Arithmetic in the Wild: Llama uses Base-10 Addition to Reason About Cyclic Concepts

Sheridan Feucht[⋆a,b] **Tal Haklay**[⋆a,c]
Usha Bhalla[a,d] **Daniel Wurgaft**[a,e] **Can Rager**[a]
Raphaël Sarfati[a] **Jack Merullo**[a] **Thomas McGrath**[a] **Owen Lewis**[a]
Ekdeep Singh Lubana[†a] **Thomas Fel**[†a] **Atticus Geiger**[†a]

[⋆]Equal contribution [†]Equal senior contribution

[a]Goodfire AI [b]Northeastern University [c]Technion IIT
[d]Harvard University [e]Stanford University

Abstract

Does structure in representations imply structure in computation? We study how Llama-3.1-8B reasons over cyclic concepts (e.g., *"what month is six months after August?"*). Even though Llama-3.1-8B's representations for these concepts are circularly structured, we find that instead of directly computing modular addition in the period of the cyclic concept (e.g., 12 for months), the model re-uses a generic addition mechanism across tasks that operates independently of concept-specific geometry. First, it computes the sum of its two inputs using base-10 addition (*six* + *August*=14). Then, it maps this sum back to cyclic concept space (14→February). We show that Llama-3.1-8B uses task-agnostic Fourier features to compute these sums—in fact, these features have periods that respect standard base-10 addition, e.g., 2, 5, and 10, rather than the cyclic concept period (e.g., 12 for months). Furthermore, we identify a sparse set of 28 MLP neurons re-used across all tasks (approximately 0.2% of the MLP at layer 18) that can be partitioned into disjoint clusters, each computing the sum for a Fourier feature with a different period. Our work highlights how an interplay between causal abstraction and feature geometry can deepen our mechanistic understanding of LMs.[1]

1 Introduction

What is the algorithmic role of representation geometry in language models? We investigate cyclic concepts such as weekdays and months as a case study, which LMs represent using circular geometry (Engels et al., 2025; Modell et al., 2025; Karkada et al., 2026; Prieto et al., 2026; Park et al., 2025a). Because small transformers trained on modular addition operate over a periodic basis (Nanda et al., 2023; Zhong et al., 2023; Furuta et al., 2024), which also fits a circular geometry (Morwani et al., 2024), a natural hypothesis is that LMs compute answers to questions like "`what month is six months after August?`" using a similar modular arithmetic algorithm. However, we find that this is not the case: Llama-3.1-8B (Llama-Team, 2024), in fact, uses a base-10 addition mechanism for cyclic tasks, converting the resulting sum back to circular representations in late layers.

Using causal analysis (Vig et al., 2020; Geiger et al., 2025a;b; Mueller et al., 2026), we isolate a base-10 addition mechanism that Llama-3.1-8B uses to solve cyclic tasks (Figure 1). This mechanism, which is re-used for months, weekdays, hours, *and* standard addition, computes a sum in numerical space (e.g., *six* + *August* = *6* + *8* = *14*) that is mapped to concept space in subsequent layers (14→February). Because this addition mechanism is shared across tasks, we can patch between standard addition prompts (*a*+*b*=) and cyclic prompts with predictable results.

[1]Code and data available at `https://github.com/goodfire-ai/arithmetic-wild`

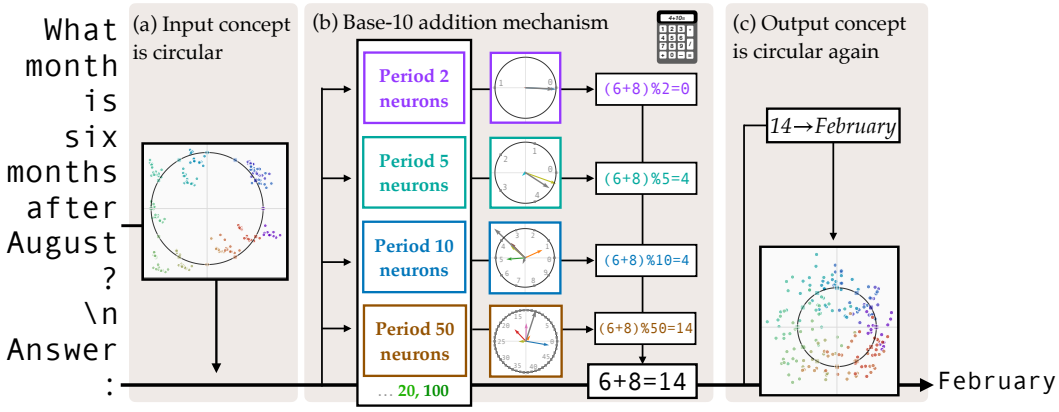


Figure 1: Llama-3.1-8B calculates *six months after August* with a standard addition mechanism that is used for numbers, months, weekdays, and hours. (a) Cyclic concepts are represented with circular geometry at the input token position (Engels et al., 2025). (b) The model computes addition in a base-10 Fourier number space (Kantamneni & Tegmark, 2025; Fu et al., 2026) using the same neurons for all tasks. We discover 28 MLP neurons forming distinct clusters, where each cluster computes the sum for a specific periodicity (circle radii scaled for legibility). (c) This sum is mapped back to cyclic concept space in late layers.

We analyze these numerical representations by probing for "Fourier features", i.e., representations of sinusoidal functions that when composed together, by the Fourier transform, can be used to represent continuous maps (Fourier, 1807). Prior work has found that models represent numbers using Fourier features with base-10 periodicity, specifically, $T \in \{2, 5, 10, 20, 50, 100\}$ (Kantamneni & Tegmark, 2025; Zhou et al., 2024; 2025; Fu et al., 2026), but we show that these features are also used to calculate addition for months, weekdays, and hours. We isolate a sparse set of 28 MLP neurons that perform addition across all tasks: these neurons write to the Fourier features we found, and can be partitioned into clusters of neurons (Nikankin et al., 2025; Hanna et al., 2023; Gurnee et al., 2023) that compute the sum for each period. Our work provides an in-depth understanding of a shared addition mechanism that re-uses the same geometry across all tasks, regardless of the structure of the output domain.

2 Causal Abstraction over Cyclic Tasks

Task setup. We focus on months of the year (*what is six months after August?*), weekdays (*what day is three days after Friday?*), and 24-hour time (*It is currently 13:00. What time will it be in four hours?*). We evaluate Llama-3.1-8B on every combination of input concept and offset; offsets range from 1 to $2p$, where p is the cycle length of the concept (e.g., for months, p=12). We also include a standard addition task a+b= with $a, b \in \{1, \ldots, 100\}$. Restricted to in-cycle offsets (offset $\leq p$), Llama-3.1-8B achieves 82% accuracy on `months`, 92% on `weekdays`, and 97% on `hours`. Within this range, most errors come from prompts where the answer must "loop around" to the start of the cycle (e.g., *two months after December*; see App. C.2 for details). We also find that Llama-3.1-8B is surprisingly unable to perform explicit modular arithmetic for modulo 7, 12, and 24 (App. C.3).

