MIIII: MECHANISTIC INTERPRETABILITY ON MULTITASK IRREDUCIBLE INTEGER IDENTIFIERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Understanding how neural networks internalize algorithms remains a central challenge in deep learning. We investigate mechanistic interpretability in a multitask setting by training transformer models on 29 parallel modular arithmetic tasks, predicting the remainder of two-digit base-113 numbers modulo distinct primes less than 113 (of which there are 29). In addition to replicating prior findings on the dynamics of generalization ("grokking") and overfitting, we discover a third, previously unreported class of gradient updates that is neither overfitting nor part of a final generalized solution. Through comprehensive embedding, singular value decomposition, and Fourier analysis, we observe that these gradients updates are associated with general but temporary structures. Our results reveal how multitask learning shapes the development and interplay of internal mechanisms in deep models, and suggest future directions for speeding up generalization through more targeted gradient filtering in multitask environments. Code: `https://anonymous.4open.science/r/miiii-3B9E/`

1 INTRODUCTION

Recent years have seen deep learning (DL) models achieve remarkable proficiency in complex computational tasks, including protein structure prediction (Jumper et al., 2021), strategic reasoning (Dinan et al., 2022), and natural language generation (Radford and Narasimhan, 2018)—areas previously thought to be the exclusive domain of human intelligence. DL models are, however, subsymbolic, meaning that the models' atomic constituents do not map directly to mathematical notation. Recent works still attempts to define what interpretability even means in the DL context (Lipton, 2018).

Mathematically, DL refers to a set of methods that combine linear maps (matrix multiplications) with non-linearities (activation functions). Formally, all the potential numerical values of a given model's weights W can be thought of as a hypothesis space $\mathcal{H}$. Often, $\mathcal{H}$ is determined by human decisions (number of layers, kinds of layers, sizes of layers, etc.). $\mathcal{H}$ is then navigated using some optimization heuristic (such as gradient descent), in the hope of finding a $W \in \mathcal{H}$ that "performs well" (i.e., successfully minimizes some loss $\mathcal{L}$ often computed by a differentiable function with respect to W) on whatever training data is present. This vast, sub-symbolic hypothesis space, while frequently enabling the discovery of models that solve highly abstract tasks, makes it challenging to understand how any one particular solution actually works (i.e., a black box algorithm). Attempts at making the hypothesis space inherently interpretable are ongoing (Gavranović et al., 2024; Bronstein et al., 2021) though no particular approach has been widely adopted.

Intuitively, the ways in which a given W can achieve a low $\mathcal{L}$ can be placed on a continuum: on one side, we have overfitting, remembering the training data, (i.e. functioning as an archive akin to traditional lossy or even lossless compression; Cover and Thomas (2006)); and on the other, we have generalization, learning the rules that govern the relationship between input and output (i.e. functioning as an algorithm; Bishop (2006)).

When attempting to give a *mechanistic* explanation of a given DL model's behavior, it necessarily entails the *existence* of a mechanism. Mechanistic interpretability (MI) frequently assumes this mechanism to be general, thus making generalization a necessary (though insufficient) condition. Generalization ensures that there *is* a mechanism/algorithm present to be uncovered (necessity);

however, it is still possible for the given algorithm to be so obscurely implemented that reverse engineering, for all intents and purposes, is impossible (insufficiency). Various forms of regularization are used to incentivize the emergence of algorithmic (generalized) and interpretable, rather than archiving (over-fitted) behavior (Ba et al., 2016; Krizhevsky et al., 2017; Krogh and Hertz, 1991). As of writing, no MI work has explored the effect of multi-task learning in contrast to single task learning on algorithmically generated data—the focus of this paper. Multitask learning has a regularizing effect (Baxter, 2011). Formally, the set of hypotheses spaces for each task of a set of tasks (often called environment) is denoted by $\mathcal{H} \in \mathbb{H}$. When minimizing the losses across all tasks in parallel, generalizing W's are thus incentivized, as these help lower loss *across* tasks (in contrast to a memorizing W's that lower loss for one task). A W derived from a multi-task training process can thus be thought of as the intersection of the high-performing areas of all $\mathcal{H} \in \mathbb{H}$.

In this spirit, the present paper builds on the work of Nanda et al. (2023) which trains a transformer (Vaswani et al., 2017) model to perform modular addition, as seen in Eq. 1. The task is denoted as $\mathcal{T}_{\text{nanda}}$ throughout the paper.

$$(x_0 + x_1) \bmod p, \quad \forall\, x_0, x_1 < p, \quad p = 113 \tag{1}$$

The task in this paper focuses on predicting remainders modulo all primes q less than p, where x is interpreted as $x_0 p^0 + x_1 p^1$, formally shown in Eq. 2, and is referred to as $\mathcal{T}_{\text{miiii}}$:

$$\left(x_0 p^0 + x_1 p^1\right) \bmod q, \quad \forall\, x_0, x_1 < p, \quad \forall\, q < p, \quad p = 113 \tag{2}$$

$\mathcal{T}_{\text{miiii}}$ differentiates itself from $\mathcal{T}_{\text{nanda}}$ in two significant ways: *1)* it is non-commutative, and *2)* it is, as mentioned, multi-task. These differences present unique challenges for mechanistic interpretation, as the model must learn to handle both the order-dependent nature of the inputs and develop shared representations across multiple modular arithmetic tasks[1]. Further, as $\mathcal{T}_{\text{miiii}}$ is harder than $\mathcal{T}_{\text{nanda}}$ the model can be expected to generalize slower when trained on the former. Therefore, Lee et al. (2024)'s recent work on speeding up generalization by positing the model parameters gradients through time can be viewed as a sum of *1)* a slow varying generalizing component (which is boosted), and *2)*, a quick varying overfitting component (which is suppressed), is (successfully) replicated to make training tractable given our available computational resources.

More generally, modular arithmetic on primes is a particularly useful task for MI as it ensures uniformity among the output classes, allows for comparison with other MI work (Nanda et al., 2023). Further, from a number-theoretic point of view, primes contain mysteries ranging from the trivially solved—are there an infinite number of primes?—to the deceptively difficult—can all even numbers larger than 4 be described as the sum of two primes? The latter, known as Goldbach's Conjecture, remains unsolved after centuries. The choice of using every prime less than the square root of the largest number of the dataset also serves the following purpose: to test if a given natural number is prime, it suffices to test that it is not a multiple of any prime less than its square root—the set of tasks trained for here, can thus be viewed in conjunction as a single prime detection task (primes are the only samples whose target vector contains no zeros, since it is not a multiple of any of the factors q). There are about $n/\ln(n)$ primes less than n.

2 Related Work

Multiple papers describe the use of deep learning to detect prime numbers (Egri and Shultz, 2006; Lee and Kim, 2024; Wu et al., 2023). None are particularly promising as prime detection algorithms, as they do not provide speedups, use more memory, and are less accurate than traditional methods. However, in exploring the foundations of deep learning, the task of prime detection is interesting, as it is a simple task that is difficult to learn, and is synthetic, meaning that the arbitrary amounts of data are generated by a simple algorithm.

[1]Theoretically the tasks could all be solved using non-overlapping circuitry, though as the tasks are similar and the model is heavily l2 regularized during training this is disincentivesd from happening.

2.1 MECHANISTIC INTERPRETABILITY

Seminal works in MI include Nanda et al. (2023)'s focus on reverse engineering a model trained on Eq. 1. This work should be viewed in the context of Olah et al. (2018; 2017)'s work on interpretability through extensive visualization of model activation and weights.

