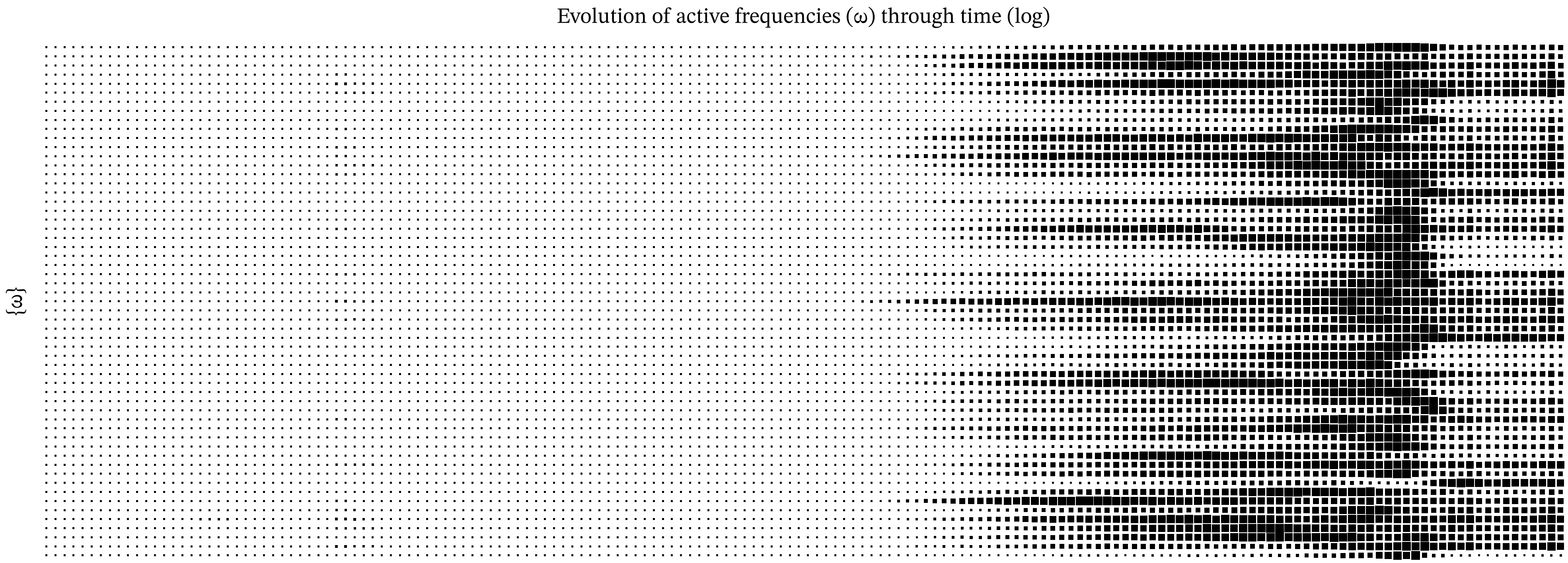

Figure 12: Number of neurons with frequency ω above the theshold $\mu_\omega + 2\sigma_\omega$

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
$\lvert\boldsymbol{\omega}\rvert$	0	0	10	18	10

Table 2: active ω's through training

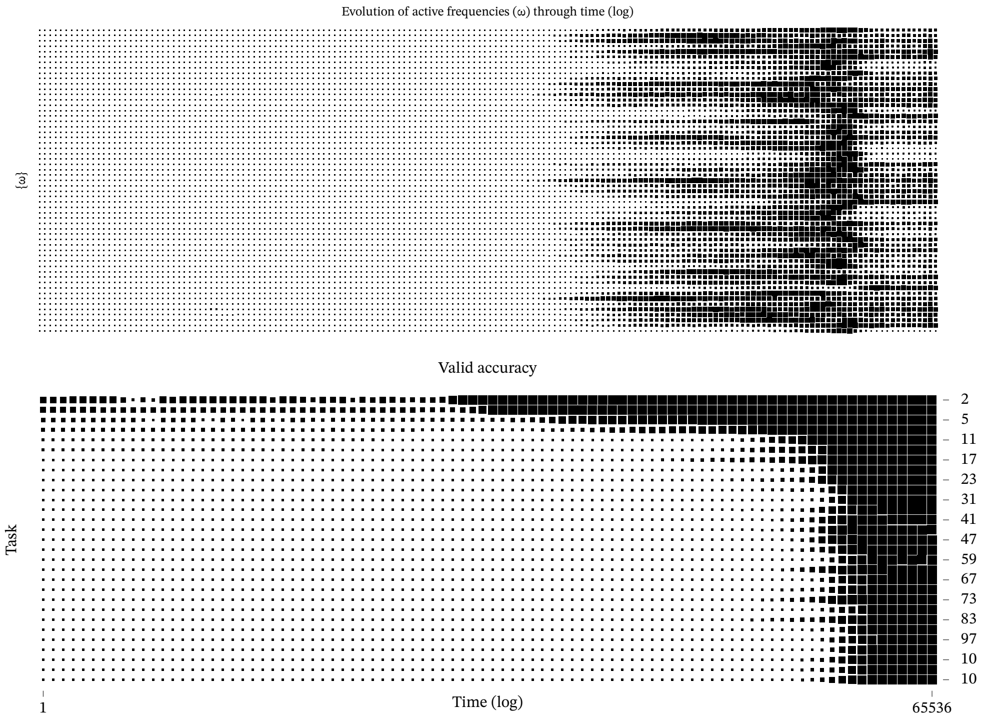

Figure 13: Figure 12 (top) and validation accuracy from Figure 3 (bottom)

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] A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra, “Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets,” no. arXiv:2201.02177. arXiv, Jan. 2022. doi: 10.48550/arXiv.2201.02177.

[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.

[3] 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.

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 (Appendix B). 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.