Approach. Distributed alignment search (DAS; Geiger et al. 2023) is an optimization procedure that finds a subspace containing information relevant for LM behavior, represented as an abstract causal variable V in a causal model $\mathcal{A}$. We use DAS to localize such variables to low-rank subspaces of the residual stream. Let d_{model} be the residual stream dimension and k be the dimension of the subspace we want to find, which is a hyperparameter. Then, we learn a low-rank matrix $\mathbf{R} \in \mathbb{R}^{d_{\text{model}} \times k}$ with orthonormal columns (with the neural network $\mathcal{N}$ frozen) such that patching within the subspace defined by $\mathbf{R}$ from a counterfactual to

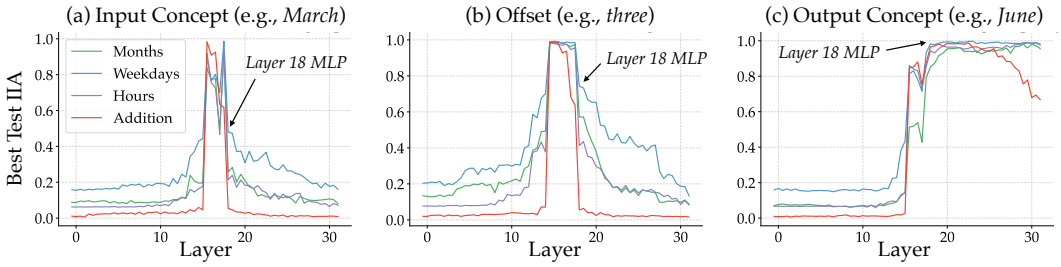


Figure 2: DAS results provide strong evidence that input concept and offset information are combined to produce the output concept in layer 18 at the last token position. (a) **Input concept** interchange accuracy (IIA) reaches near-100% immediately *before* the layer 18 MLP is applied; afterwards, IIA drops, indicating that this information has been "consumed." (b) **Offset** variables are copied to the last token position at layer 15 before being "consumed" after layer 18, when IIAs drop. (c) **Output concept** information materializes after layer 18, where IIA reaches 100%. We show the best subspace dimensions for each task ($k = 8$ for `months` and `weekdays`, and $k = 16$ for `hours` and `addition`). See App. F for full results.

an original prompt ($c \to o$) causes the model to predict a counterfactual output for o. The training objective is:

$$\text{CrossEntropy}\left(\mathcal{N}_{\mathbf{R}\leftarrow\mathcal{N}(c)}(o), \mathcal{A}_{V\leftarrow\mathcal{A}(c)}(o)\right), \tag{1}$$

where the abstract causal model $\mathcal{A}$ defines our expectation of what should happen when we patch V for any (c, o). See App. D for full definitions.

For example, we can train DAS on the `months` task to localize a k-dimensional subspace of the residual stream that encodes the input concept at a particular layer and token position. We measure success with *interchange intervention accuracy* (IIA), which in this case indicates whether patching within this subspace from e.g., *six months after* ***October***$\to$*two months after* ***April*** causes the model to output *December* (two + **October**). To patch within the subspace spanned by $\mathbf{R}$, we replace the original hidden state $\mathbf{h}_o$ with a new state

$$\mathbf{h} = \mathbf{h}_o + \mathbf{R}(\mathbf{R}^T\mathbf{h}_c - \mathbf{R}^T\mathbf{h}_o), \tag{2}$$

where $\mathbf{h}_c$ is the hidden state for the counterfactual prompt c. We use DAS to localize subspaces for the input concept, offset, and output concept at the last token position[2], training on the residual stream after each attention/MLP sublayer. All tasks are trained separately. We sweep across subspace dimensions $1 \leq k \leq 8$ for `months` and `weekdays`, and $1 \leq k \leq 16$ for `hours` and `addition`, reporting results for the dimension with the best performance (empirically, this is usually the largest k). From this point on, we refer to the subspace with the best test IIA as simply "the DAS subspace" for a given task and causal variable. See training details in App. F.

Evidence for a shared mechanism. Figure 2 shows results for the best subspace for each task across sublayers. Despite DAS being trained separately for each task, results are remarkably similar across domains. For cyclic tasks, input concept and offset variables can be causally isolated with >95% accuracy in the input to MLP 18. Immediately *after* this MLP, we cannot cleanly intervene on input arguments, e.g., IIA for input month drops by about 80 points, implying that the model has started to compute the final output. This pattern persists even for the `addition` task, which is not cyclic. Overlap between subspaces for the same variable across tasks also peaks in similar layers (App. F.4). *Taken together, these results point to a shared mechanism at layer 18 in the last token position for all tasks.*

Where is the circle? We replicate the experiment from Engels et al. (2025) and train "circular probes" for each task to recover the circular geometry that Llama-3.1-8B uses to

[2]We choose the last token position based on initial intervention experiments from App. E.

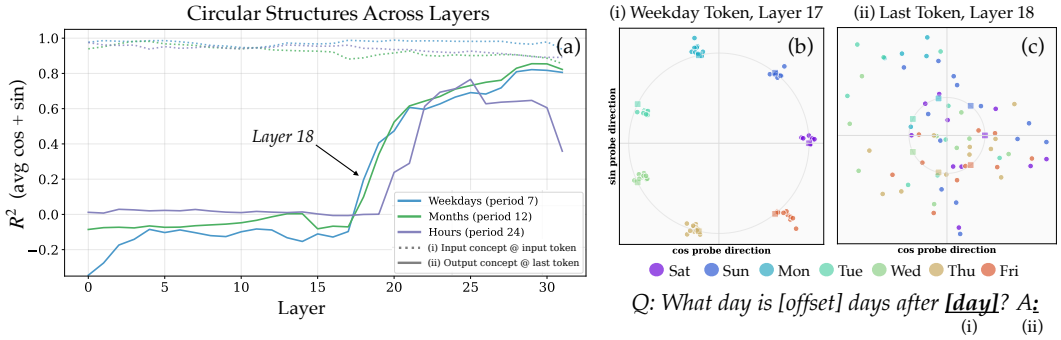


Figure 3: Probes reveal that circular structure (Engels et al., 2025) is not present in layer 18 at the final token, where the input concept and offset are combined. (a) We train circular probes for (i) the input concept at the input concept token position and (ii) the output concept at the final token position. Circular structure for (i) is consistently recovered across layers; see (b) for an example of the weekdays circle at layer 17. In contrast, circular structure for (ii) emerges only in later layers. Notably, at layer 18, we do not yet see circular structure for the output concept, despite results in Section 2 indicating that the inputs have already been combined at this stage (see (c)).

represent cyclic concepts: in particular, we search for these circular structures in a PCA-reduced activation subspace within the top five principal components (see App. I.4 for details). If Llama-3.1-8B computed e.g., *three days after Wednesday* by somehow rotating along the weekday circle discovered by Engels et al. (2025), we would expect to find this structure in the layers relevant for computation. Although we can recover circular structure at the input concept token position across layers (Figure 3), we cannot reliably probe for circular structure at the *final* token position until layers 22-25, even though Figure 2c shows that the output concept has started to emerge by layer 18. This suggests that the causal mechanism at layer 18 does *not* rely on task-specific circular geometry.