Methods and tools used so far in MI include: Activation visualization across ordered samples; Singular value decomposition of weight matrices; Ablation studies to identify critical circuits. Conmy et al. (2023) even successfully automate circuit discovery; in the context of MI, "circuit" refers to a subgraph of a neural network that performs a particular function. Many reverse engineering methods from other fields, such as computational neuroscience or signal processing, almost certainly have their uses here as well.

In spite of deep learning's practical successes, uncertainty remains about its theoretical underpinnings, echoing the interpretability debate. Recent work attempts to place different DL architectures and concepts in a either geometric (Bronstein et al., 2021), information theoretic (Yu et al., 2021), or even category theoretic (Gavranović et al., 2024) context. However, no unified theory has emerged. Much interesting deep learning research thus focuses on practical, simple, or algorithmic tasks with known solutions and architectures. For example, grokking (Power et al., 2022), the (relatively) sudden generalization after overfitting, as elaborated later, is a recent and practical discovery.

Recent work by Bricken et al. (2023); Templeton et al. (2024) has focused on the use of sparse auto encoders to understand single layers of large language models.

2.1.1 MODULAR ADDITION

One such practical discovery is made by Nanda et al. (2023). A single layer transformer model with ReLU activation function was trained to perform modular addition ($\mathcal{T}_{\text{nanda}}$). Nanda et al. (2023)'s analysis of their trained model exemplifies MI methodology. They discovered that: *1)* The embedding layer learns trigonometric lookup tables of sine and cosine values as per Eq. 3; *2)* The feed-forward network combines these through multiplication and trigonometric identities (4), and *3)* The final layer performs the equivalent of argmax (Eq. 5).

$$\begin{aligned} x_0 &\to \sin(wx_0),\ \cos(wx_0) \\ x_1 &\to \sin(wx_1),\ \cos(wx_1) \end{aligned} \tag{3}$$

$$\begin{aligned} \sin(w(x_0+x_1)) &= \sin(wx_0)\cos(wx_1) + \cos(wx_0)\sin(wx_1) \\ \cos(w(x_0+x_1)) &= \cos(wx_0)\cos(wx_1) - \sin(wx_0)\sin(wx_1) \end{aligned} \tag{4}$$

$$\begin{aligned} \text{Logit}(c) &\propto \cos(w(x_0+x_1-c)) \\ &= \cos(w(x_0+x_1))\cos(wc) + \sin(w(x_0+x_1))\sin(wc) \end{aligned} \tag{5}$$

2.2 GENERALIZATION AND GROKKING

Power et al. (2022) shows generalization can happen "[...] well past the point of overfitting", dubbing the phenomenon "grokking". The phenomenon is now well established (Nanda et al., 2023; Humayun et al., 2024; Wang et al., 2024). Nanda et al. (2023) shows that a generalized circuit "arises from the gradual amplification of structured mechanisms encoded in the weights," rather than being a relatively sudden and stochastic encounter of an appropriate region of $\mathcal{H}$. The important word of the quote is thus "gradual".

By regarding the series of gradients in time as a stochastic signal, Lee et al. (2024) proposes decomposing the signal. Conceptually, Lee et al. (2024) argues that in the case of gradient descent, the ordered sequence of gradient updates can be viewed as consisting of two components: *1)* a fast varying overfitting component, and *2)* a slow varying generalizing components. The general algorithm explaining the relationship between input and output is the same for all samples, whereas the weights that allow a given model to function are unique for all samples. Though not proven, this intuition bears out in that generalization is sped up fifty-fold in some cases.

This echoes the idea that generalized circuits go through *gradual* amplification (Nanda et al., 2023). To the extent that this phenomenon is widespread, it bodes well for generalizable DL in that the generalizing signal that one would want to amplify might exist long before the model is fully trained and could potentially be boosted in a targeted way by the method described by Lee et al. (2024).

2.3 Multi-task learning in deep learning

As stated, multi-task learning has been shown to have a regularizing effect (Baxter, 2011; Maurer) as the hypothesis W that performs well across all of the hypothesis spaces $\mathcal{H} \in \mathbb{H}$ is more likely to be general. Viewed information theoretically, this concept is reminiscent of Shannon (2001)'s asymptotic equipartition property (Cover and Thomas, 2006), or even more generally, the law of large numbers, which states that the more samples we have of a distribution, the closer our *estimates* are to its underlying properties will align with the *true* underlying properties.

In the context of $\mathcal{T}_{\text{miiii}}$, multi-task learning is done by having the last layer output predictions for all tasks in parallel. Thus, whereas $\mathcal{T}_{\text{nanda}}$ outputs a single one-hot 1×113 vector for each of the potential remainders, $\mathcal{T}_{\text{miiii}}$, as we shall see, outputs a $1 \times q$ vector for each prime $q < p$ (i.e., 29 output-task vectors when $p = 113$). The embeddings layer and the transformer block are thus shared for all tasks, meaning that representations that perform well across tasks are incentivized.

3 Methodology

How exactly a given model implements an algorithm is a non-trivial question—even modular addition is implemented in a relatively obscure way Nanda et al. (2023), as per Eqs. 3, 4, and 5.

This investigation probes the fundamental algorithmic structures internalized by a transformer model trained on a set of basic prime number-related modular arithmetic tasks with slight variations in complexity. This approach provides insights into how and why specific algorithmic patterns emerge from seemingly straightforward learning processes.

As stated, the setup here differentiates itself from $\mathcal{T}_{\text{nanda}}$ in two crucial ways: *1)* It is non-commutative; and *2)* it is multitask.

3.1 Tasks

Stated plainly: the task $\mathcal{T}_{\text{miiii}}$ predicts the remainder when dividing a two-digit base-p number by each prime factor q less than p. The set of prime factors we construct tasks for is thus $q = q \in \mathbb{P} : q < p$. For $p = 113$, this yields 29 parallel tasks, one for each prime less than p. Each task predicts a remainder in the range $[0, q-1]$. This means smaller primes like 2 and 3 require binary and ternary classification, respectively, while the largest prime less than p, 109, requires predictions across 109 classes. The tasks thus naturally vary in difficulty: predicting $\mod 2$ requires distinguishing odd from even numbers (which in binary amounts to looking at the last bit) while predicting $\mod 109$ involves making a selection between many relatively similar classes. The expected cross entropy for an n-class problem is $\ln(n)$, which has implications for the construction of the loss function, further discussed in Section 3.2. Additionally, a baseline task $\mathcal{T}_{\text{baseline}}$ was constructed by shuffling the y-labels of $\mathcal{T}_{\text{miiii}}$, and a task ablation test $\mathcal{T}_{\text{masked}}$ was constructed by masking away the four simplest tasks $q \in 2, 3, 5, 7$.

3.1.1 Data

Input Space (X): Each input $x \in X$ represents a number in base p using two digits, (x_0, x_1), where the represented number is $x_0 p^0 + x_1 p^1$. For example, with $p = 11$, the input space consists of all pairs (x_0, x_1) where $x_0, x_1 < 11$, representing numbers up to $11^2 - 1 = 120$. This yields a dataset of 121 samples. Figure 18 visualizes this input space, with each cell representing the value $x_0 p^0 + x_1 p^1$.

Output Space (Y): For each input x, a vector $yinY$ contains the remainder when dividing by each prime less than p. For $p = 11$, this means predicting the remainder when dividing by 2, 3, 5, and 7. Each element y_i ranges from 0 to $q_i - 1$ where q_i is the i-th prime. Figure 16 visualizes these remainders, with each subplot showing the remainder pattern for a specific prime divisor. For

comparison, the rightmost plot shows the output space of Nanda et al. (2023)'s modular addition task.