3 The Shared Mechanism is Addition

Given the similarity of DAS results between `addition` and cyclic tasks (Section 2), one hypothesis is that Llama-3.1-8B uses base-10 addition for cyclic tasks. Under this hypothesis, the output concept subspace in a forward pass for *6+8=* should contain the same information as it does for *six months after August*. We show that this is in fact true: we can reliably patch between `addition` prompts and cyclic tasks with predictable effects, suggesting that cyclic tasks are represented as numbers around layer 18.

Patching from addition to cyclic tasks. First, we patch from `addition` into cyclic tasks, with the expectation that the sum from the addition task will be decoded into a concept, e.g., patching from *6+8=* to any `months` prompt should cause the model to output *February*, as (6+8) mod 12 = 2 and February is the second month. At every layer, we patch within the union of the `addition` and target output DAS subspaces. Figure 4 shows that for all three cyclic tasks, Llama-3.1-8B converts the `addition` sum to a cyclic concept with at least 40% accuracy. For `months`, performance is comparable to the model's accuracy in a clean run. These results imply that Llama-3.1-8B uses numerical representations in middle layers for cyclic tasks.

Patching from cyclic tasks to addition. Next, we perform the opposite experiment: we patch from cyclic tasks into standard addition prompts. We expect that patching from *What is six months after August?* to any prompt *a+b=* will cause Llama-3.1-8B to output the number *14*, since August is the eighth month and 6+8=14. Again, we patch within the union of per-task output spaces at each layer. The clean diagonal we observe in Figure 5b shows that prompts with the same output month but different pre-modulo sums (e.g., 3,15→March) always cause the model to output *different* numbers under patching. Even

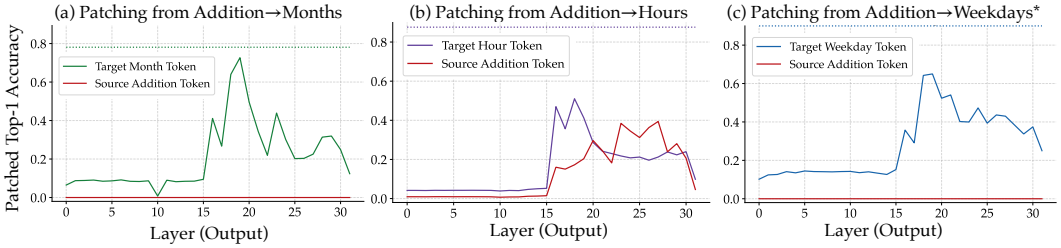


Figure 4: Patching from `addition` to cyclic tasks within the shared subspace of both tasks shows that base-10 numerical representations at layer 18 are converted into cyclic concepts. (e.g., *6+8=14*→*February*). Patching at layer 18 consistently causes the model to output the predicted concept. This patch does not cause the model to output the source number token (red line), except for `hours`, which also uses number tokens. Dotted lines indicate clean model performance for equivalent number ranges. We enumerate weekdays based on results from other experiments; see App. G.

for `months` prompts that the model answers incorrectly, like *twenty months after October*, patching into `addition` reveals that the model had computed the correct sum (e.g., 30) in 63% of interventions. This means that for many `months` prompts, Llama-3.1-8B can internally compute the sum, but then struggles to map that sum to the correct output month. Notably, even though we are patching from `months`, this intervention never increases the model's predicted probability for a month. These results hold for `hours` and `weekdays` (App. H), implying that numerical representations are indeed computed in cyclic forward passes.

Figure 5b reveals an "echo" pattern: patching into an addition context also raises the probability of 100 + the target token, occasionally surpassing the target token itself (e.g., patching *six months after August* increases the probability of 14 and 114). The same pattern arises for `weekdays` and `hours`. A possible explanation is that Llama-3.1-8B never needs to represent the hundreds place when reasoning over cyclic concepts, since the relevant sums are always small, consistent with the finding that larger Fourier periodicities are causally unimportant for these tasks (App. I.3).

4 Fourier Probes Trained on Addition Can Steer Cyclic Tasks

If the model uses addition for these cyclic tasks, prior work suggests it may use "Fourier features" (Nanda et al., 2023; Zhong et al., 2023; Zhou et al., 2024; Kantamneni & Tegmark, 2025; Heinzerling & Inui, 2024). We first replicate results from prior work, training probes on the `addition` task that best recover periods $T \in \{2, 5, 10, 20, 50, 100\}$ (Levy & Geva, 2025; Kantamneni & Tegmark, 2025; Fu et al., 2026; Gould et al., 2023; Zhu et al., 2025; Kadlčík et al., 2025). Then, we show that these Fourier probes generalize to cyclic tasks by using them to steer outputs for `months`, `weekdays`, and `hours`. Because probes do not always find causal features (Belinkov, 2022; Sharkey et al., 2025), the fact that these probes trained on addition have a causal effect on unseen tasks is strong evidence for the importance of Fourier features in this setting.

Training Fourier probes. Let $\mathbf{h}^{(l)}_{a+b}$ be a hidden state at layer l for the last token in the prompt *a+b=*. For each period $1 \leq T \leq 150$, we train two affine probes to follow sine and cosine harmonics by minimizing the following losses:

$$\mathsf{MSE}\left(\langle \mathbf{w}^{(l,T)}_{\sin} \mathbf{h}^{(l)}_{a+b} \rangle + b^{(l,T)}_{\sin}, \sin\left(\tfrac{2\pi(a+b)}{T}\right)\right), \ \mathsf{MSE}\left(\langle \mathbf{w}^{(l,T)}_{\cos} \mathbf{h}^{(l)}_{a+b} \rangle + b^{(l,T)}_{\cos}, \cos\left(\tfrac{2\pi(a+b)}{T}\right)\right). \tag{3}$$

Here, $\mathbf{w}^{(l,T)}_{\sin}, \mathbf{w}^{(l,T)}_{\cos} \in \mathbb{R}^d$ and $b^{(l,T)}_{\sin}$, $b^{(l,T)}_{\cos}$ are scalar biases. For example, given the prompt *8+5=13*, our probes would be trained to recover the gold labels $\sin(13(2\pi/T))$ and $\cos(13(2\pi/T))$ from $\mathbf{h}^{(l)}_{8+5}$. When we train these probes for the `addition` task, we find

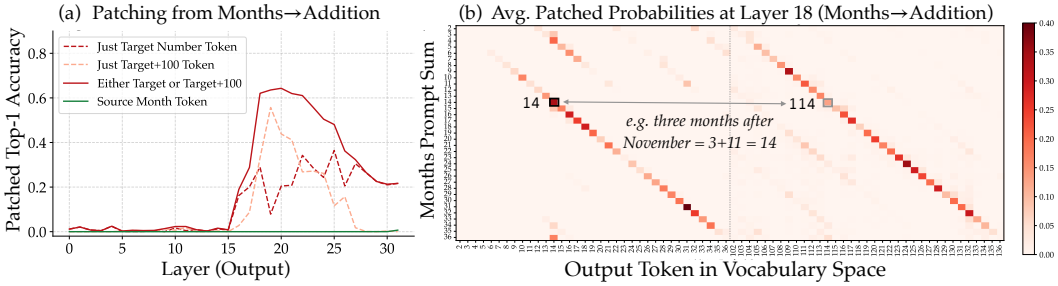


Figure 5: Patching from `months` → `addition` within the shared subspace of both tasks shows that Llama-3.1-8B represents cyclic concepts using base-10 numerical representations at layer 18 (e.g., patching from *six months after August* into `addition` prompts causes the model to output 14, suggesting that the model computed *6+8* as an intermediate step). We observe a surprising +100 echo, where the expected sum is sometimes output as 114 as well as 14. (a) Including +100 echoes, this intervention causes Llama-3.1-8B to output the predicted sum in over 60% of examples, without ever causing the model to output a month. (b) We observe a clean diagonal, implying that the model distinguishes between, e.g., 9 and 21 at this point (even though they both eventually map to *September*). We show similar results for other cyclic tasks in App. H.

that periods $T \in \{2, 5, 10, 20, 50, 100\}$ can be reliably recovered. These periods emerge in middle layers, in agreement with our causal analysis. See App. I.1 for full results. Applying probes to layer 18 activations reveals circular structures with base-10 periodicities across all tasks (Figure 6). Although the hours, months, and weekdays tasks have natural periods (e.g., 24, 12, and 7), the model represents the intermediate sums in these tasks using a base-10 system.

Steering on cyclic tasks with Fourier probes. To show that these features are causally important, we use the Fourier probes trained on `addition` to steer outputs for cyclic tasks. In essence, we use the probe directions to set the model's pre-modulo sum n to some counterfactual n', expecting the model to then map the counterfactual sum n' to concept space. For example, for the prompt *What is four months after March?*, the original sum is n=7. If we steer to a target sum of n'=6, we expect the model to output *June* instead of *July*.

Let $\mathbf{h}^{(l)}_{\text{Cyclic}(a,b)}$ be the residual stream representation at layer l for the last token in a cyclic task prompt. To steer the model to a target sum n' for this prompt, we modify the residual stream so that its Fourier coefficients match the target phase for each period T. For each probe with period T, we compute the new desired phase $\theta_T^{n'} = n'(2\pi/T)$ and cache the empirically observed radius r_T from the original forward pass:

$$\hat{s} = \mathbf{w}^{(l,T)}_{\sin}\mathbf{h}^{(l)}_{\text{Cyclic}(a,b)} + b^{(l,T)}_{\sin}, \qquad \hat{c} = \mathbf{w}^{(l,T)}_{\cos}\mathbf{h}^{(l)}_{\text{Cyclic}(a,b)} + b^{(l,T)}_{\cos}, \qquad r_T = \sqrt{\hat{s}^2 + \hat{c}^2}.$$

Then, we construct a steered activation with scaling factor α that increases the radius proportional to the original values:

$$\tilde{\mathbf{h}}^{(l)}_{\text{Cyclic}(a,b)} \leftarrow \mathbf{h}^{(l)}_{\text{Cyclic}(a,b)} + \left(\alpha\, r_T \sin(\theta_T^{n'}) - \hat{s}\right)\hat{\mathbf{w}}^{(l,T)}_{\sin} + \left(\alpha r_T \cos(\theta_T^{n'}) - \hat{c}\right)\hat{\mathbf{w}}^{(l,T)}_{\cos}, \quad (4)$$

where $\hat{\mathbf{w}}^{(l,T)}_{\sin}, \hat{\mathbf{w}}^{(l,T)}_{\cos}$ are normalized vectors for the probes. To determine which T to steer on for each task, we measure overlap of Fourier probes with DAS subspaces; this means that e.g., period $T = 50$ is not used for the weekday task. For details, see App. I.3.

Figure 7 shows the resulting output distributions for steering every task at layer 18. Each row corresponds to the mean output distribution after steering all prompts in the task toward a specific target value; successful steering should yield a strong diagonal pattern, indicating that probability mass shifts to the intended target. In all experiments, we use a

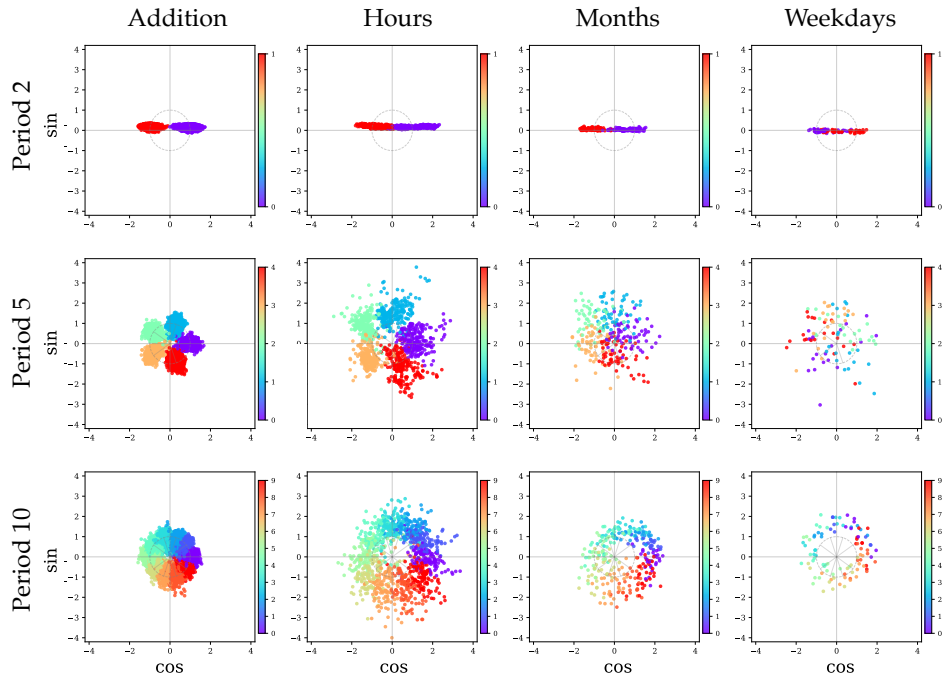


Figure 6: Applying Fourier probes trained on the `addition` task to layer 18 activations reveals circular structures with base-10 periodicities across all tasks; here, we show $T \in \{2, 5, 10\}$. Although hours, months, and weekdays have their own natural periods (e.g., 24, 12, and 7), the model represents intermediate sums in these tasks using a base-10 system.