3.1.2 MODEL

The model follows the original transformer architecture (Vaswani et al., 2017) with several key design choices aligned with recent work on mechanistic interpretability (Nanda et al., 2023; Lee et al., 2024): biases are disabled, and layer normalization is not used. The model consists of three main components: an embedding layer, transformer blocks, and an output layer. All weights are initialized following He et al. (2015).

3.2 TRAINING

Hyper parameter optimization was conducted using Optuna (Akiba et al., 2019), searching over the parameters shown in Table 1.

dropout	λ	wd	d	lr	heads
$\{0,\ \frac{1}{2},\ \frac{1}{5},\ \frac{1}{10}\}$	$\{0,\ \frac{1}{2},\ 2\}$	$\{0,\ \frac{1}{10},\ \frac{1}{2},\ 1\}$	$\{128,\ 256\}$	$\{3\text{e}{-4},\ 1\text{e}{-4}\}$	$\{4,\ 8\}$

Table 1: Hyperparameter search space for training.

The model is trained using AdamW (Loshchilov and Hutter, 2019) with $beta_1 = 0.9$, $beta_2 = 0.98$ following (Nanda et al., 2023). To handle the varying number of classes across tasks (from 2 classes for mod 2 to 109 classes for mod 109), a modified (weighted) mean cross-entropy loss is created, correcting for the difference in the expected loss within each task. Note that $\mathbb{E}[\mathcal{L}_{\text{MCE}}] = -\ln \text{q}$, where q is the number of classes within the task in question. Correcting for this, the loss function becomes as shown in Eq. 6.

$$\begin{aligned} L_{\mathcal{T}_{\text{miiii}}} &= \sum_{q \in \mathcal{Q}} \frac{L_{\text{MCE},q}}{\ln(q)} \\ &= \sum_{q \in \mathcal{Q}} \frac{\sum_{i=1}^{n} \sum_{j=0}^{q-1} y_{qij} \ln\left(\hat{y}_{qij}\right)}{n \ln(q)} \\ &= \sum_{q \in \mathcal{Q}} \sum_{i=1}^{n} \sum_{j=0}^{q-1} \frac{y_{qij} \ln\left(\hat{y}_{qij}\right)}{n \ln(q)} \end{aligned} \tag{6}$$

To accelerate generalization, gradient filtering as per Lee et al. (2024) is used:

$$g_t = \nabla_\theta L + \lambda \left(\alpha\, e_{t-1} + (1-\alpha)\, g_{t-1}\right) \tag{7}$$

where e_t is the exponential moving average of gradients with decay rate $\alpha = 0.98$, and λ controls the influence of the slow-varying component.

The training uses full batch gradient descent with the entire dataset of p^2 samples (12769 when $p = 113$). The model is evaluated on a held-out validation set after each epoch, tracking per-task accuracy and loss. As the setup used in $\mathcal{T}_{\text{nanda}}$, training was done on thirty percent of the total dataset, with the remaining used for validation (1000 samples) and testing (remaining). Further, as $\mathcal{T}_{\text{miiii}}$ involves the learning of 29 (when $p = 113$) tasks rather than 1, and due to each task's non-commutativity, a larger hidden dimension of 256 was added to the hyper parameter search space, as well as the potential for 8 heads ($\mathcal{T}_{\text{nanda}}$ was solved with a hidden dimension of 128, and 4 heads). The number of transformer blocks was kept at 1, as this ensures consistency with $\mathcal{T}_{\text{nanda}}$ (and as full generalization was possible; see Section 4).

Training was done on a NVIDIA V100 GPU, with Python3.11 and extensive use of "JAX 0.6" and its associated ecosystem. Neuron activations were calculated at every training step and logged for later analysis.

4 RESULTS

The best-performing hyper-parameters for training the model on $\mathcal{T}_{\text{miiii}}$ are listed in Table 2. Notably, the model did converge slower with $\lambda = 0$, confirming the utility of the gradient amplification method proposed by Lee et al. (2024) in the context of $\mathcal{T}_{\text{miiii}}$.

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

Table 2: Result of hyper-parameter search over $\mathcal{T}_{\text{miiii}}$.

4.1 MODEL PERFORMANCE

Figure 1 show the training and validation accuracy on $\mathcal{T}_{\text{miiii}}$ over time. The model achieves a perfect accuracy of 1 on the validation set across all 29 tasks. The cross-entropy loss in 13 echoes this. In short—and to use the terminology of Power et al. (2022)—the model "grokked" on all tasks. Interestingly, tasks corresponding to modulo 2, 3, 5, and 7 generalized in succession, while the remaining 25 tasks generalized around epoch 40k in no particular order. This might suggest that the model initially learned solutions for the simpler tasks and later developed a more general computational strategy that allowed it to generalize across the remaining, more complex tasks.

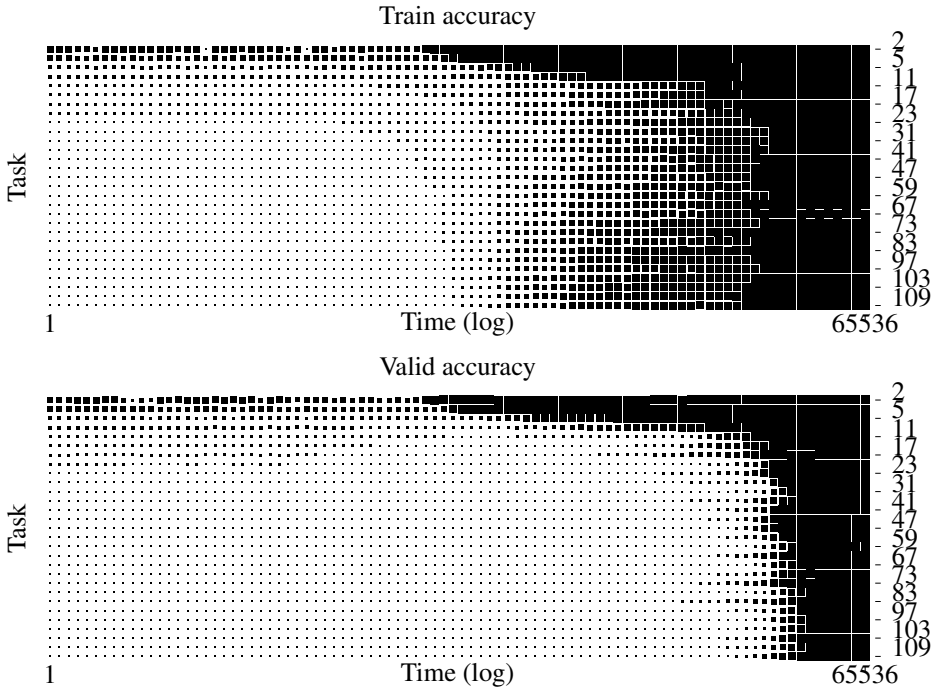

Figure 1: Accuracy training "curves": Training (top) and validation (bottom) accuracy over time (the x-axis in log-scale). We see grokking occur on all tasks, first for $q \in \{2, 3, 5, 7\}$ in that order, and then the remaining 25 in no particular order.

4.2 EMBEDDINGS

Positional embeddings play a crucial role in transformers by encoding the position of tokens in a sequence. Figure 10 compares the positional embeddings of models trained on $\mathcal{T}_{\text{nanda}}$ and $\mathcal{T}_{\text{miiii}}$.

For $\mathcal{T}_{\text{nanda}}$, which involves a commutative task, the positional embeddings are virtually identical, with a Pearson correlation of 0.95, reflecting that the position of input tokens does not significantly alter their contribution to the task. In contrast, for $\mathcal{T}_{\text{miiii}}$, the positional embeddings have a Pearson

correlation of -0.64, indicating that the embeddings for the two positions are different. This difference is expected due to the non-commutative nature of the task, where the order of x_0 and x_1 matters ($x_0\dot{p}^0! = x_0\dot{p}^1$). This confirms that the model appropriately encodes position information for solving the tasks.