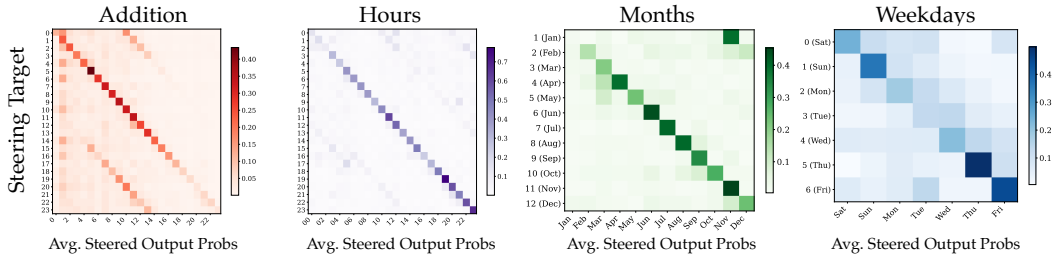


Figure 7: Steering with Fourier probes at layer 18 shows that Fourier features identified for the *addition* task are causal across all tasks. For each original prompt, we steer towards a numeric target n' by modifying the Fourier features to encode that value. Each row shows the average output distribution after steering all prompts toward a specific target. A strong diagonal indicates a successful intervention.

scaling factor of $\alpha = 10$, which controls the strength of the intervention. The need for this high α suggests that these features alone may not fully override downstream computation.

5 Decomposing The Shared MLP Addition Module into Subcircuits

If Llama-3.1-8B represents the output sum using Fourier features, how do its parameters interact with this geometry? In this section, we study a small set of 28 MLP neurons[3] at layer 18 that use Fourier features to compute the sum of two numbers.

[3]See Merrill et al. (2023) that study competition between sparse and dense subnetworks during grokking and offers a complementary perspective on why sparse arithmetic circuits could emerge.

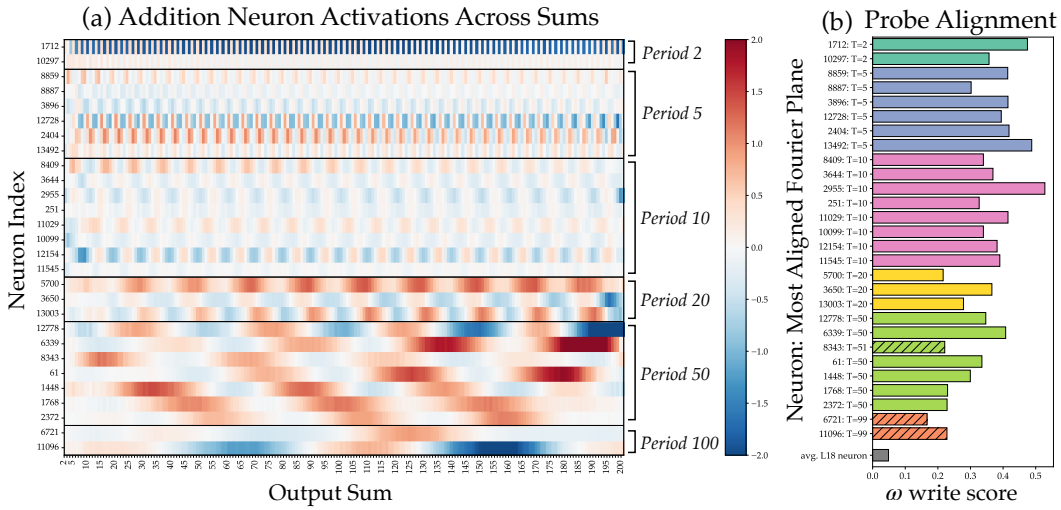


Figure 8: Addition neurons are sparse and can be grouped by the Fourier periodicities found in Section 4. (a) Activations of neurons in $\mathcal{N}_{\text{add}}$ averaged across prompts with the same output sum; colormap is clipped to [-2,2] for visibility. (b) For each neuron, we report the pair of Fourier probes that this neuron is most aligned with across all $2 \leq T \leq 150$. Every neuron writes to the Fourier plane from Section 4 that corresponds to its activation pattern across examples. As a baseline (gray), we report maximum Fourier plane alignment averaged across every MLP neuron at L18.

5.1 Identifying Addition Neurons

We focus on the MLP at layer 18, which is a gated MLP (Shazeer, 2020). Thus, if d_{mlp} is the MLP dimension (number of neurons) and d_{model} is the residual stream dimension, then for a hidden state $\mathbf{h} \in \mathbb{R}^{d_{\text{model}}}$ and $\mathbf{W}_{\text{gate}}, \mathbf{W}_{\text{up}}, \mathbf{W}_{\text{down}} \in \mathbb{R}^{d_{\text{mlp}} \times d_{\text{model}}}$, the MLP is defined as

$$\text{MLP}(\mathbf{h}) = (\text{SiLU}(\mathbf{W}_{\text{gate}}\mathbf{h}) \odot \mathbf{W}_{\text{up}}\mathbf{h})\mathbf{W}_{\text{down}} \tag{5}$$

where $\odot$ is element-wise multiplication. We refer to a single neuron n_i as the collection of three vectors: the gate, up, and down weights $\mathbf{g}_i, \mathbf{u}_i, \mathbf{d}_i \in \mathbb{R}^d$ that correspond to index i.

We want to find neurons that write to causally-important subspaces found by DAS; i.e., that are causally important for computing the sum (Geva et al., 2021; Arora et al., 2025; Hanna et al., 2023). For each task, we look for MLP neurons at layer 18 that write to the DAS output concept subspace for that task, spanned by $\mathbf{R}_{\text{task}} \in \mathbb{R}^{d \times k}$ from Section 2. For a given neuron's down weight $\mathbf{d}_i$, we calculate

$$\omega_{\text{task}}(\mathbf{d}_i) = ||\mathbf{R}_{\text{task}}\mathbf{d}_i||/||\mathbf{d}_i||. \tag{6}$$

Empirically, only a small number of MLP neurons at layer 18 have high ω. We plot the distribution of ω for each task in Figure 41; with a threshold of $\omega > 0.4$, we identify 16 neurons for `months`, 15 neurons for `weekdays`, 26 for `hours`, and 28 for `addition`. All neurons important for cyclic tasks are a strict subset of those important for `addition`, except for a single outlying `hours` neuron, which we ignore. See App. J.1 for details. For every pair of tasks, Pearson correlation of ω scores is high, with $r \geq 0.70$ ($p = 0.0$).

Neuron ablations. We focus on the set of 28 addition neurons in layer 18 that satisfy our threshold, $\mathcal{N}_{\text{add}} = \{n_i | \omega_{\text{addition}}(\mathbf{d}_i) > 0.4\}$. When these neurons are zero-ablated, model accuracy decreases significantly (e.g., from 95%→24% for `addition`). Despite making up only 0.2% of MLP neurons in layer 18, these neurons explain most of the computation in this sublayer: ablating all other layer 18 MLP neurons except for $\mathcal{N}_{\text{add}}$ retains most of the model's performance (e.g., 95%→86% for `addition`). We also find that these neurons are causally important for unseen prompt templates; see Table 6 for full ablations.

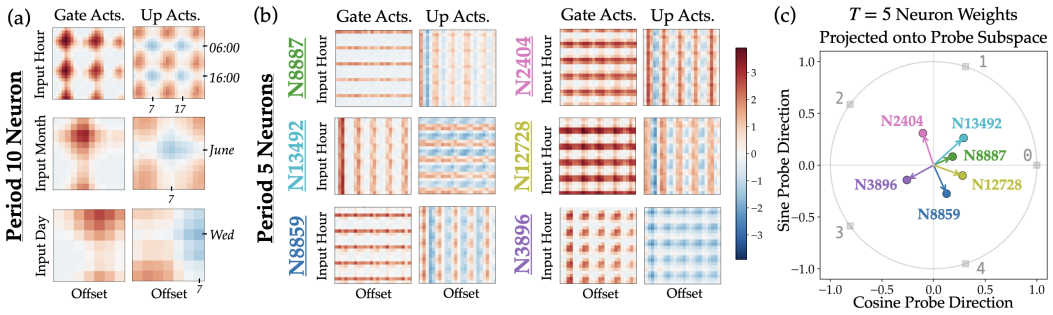


Figure 9: Addition neuron activations organized by input concept and offset, separated into gate and up activations. (a) Activations for a period 10 neuron n_{8409}. This neuron has the same activation pattern across all cyclic tasks: e.g., its up projection is negative for *seven hours after 06:00* as well as *seven months after June*. Because it is period 10, it also activates for *seventeen hours after 16:00*. (b) Activations for all period 5 neurons for the `hours` task. (c) Down projection rows for all period 5 neurons, projected onto the Fourier plane for $T = 5$.

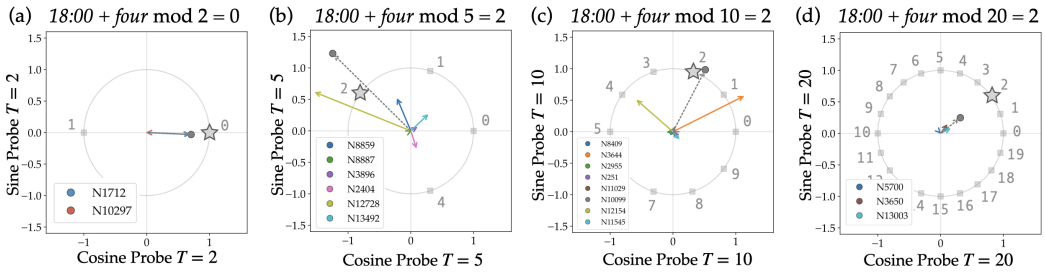


Figure 10: The model computes the sum *18+4=22* on multiple orthogonal planes, each encoding a different modulo. We visualize all addition neurons for periods $T \in \{2, 5, 10, 20\}$ projected onto their respective Fourier planes: arrows indicate each neuron's down projection row $\mathbf{d}_i$ scaled by its activation for the prompt *four hours after 18:00*. The gray dotted line indicate the sum of these vectors, and gray stars indicate the ground truth label.

Partitioning neurons according to Fourier features. First, we visualize the average activation of each neuron $n_i \in \mathcal{N}_{\text{add}}$ for every possible output sum in the `addition` task in Figure 8a, finding that neurons fire periodically across sums; all have similar activation patterns for `months`, `weekdays`, and `hours` (Figure 43). We can partition these neurons into disjoint clusters that activate with period $T \in \{2, 5, 10, 20, 50, 100\}$. In fact, each cluster of neurons writes to Fourier directions with the same period: we calculate how much each $\mathbf{d}_i$ overlaps with the span of Fourier probes $\mathbf{w}_{\text{sin}}^{(T)}, \mathbf{w}_{\text{cos}}^{(T)}$ across values of T (see Eq. 6). Figure 8b shows that each cluster of neurons activating with period T also writes to Fourier features with the same period. Measuring absolute cosine similarity of $\mathbf{d}_i$, we observe that these clusters are perfectly orthogonal to each other (Fig. 44).

Larger periods are possibly missing. Although these neurons are crucial to explain layer 18 computation, we are likely missing neurons with larger periods. When we zero-ablate all other neurons at layer 18 except for $\mathcal{N}_{\text{add}}$, the small drop in accuracy from 95%→86% comes mostly from examples where both summands are large (Figure 42), suggesting that our threshold may exclude some neurons with large T.

5.2 How Addition Neurons Compute the Sum

First, we analyze activation patterns for addition neurons. We arrange activations in grid format, where each square corresponds to a neuron's activation for a single prompt. Rows correspond to prompts with the same input concept, while columns correspond to prompts with the same offset. We separate activations into $\mathbf{W}_{\text{gate}}$ and $\mathbf{W}_{\text{up}}$ activations, before they are element-wise multiplied. Observe that all neurons have similar activation patterns across

cyclic tasks: for example, Figure 9a shows a consistent pattern for n_{8409} for `hours`, `months`, and `weekdays`. Additionally, if the cluster that a neuron belongs to is period T, it fires with periodicity T over its inputs (e.g., n_{8409} fires for *06:00* and *16:00*). See App. J.3.

Second, we observe two types of neuron activation patterns: *split* patterns and *mixed* patterns. If a neuron has a *split* activation pattern, this means that it is using its gate vector $\mathbf{g}_i$ to read from one summand and its up vector $\mathbf{u}_i$ to read from the other. We hypothesize that this split behavior helps the model take advantage of element-wise multiplication to combine information from two disparate subspaces. For example, in Figure 9b, n_{8887} has a *split* activation pattern for cyclic tasks, where its gate activation fires whenever the input concept is a multiple of five (e.g., *15:00*, horizontal stripes), and its up activation fires whenever the offset mod 5 = 1 (e.g., *in six hours, in eleven hours*, vertical stripes). *Mixed* neurons use both vectors to read both input arguments. Figure 9a shows a neuron with mixed activation patterns: its gate and up activations respond to both summands. We find that approximately 17/28 neurons have split behavior for cyclic tasks (App. J.4). For addition, all neurons have mixed activation patterns, possibly related to the fact that IIA for `addition` at this sublayer is only about 60%, indicating that input arguments are more "fused."

Third, we observe that calculation of the sum is performed in a distributed manner, where neurons with the same periodicity combine to predict the sum within that periodicity. For example, Figure 9c shows every period 5 neuron's down projection row projected to the $T = 5$ Fourier plane alongside its activations: in a forward pass, these down projection rows are scaled by their respective activations such that the sum of these vectors "points to" the correct output. Figure 10 shows real neuron activations for periods $T \in \{2, 5, 10, 20\}$ for the prompt *four hours after 18:00*. Within each period, neurons work together, summing to a vector with an angle encoding the output $4 + 18 \pmod{T}$.