Recall that a matrix $\mathbf{M}$ of size $m \times n$ can be decomposed into its singular values as $\mathbf{M} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^T$ (where the transpose is the complex conjugate if $\mathbf{M}$ is complex), where $\mathbf{U}$ is $m \times m$, $\mathbf{\Sigma}$ is an $m \times n$ rectangular diagonal matrix (whose diagonal is represented as a flat vector throughout this paper), and $\mathbf{V}^T$ is an $n \times n$ matrix. Intuitively, this can be thought of as rotating in the input space, then scaling, and then rotating in the output space.

Figure 2 displays the singular values of the token embeddings learned for $\mathcal{T}_{\text{nanda}}$ and $\mathcal{T}_{\text{miiii}}$. The singular values for $\mathcal{T}_{\text{miiii}}$ are more diffuse, indicating that a larger number of components are needed to capture the variance in the embeddings compared to $\mathcal{T}_{\text{nanda}}$. This suggests that the token embeddings for $\mathcal{T}_{\text{miiii}}$ encode more complex information, reflecting the increased complexity of the multi-task learning scenario.

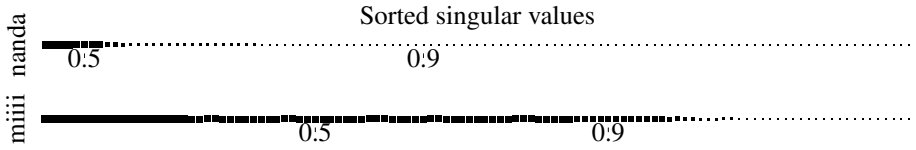

Figure 2: First 83 of 113 singular values (truncated for clarity) of U for $\mathcal{T}_{\text{nanda}}$ (top) and $\mathcal{T}_{\text{miiii}}$ (bottom). The ticks indicate the points where 50% and 90% of the variance is accounted for. We thus see that for $\mathcal{T}_{\text{miiii}}$, the embedding space is much more crammed.

Figures 15 and 11 present the most significant singular vectors of U for $\mathcal{T}_{\text{nanda}}$ and $\mathcal{T}_{\text{miiii}}$, respectively. Visual inspection shows periodicity in the top vectors for both models, but the $\mathcal{T}_{\text{miiii}}$ model requires more vectors to capture the same amount of variance, consistent with the diffuse singular values observed in Figure 2.

To further understand the structure of the token embeddings, we applied the Fast Fourier Transform (FFT). Only a few frequencies are active for $\mathcal{T}_{\text{nanda}}$ as seen in Figure 3, consistent with the model implementing a cosine-sine lookup table as described in Nanda et al. (2023).

For the $\mathcal{T}_{\text{miiii}}$ model, we observe a broader spectrum of active frequencies (4). This is expected due to the model having to represent periodicity corresponding to 29 primes.

Comparing with $\mathcal{T}_{\text{baseline}}$ in Figure 12, the periodicity is understood to be a structure inherent to the data picked up by the model.

4.3 Analysis of Neuron Activations and Frequencies

To understand the internal mechanisms developed by the model, we analyzed the neuron activations after the output weight matrix W_{out} for the model trained on $\mathcal{T}_{\text{miiii}}$. Figure 7 shows that these activations exhibit periodic patterns with respect to (x_0, x_1). This periodicity aligns with the modular arithmetic nature of the tasks, mirroring Nanda et al. (2023) ($\mathcal{T}_{\text{nanda}}$).

For comparison, Figure 9 shows the neuron activations for a model trained on $\mathcal{T}_{\text{baseline}}$. These activations do *not* exhibit periodicity, confirming that the observed periodic patterns in the models trained for $\mathcal{T}_{\text{miiii}}$ and $\mathcal{T}_{\text{nanda}}$ are indeed a result of the modulo operations inherent in the tasks.

The analysis of active frequencies *through training* using the Fast Fourier Transform (FFT) is illustrated in Figure 5, with the core findings showing a spike in frequency activation around epoch 16,384 (see Table 3). The top plot shows the different frequencies of the transformer block's feed-forward neurons evolving as the model learns. The bottom plot displays the variance of frequency activations and the number of frequencies exceeding a significance threshold $\omega > \mu + 2\sigma$ (i.e., where spots like the ones of the bottom row of Figure 7 are active). Initially, a handful of frequencies become dominant as the model generalizes on the first four tasks. As training progresses and the model begins to generalize on the remaining tasks, more frequencies become significant, suggesting that the model is developing more complex internal representations to handle the additional tasks.

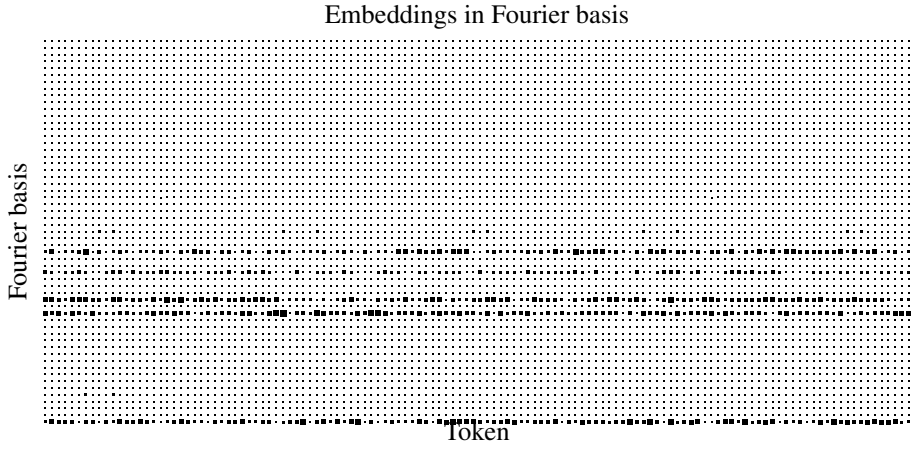

Figure 3: $\mathcal{T}_{\text{nanda}}$ tokens in Fourier basis: Note how all tokens are essentially linear combinations of the five most dominant Fourier basis vectors. The sparsity echoes the findings in 2 that very few directions in the embedding space are used.

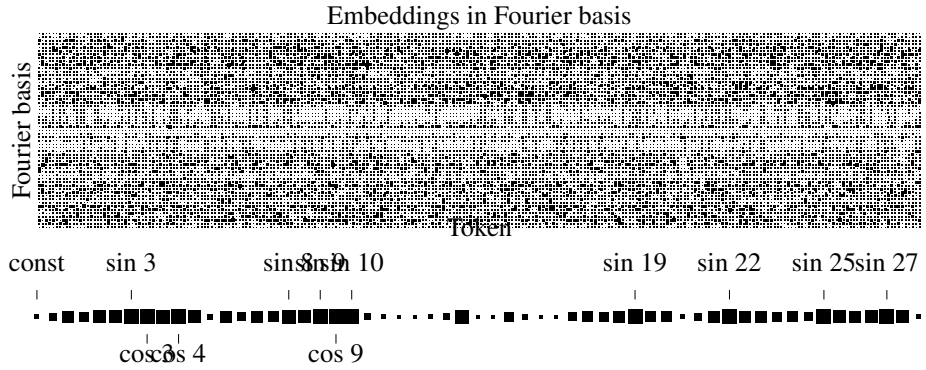

Figure 4: The periodicity in the $\mathcal{T}_{\text{miiii}}$ embeddings involves a much larger fraction of the Fourier basis, echoing the multiple tasks and their innate difference in frequency (recall that all tasks are performed on unique primes q).