Why use several periods? One might wonder why Llama-3.1-8B needs to represent smaller periods at all if, e.g., $T = 100$ could theoretically differentiate between numbers up to 100. As Zhou et al. (2024) explain, smaller periods help to sharpen the model's representations. Take $T = 2$ as an example. We find two parity neurons that play complementary roles, writing in opposite directions: n_{10297} is an "odd neuron" that fires for *even+odd=odd*, while n_{1712} is an "even neuron" that fires only for *even+even=even*.[4] Figure 48b shows that if we zero-ablate *only* these two neurons at the last token position, probability mass shifts away from the correct answer and towards neighboring answers, "blurring" the model's outputs.

6 Conclusion

In this work, we discovered a generic addition mechanism that is re-used across several domains. Although prior work shows that transformers can compute modular addition in one step, we show that Llama-3.1-8B implements a two-step approach: it first uses a base-10 addition mechanism to compute a sum using Fourier features before mapping that number back to a concept-specific period (e.g., 7, 12, or 24). Furthermore, we were able to isolate a sparse set of 28 neurons writing to specific Fourier features of different periods. See Appendix A for extended discussion of related work. Our work highlights the interplay between causal analysis and geometric understanding: causal analysis enabled us to localize subspaces containing information used by the model for computation, and analyzing the geometry of the representations within those subspaces provided deeper insight into how these computations are implemented.

Acknowledgments

We would like to thank Thomas Icard, Vasudev Shyam, Jing Huang and Ren Makino for helpful discussions throughout the course of this project. Additionally, we thank Yonatan Belinkov, David Bau, Arjun Khurana, and Andy Arditi for feedback on our initial drafts.

[4] Cosine similarity of $\mathbf{d}_{10297}, \mathbf{d}_{1712}$ in the output subspace for `addition` is 0.997, but activations for n_{1712} are always negative; thus, these two neurons write in opposite directions.

References

Omer Antverg and Yonatan Belinkov. On the pitfalls of analyzing individual neurons in language models, 2022. URL https://arxiv.org/abs/2110.07483.

Aryaman Arora, Dan Jurafsky, and Christopher Potts. CausalGym: Benchmarking causal interpretability methods on linguistic tasks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 14638–14663, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.785.

Aryaman Arora, Zhengxuan Wu, Jacob Steinhardt, and Sarah Schwettmann. Language model circuits are sparse in the neuron basis, November 2025.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. Transactions of the Association for Computational Linguistics, 6:483–495, 2018. doi: 10.1162/tacl_a_00034. URL https://aclanthology.org/Q18-1034/.

Xiaoyan Bai, Itamar Pres, Yuntian Deng, Chenhao Tan, Stuart M. Shieber, Fernanda B. Viégas, Martin Wattenberg, and Andrew Lee. Why can't transformers learn multiplication? reverse-engineering reveals long-range dependency pitfalls. CoRR, abs/2510.00184, 2025. doi: 10.48550/ARXIV.2510.00184. URL https://doi.org/10.48550/arXiv.2510.00184.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. Identifying and controlling important neurons in neural machine translation, 2018a. URL https://arxiv.org/abs/1811.01157.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks, 2018b. URL https://arxiv.org/abs/1811.10597.

Sander Beckers and Joseph Halpern. Abstracting causal models. In AAAI Conference on Artificial Intelligence, 2019.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. Computational Linguistics, 48(1):207–219, 2022.

Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. LEACE: perfect linear concept erasure in closed form. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/d066d21c619d0a78c5b557fa3291a8f4-Abstract-Conference.html.

Michel Besserve, Arash Mehrjou, Rémy Sun, and Bernhard Schölkopf. Counterfactuals uncover the modular structure of deep generative models. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020. URL https://openreview.net/forum?id=SJxDDpEKvH.

Sasha Brenner, Thomas R Knösche, and Nico Scherf. Grid-world representations in transformers reflect predictive geometry. arXiv preprint arXiv:2603.16689, 2026.

Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, Ludwig Schubert, Chelsea Voss, Ben Egan, and Swee Kiat Lim. Thread: Circuits. Distill, 2020. doi: 10.23915/distill.00024. https://distill.pub/2020/circuits.

Valérie Costa, Thomas Fel, Ekdeep Singh Lubana, Bahareh Tolooshams, and Demba Ba. From flat to hierarchical: Extracting sparse representations with matching pursuit, 2025. URL https://arxiv.org/abs/2506.03093.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. Recurrent neural networks learn to store and generate sequences using non-linear representations. In The Twelfth International Conference on Learning Representations, 2024.

Róbert Csordás, Christopher Potts, Christopher D. Manning, and Atticus Geiger. Recurrent neural networks learn to store and generate sequences using non-linear representations. In Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP, 2024. URL https://aclanthology.org/2024.blackboxnlp-1.17.

Xander Davies, Max Nadeau, Nikhil Prakash, Tamar Rott Shaham, and David Bau. Discovering variable binding circuitry with desiderata. CoRR, abs/2307.03637, 2023. doi: 10.48550/ARXIV.2307.03637. URL https://doi.org/10.48550/arXiv.2307.03637.

Fenil R Doshi, Thomas Fel, Talia Konkle, and George Alvarez. Bi-orthogonal factor decomposition for vision transformers. arXiv preprint arXiv:2601.05328, 2026.

Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Pareto frontiers in deep feature learning: Data, compute, width, and luck. Advances in Neural Information Processing Systems, 36:48021–48034, 2023.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. In Proceedings of the 2020 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. Association for Computational Linguistics, November 2020. doi: 10.18653/v1/W18-5426.

Joshua Engels, Eric J. Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear, 2025. URL https://arxiv.org/abs/2405.14860.

Thomas Fel, Binxu Wang, Michael A. Lepori, Matthew Kowal, Andrew Lee, Randall Balestriero, Sonia Joseph, Ekdeep Singh Lubana, Talia Konkle, Demba E. Ba, and Martin Wattenberg. Into the rabbit hull: From task-relevant concepts in DINO to minkowski geometry. CoRR, abs/2510.08638, 2025. doi: 10.48550/ARXIV.2510.08638. URL https://doi.org/10.48550/arXiv.2510.08638.

Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=zb3b6oKO77.

Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart Shieber, Tal Linzen, and Yonatan Belinkov. Causal analysis of syntactic agreement mechanisms in neural language models. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1828–1843, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.144. URL https://aclanthology.org/2021.acl-long.144.

Jean Baptiste Joseph Fourier. Théorie de la propagation de la chaleur dans les solides. 1807.

Deqing Fu, Tianyi Zhou, Mikhail Belkin, Vatsal Sharan, and Robin Jia. Convergent evolution: How different language models learn similar number representations, 2026. URL https://arxiv.org/abs/2604.20817.

Hiroki Furuta, Gouki Minegishi, Yusuke Iwasawa, and Yutaka Matsuo. Interpreting grokked transformers in complex modular arithmetic. arXiv preprint, 2024.