Figure 14 shows the L2 norms of gradients over time for the different weight matrices of the model trained on $\mathcal{T}_{\text{miiii}}$. The gradient norms provide insights into how different parts of the model are being updated during training. Like with Nanda et al. (2023), the attention layer converges quickly, echoing their finding that it does not contribute much to solving the modular arithmetic task.

These results demonstrate that more frequencies are involved when training on $\mathcal{T}_{\text{miiii}}$ compared to $\mathcal{T}_{\text{nanda}}$. The increased frequency components reflect the need for the model to encode multiple periodic patterns corresponding to the various modular arithmetic tasks.

Combining the analysis of embeddings and the transformer block neurons, we see that: 1. A lot more frequencies are in play for $\mathcal{T}_{\text{miiii}}$ than in $\mathcal{T}_{\text{nanda}}$. 2. Neurons remain highly reactive to a very small set of frequencies. 3. The periodicity is an artifact of the modulo group by analysis of $\mathcal{T}_{\text{baseline}}$

5 Discussion

Recall that, when viewed individually, the sub-tasks of $\mathcal{T}_{\text{miiii}}$ differ from $\mathcal{T}_{\text{nanda}}$ only in commutativity (making the task harder) and in prime since $q < p$ (making the task easier for smaller q's like $2, 3, 5, 7$—though less so as q approaches p). Figure 10 indicates that the model learns to account for the commutativity by using positional embeddings.

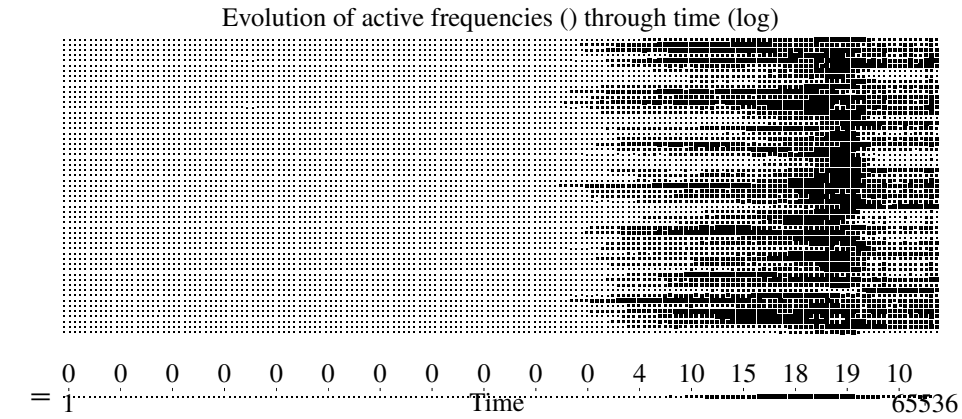

Figure 5: Top: Each row is a time series of activations of a particular frequency. Frequency dominance is averaged over all neurons through training (average activation of frequencies). We see four distinct phases: *1)* no significant frequencies, *2)* frequencies gradually emerging, *3)* a wall of frequencies, and *4)* frequency count similar to phase 2. Bottom: total number of active frequencies at the corresponding time step. A frequency ω is active when it is more than two standard deviations above the mean activation ($\omega > (\mu + 2\sigma)$).

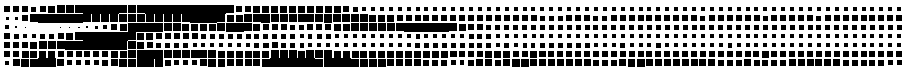

Figure 6: Each row is the sum of all rows for the training run of a specific task subset (linear time). Note how the number of active frequencies consistently reaches a maximum early during training, independently of the task mask.

During training, the changes in the number of active neuron frequencies ω in the feed-forward layer (see 5) echo the loss and accuracy progression seen in Figures 1 and 13. Further, as the model groks on the primes $q \in 2, 3, 5, 7$, a handful of frequencies become dominant, similar to the original model trained on $\mathcal{T}_{\text{nanda}}$.

Thus we can conclude that *1)* the model learns to correct for commutativity, and *2)* the model is marred by periodicity as per both the token embedding (4.2) and feed-forward layer analysis (4.3). Combining these facts, we can assume the learned mechanism to be extremely similar (and perhaps identical when ignoring commutativity) to the one outlined by Nanda et al. (2023) in Eqs. 3, 4, and 5.

As the remaining 25 tasks are learned, we see a temporary spike in the number of active frequencies, disappearing again as the model generalizes fully (reaches perfect accuracy). The observation that the number of active frequencies before and after the remaining 25 tasks are learned are the same indicates a reuse of circuitry. However, the fact of the spike suggests that additional circuits form during the generalization process. Viewed in the context of Lee et al. (2024)'s method for boosting slow-varying generalizing components a question emerges: are there circuits that only facilitate the development of generalization, but are not present in the generalized mechanisms? Real life gives plenty of examples of phenomena that make itself obsolete (e.g., a medicine that fully eradicates an illness and is thus no longer needed). Viewed this way, the spike suggests we might divide the set of circuits in two: *1)*; those useful for the mechanism, and *2)*, those useful for *learning* the mechanism.

6 Conclusion

This paper explored the impact of multi-task learning on mechanistic interpretability by training a transformer model on a non-commutative, multi-task modular arithmetic problem $\mathcal{T}_{\text{miiii}}$. The model successfully generalizes across all tasks, learning complex internal representations that capture the unique periodic nature of modular arithmetic across multiple primes. Analysis reveals that while the

epoch	256	1024	4096	16384	65536
active freqs.	0	0	10	18	10

Table 3: Number of active frequencies on $\mathcal{T}_{\text{miiii}}$ over epochs.

model reuses and integrates circuits for simpler tasks, additional circuits may form during training to facilitate generalization to more complex tasks.

These findings highlight that multi-task learning influences the emergence and complexity of internal mechanisms, posing challenges for mechanistic interpretability but also offering insights into how models internalize algorithms. Understanding these dynamics is important for advancing interpretability and reliability in deep learning systems. Future work includes exploring the distinction between circuits that aid in learning versus those that contribute to the final mechanism/solution and investigating how variations in task design impact the development of internal representations. Furthermore, the findings indicate that Lee and Kim (2024)a might be further modified to dynamically induce temporarily useful circuitry. Advancing the understanding of how deep learning models handle multiple tasks contributes to the larger goal of making these models more interpretable and reliable.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 2623–2631, New York, NY, USA, July 2019. Association for Computing Machinery. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330701.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016.

J. Baxter. A Model of Inductive Bias Learning, June 2011.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, 2006. ISBN 978-0-387-31073-2.

Trenton Bricken, Alex Templeton, Jonathan Uesato, Brando Miranda, Brian Chen, Adam Jermyn, Thomas Conerly, Nick Turner, Ethan Perez, Shan Carter, Ludwig Schubert, Chris Olah, and Amanda Askell. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits*, 2023. URL `https://transformer-circuits.pub/2023/monosemantic-features/`.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, May 2021.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards Automated Circuit Discovery for Mechanistic Interpretability, October 2023.

T. M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, Hoboken, N.J, 2nd ed edition, 2006. ISBN 978-0-471-24195-9.

Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra. Human-level play in the game of *Diplomacy* by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, December 2022. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.ade9097.

László Egri and Thomas R Shultz. A Compositional Neural-network Solution to Prime-number Testing. 2006.

Bruno Gavranović, Paul Lessard, Andrew Dudzik, Tamara von Glehn, João G. M. Araújo, and Petar Veličković. Categorical Deep Learning: An Algebraic Theory of Architectures, February 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, February 2015.

Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Deep Networks Always Grok and Here is Why, June 2024.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, August 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017. ISSN 0001-0782, 1557-7317. doi: 10.1145/3065386.

Anders Krogh and John Hertz. A Simple Weight Decay Can Improve Generalization. In *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991.

Jaerin Lee, Bong Gyun Kang, Kihoon Kim, and Kyoung Mu Lee. Grokfast: Accelerated Grokking by Amplifying Slow Gradients, June 2024.

Serin Lee and S. Kim. Exploring Prime Number Classification: Achieving High Recall Rate and Rapid Convergence with Sparse Encoding, February 2024.

Zachary C. Lipton. The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, June 2018. ISSN 1542-7730. doi: 10.1145/3236386.3241340.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019.

Andreas Maurer. Bounds for Linear Multi-Task Learning.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, October 2023.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature Visualization. *Distill*, 2(11): 10.23915/distill.00007, November 2017. ISSN 2476-0757. doi: 10.23915/distill.00007.

Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The Building Blocks of Interpretability. *Distill*, 3(3): 10.23915/distill.00010, March 2018. ISSN 2476-0757. doi: 10.23915/distill.00010.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets, January 2022.

Alec Radford and Karthik Narasimhan. Improving Language Understanding by Generative Pre-Training. 2018.

C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, January 2001. ISSN 1559-1662, 1931-1222. doi: 10.1145/584091.584093.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C Daniel Freeman, Theodore R Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits*, 2024. URL `https://transformer-circuits.pub/2024/scaling-monosemanticity/`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, December 2017.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokked Transformers are Implicit Reasoners: A Mechanistic Journey to the Edge of Generalization, May 2024.

Da Wu, Jingye Yang, Mian Umair Ahsan, and Kai Wang. Classification of integers based on residue classes via modern deep learning algorithms. https://arxiv.org/abs/2304.01333v3, April 2023.

Shujian Yu, Luis Sanchez Giraldo, and Jose Principe. Information-Theoretic Methods in Deep Neural Networks: Recent Advances and Emerging Opportunities. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4669–4678, Montreal, Canada, August 2021. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-9-6. doi: 10.24963/ijcai.2021/633.

A ACTIVATIONS

Since the entire dataset consists of only 12,769 samples, we can, for every neuron in the hidden layer, log its activation as we vary the first and second input.

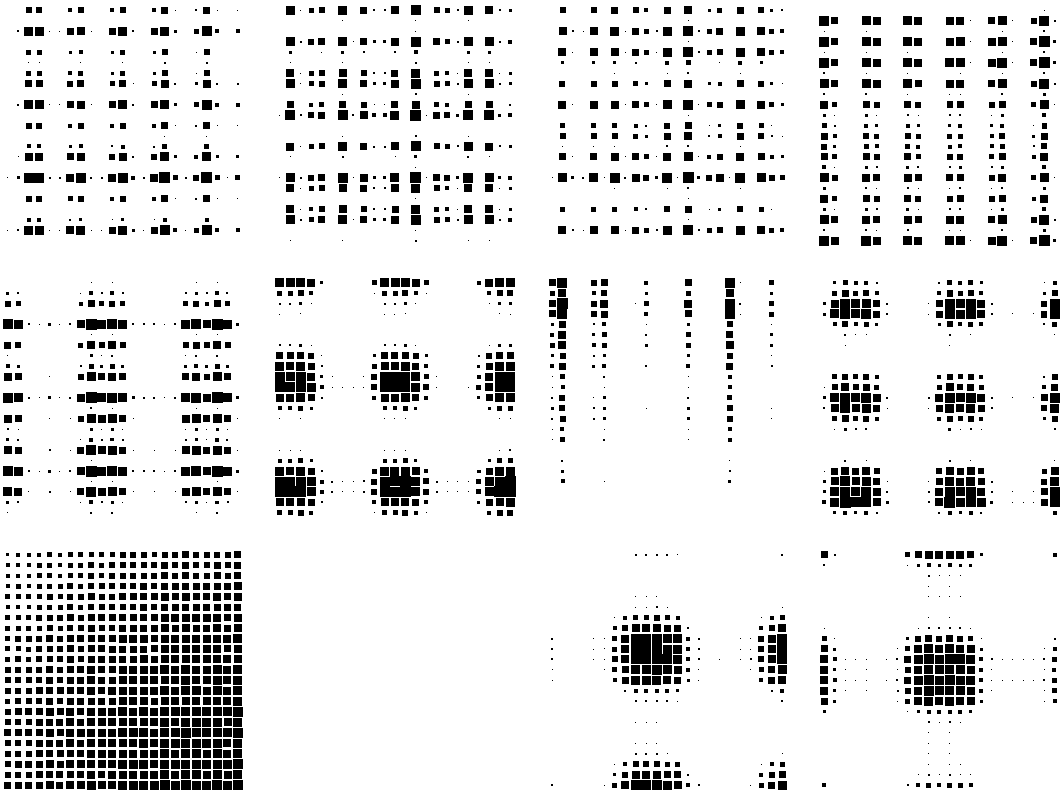

Figure 7: Neuron activations as x_0 and x_1 vary from 0 to 23 on the first and second axes respectively (corresponding to the top left corner of the full neuron) for a 256-d model trained on all 30 tasks

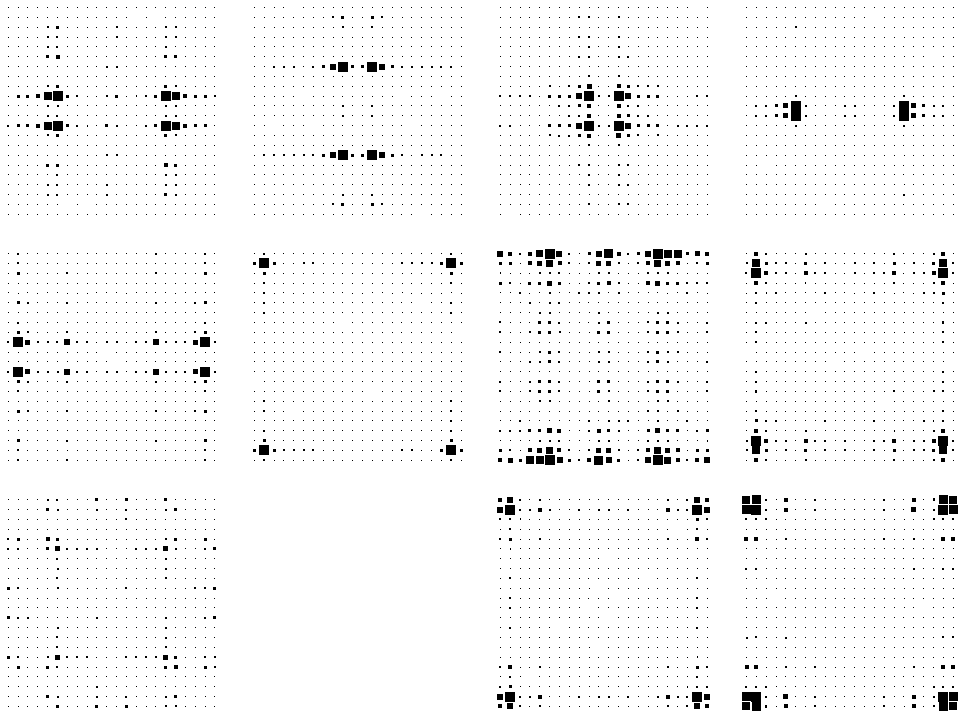

Figure 8: Fourier transform of neuron activations as x_0 and x_1 vary from 0 to 23 on the first and second axes respectively (corresponding to the top left corner of the full neuron) for a 256-d model trained on the tasks related to primes $\{103, 107, 109, 113\}$. Note neurons tend to be less active compared to Figure 7

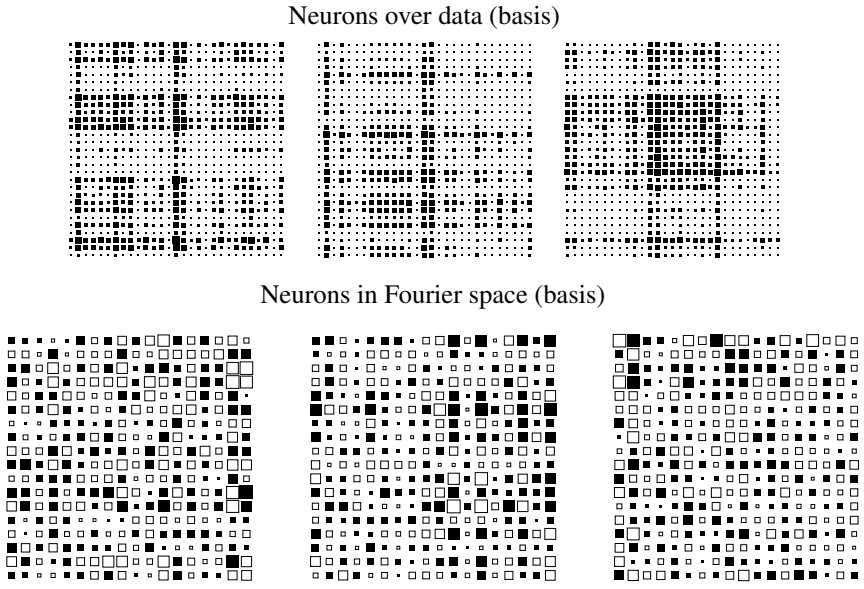

Figure 9: Neuron activations for model trained on $\mathcal{T}_{\text{baseline}}$. As in Figure 12, no periodicity is observed for the baseline.

B EMBEDDINGS

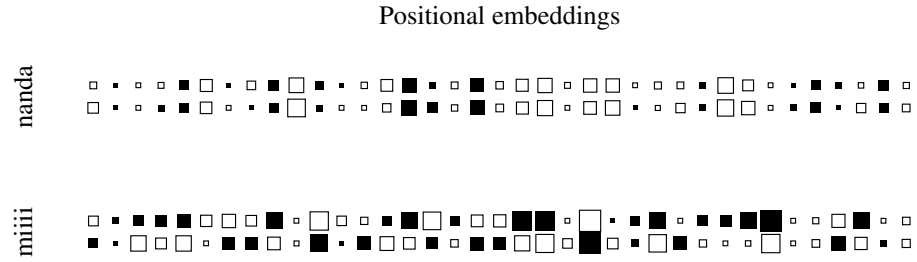

Figure 10: Positional embeddings for (x_0, x_1) for models trained on $\mathcal{T}_{\text{nanda}}$ (top) and $\mathcal{T}_{\text{miiii}}$ (bottom). Pearson's correlation is 0.95 and -0.64, respectively. This reflects the commutativity of $\mathcal{T}_{\text{nanda}}$ and the lack thereof for $\mathcal{T}_{\text{miiii}}$. Hollow cells indicate negative numbers.

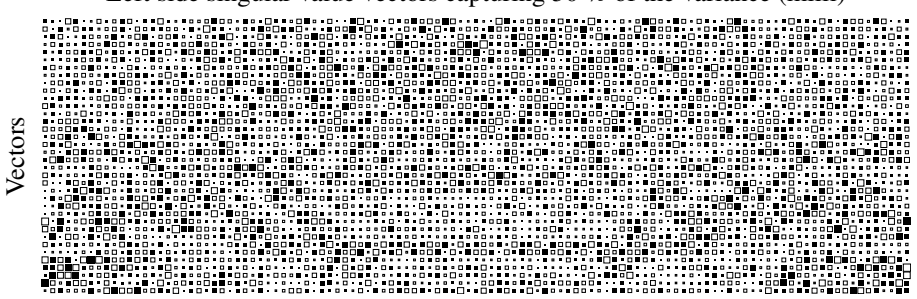

Figure 11: $\mathcal{T}_{\text{miiii}}$'s most significant vectors of U. Note that, like in Figure 15, we still observe periodicity, but there are more frequencies in play, as further explored in Figure 4.

C TRAINING LOSS

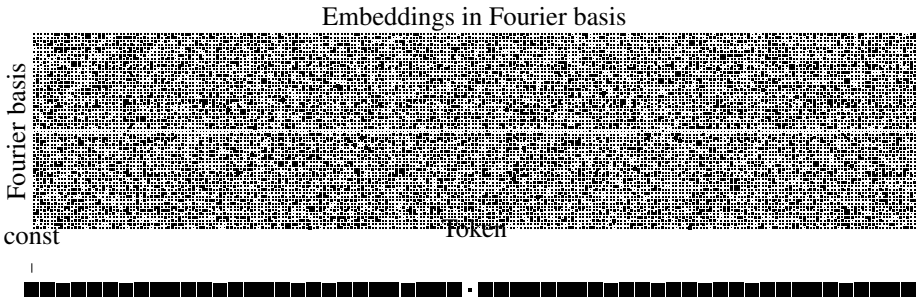

Figure 12: Sanity check: Embeddings for $\mathcal{T}_{\text{baseline}}$ in Fourier basis have no periodicity. The periodicity is indeed an artifact of the modulo operator.

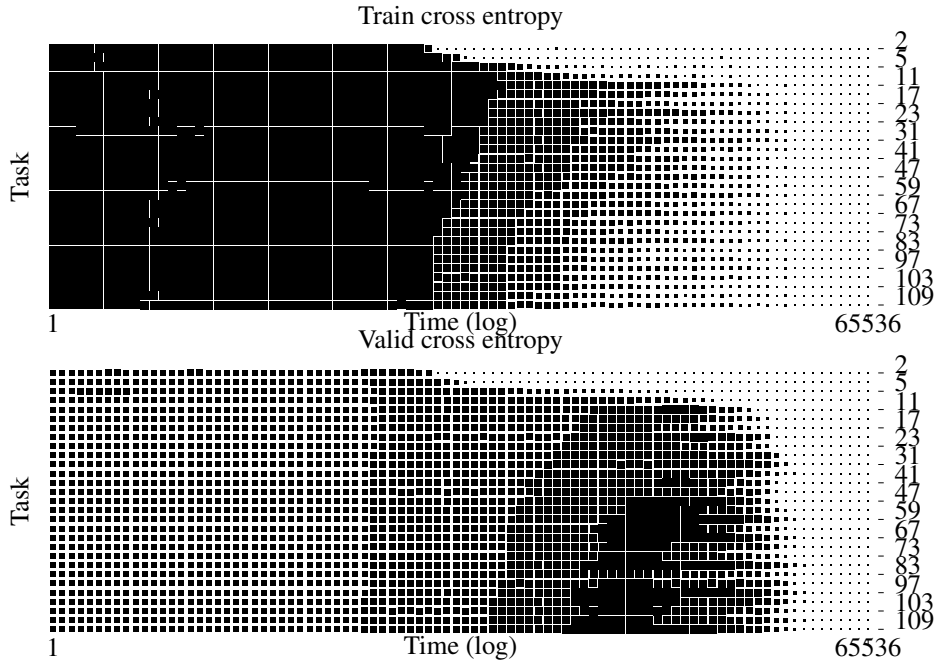

Figure 13: Cross-entropy loss on training (top) and validation (bottom) over time (note the log scale on the x-axis).

D VISUALIZING X AND Y

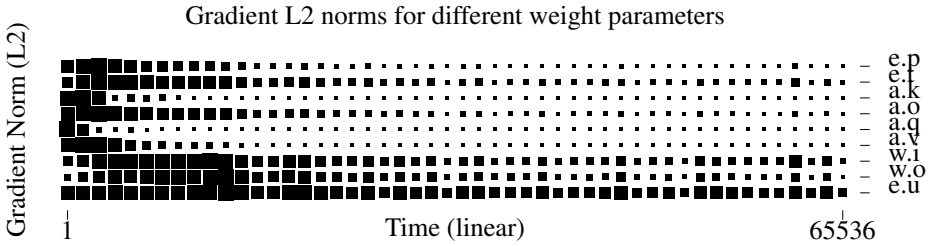

Figure 14: L2 norms of gradients over time for the different weight matrices of the model trained on $\mathcal{T}_{\text{miiii}}$. Row order corresponds to how deep in the model the weight is used. This shows when during training what parts of the model are updated. `e.*` are embeddings, `a.*` attention layer weights, and `w.*` weights of the feed-forward module; `e.u` is the final un-embedding layer.

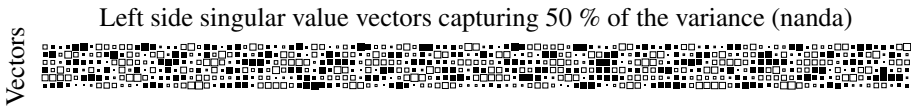

Figure 15: $\mathcal{T}_{\text{nanda}}$'s most significant (cutoff at 0.5 as per Figure 2) singular vectors of U from the singular value decomposition. Note this looks periodic!

E MULTI-TASK RUNS

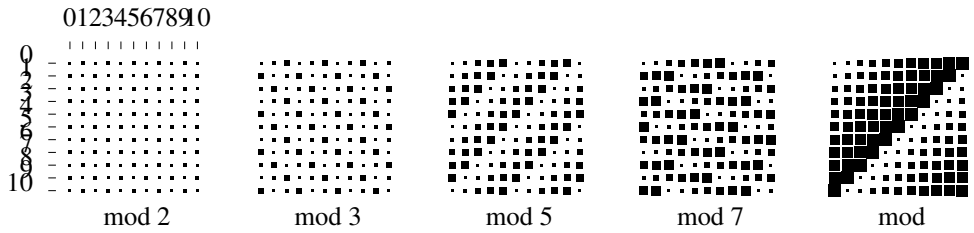

Figure 16: Visualizing tasks in Y (for $p = 11$). x_0 and x_1 vary on the two axis, with the remainder modulo $q \in 2, 3, 5, 7$ indicated by the square size. Note the innate periodicity of the modulo operator. Further note that $x \mod p$ is symmetrical across the matrix diagonal, which is an expression of that particular subtask's commutativity.

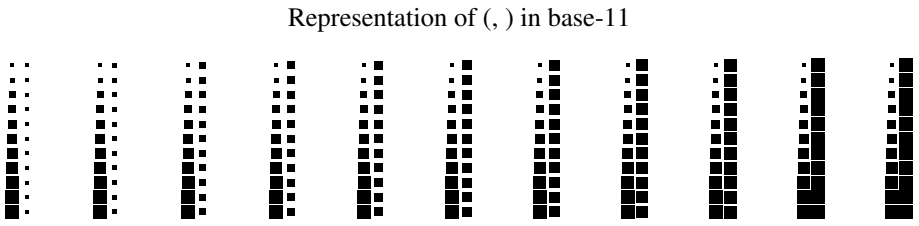

Figure 17: Visualizing X (for a small dataset where $p = 11$). Each cell represents the tuple (x_0, x_1). The top left shows 0 $(0, 0)$, and the bottom right shows 120 $(10, 10)$—both in base-11.

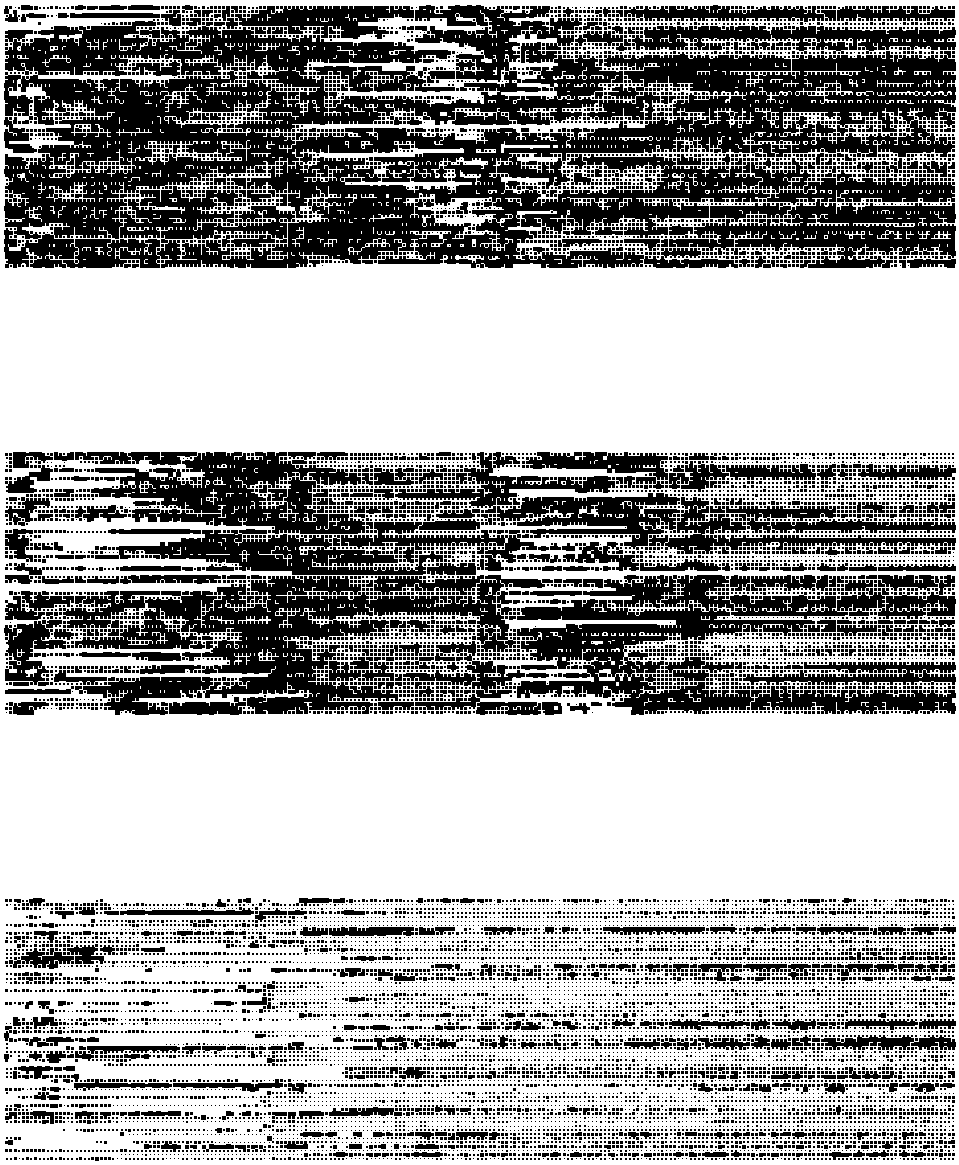

Figure 18: Frequency heatmaps for all tasks (top) top 10 largest primes (middle) and original Nanda et al. (2023) task (bottom). Rows are different frequencies ω and columns are log time steps. The size of the black square at a given location is proportional to the number of neurons in the MLP layers that has the given frequency ω significantly active at the given point in time.