Atticus Geiger, Kyle Richardson, and Christopher Potts. Neural natural language inference models partially embed theories of lexical entailment and negation. In Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, 2020. URL https://www.aclweb.org/anthology/2020.blackboxnlp-1.16.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In Proceedings of the 35th International Conference on Neural Information Processing Systems, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas F. Icard, and Noah D. Goodman. Finding alignments between interpretable causal variables and distributed neural representations. ArXiv, abs/2303.02536, 2023. URL https://api.semanticscholar.org/CorpusID:257365438.

Atticus Geiger, Jacqueline Harding, and Thomas Icard. How causal abstraction underpins computational explanation, 2025a. URL https://arxiv.org/abs/2508.11214.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. Journal of Machine Learning Research, 26(83):1–64, 2025b. URL http://jmlr.org/papers/v26/23-0058.html.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. Journal of Machine Learning Research, 2025c. URL http://jmlr.org/papers/v26/23-0058.html.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 5484–5495, 2021.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models, 2023. URL https://arxiv.org/abs/2304.14767.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem H. Zuidema. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In Tal Linzen, Grzegorz Chrupala, and Afra Alishahi (eds.), Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018, pp. 240–248. Association for Computational Linguistics, 2018. doi: 10.18653/v1/w18-5426. URL https://doi.org/10.18653/v1/w18-5426.

Pulkit Gopalani and Wei Hu. What happens during the loss plateau? understanding abrupt learning in transformers. arXiv preprint arXiv:2506.13688, 2025.

Pulkit Gopalani, Ekdeep S Lubana, and Wei Hu. Abrupt learning in transformers: A case study on matrix completion. Advances in Neural Information Processing Systems, 37: 55053–55085, 2024.

Rhys Gould, Euan Ong, George Ogden, and Arthur Conmy. Successor heads: Recurring, interpretable attention heads in the wild. arXiv preprint, 2023.

Satchel Grant, Noah D. Goodman, and James L. McClelland. Emergent symbol-like number variables in artificial neural networks, 2025. URL https://arxiv.org/abs/2501.06141.

Clément Guerner, Anej Svete, Tianyu Liu, Alexander Warstadt, and Ryan Cotterell. A geometric notion of causal probing. CoRR, abs/2307.15054, 2023. doi: 10.48550/ARXIV.2307.15054. URL https://doi.org/10.48550/arXiv.2307.15054.

Yoav Gur-Arieh, Roy Mayan, Chen Agassy, Atticus Geiger, and Mor Geva. Enhancing automated interpretability with output-centric feature descriptions. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2025. URL https://aclanthology.org/2025.acl-long.288/.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. Transactions on Machine Learning Research, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=JYs1R9IMJr.

Wes Gurnee, Emmanuel Ameisen, Isaac Kauvar, Julius Tarng, Adam Pearce, Chris Olah, and Joshua Batson. When models manipulate manifolds: The geometry of a counting task. Transformer Circuits Thread, 2025. URL https://transformer-circuits.pub/2025/linebreaks/index.html.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/efbba7719cc5172d175240f24be11280-Abstract-Conference.html.

Benjamin Heinzerling and Kentaro Inui. Monotonic representation of numeric properties in language models. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, 2024.

Jing Huang, Zhengxuan Wu, Christopher Potts, Mor Geva, and Atticus Geiger. Ravel: Evaluating interpretability methods on disentangling language model representations. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2024. URL https://aclanthology.org/2024.acl-long.470.

David Hume. An Enquiry Concerning Human Understanding. A. Millar, London, 1748.

Marek Kadlčík, Michal Štefánik, Timothee Mickus, Michal Spiegel, and Josef Kuchař. Pre-trained language models learn remarkably accurate representations of numbers. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, 2025.

Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition, 2025. URL https://arxiv.org/abs/2502.00873.

Dhruva Karkada, Daniel J. Korchinski, Andres Nava, Matthieu Wyart, and Yasaman Bahri. Symmetry in language statistics shapes the geometry of model representations, 2026. URL https://arxiv.org/abs/2602.15029.

Daniel J Korchinski, Dhruva Karkada, Yasaman Bahri, and Matthieu Wyart. On the emergence of linear analogies in word embeddings. arXiv preprint arXiv:2505.18651, 2025.

Daniel Kunin, Giovanni Luca Marchetti, Feng Chen, Dhruva Karkada, James B Simon, Michael R DeWeese, Surya Ganguli, and Nina Miolane. Alternating gradient flows: A theory of feature learning in two-layer neural networks. arXiv preprint arXiv:2506.06489, 2025.

Amit Arnold Levy and Mor Geva. Language models encode numbers using digit representations in base 10. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 2: Short Papers, Albuquerque, New Mexico, April 29 - May 4, 2025, pp. 385–395. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.NAACL-SHORT.33. URL https://doi.org/10.18653/v1/2025.naacl-short.33.

Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure, 2017. URL https://arxiv.org/abs/1612.08220.

Llama-Team. The llama 3 herd of models. CoRR, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

Ekdeep Singh Lubana, Can Rager, Sai Sumedh R. Hindupur, Valerie Costa, Greta Tuckute, Oam Patel, Sonia Krishna Murthy, Thomas Fel, Daniel Wurgaft, Eric J. Bigelow, Johnny Lin, Demba Ba, Martin Wattenberg, Fernanda Viegas, Melanie Weber, and Aaron Mueller. Priors in time: Missing inductive biases for language model interpretability, 2025. URL https://arxiv.org/abs/2511.01836.

Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks. Advances in neural information processing systems, 32, 2019.

Giovanni Luca Marchetti, Daniel Kunin, Adele Myers, Francisco Acosta, and Nina Miolane. Sequential group composition: A window into the mechanics of deep learning. arXiv preprint arXiv:2602.03655, 2026.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. Advances in Neural Information Processing Systems, 36, 2022. arXiv:2202.05262.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023. URL https://openreview.net/pdf?id=MkbcAHIYgyS.

William Merrill, Nikolaos Tsilivis, and Aman Shukla. A tale of two circuits: Grokking as competition of sparse and dense subnetworks. arXiv preprint, 2023.

Alexander Modell, Patrick Rubin-Delanchy, and Nick Whiteley. The origins of representation manifolds in large language models, 2025. URL https://arxiv.org/abs/2505.18235.

Depen Morwani, Benjamin L. Edelman, Cosmin Oncescu, Rosie Zhao, and Sham M. Kakade. Feature emergence via margin maximization: Case studies in algebraic tasks. In The Twelfth International Conference on Learning Representations, 2024.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fiotto-Kaufman, Tal Haklay, Michael Hanna, Jing Huang, Rohan Gupta, Yaniv Nikankin, Hadas Orgad, Nikhil Prakash, Anja Reusch, Aruna Sankaranarayanan, Shun Shao, Alessandro Stolfo, Martin Tutek, Amir Zur, David Bau, and Yonatan Belinkov. Mib: A mechanistic interpretability benchmark, 2025. URL https://arxiv.org/abs/2504.13151.

Aaron Mueller, Jannik Brinkmann, Millicent Li, Samuel Marks, Koyena Pal, Nikhil Prakash, Can Rager, Aruna Sankaranarayanan, Arnab Sen Sharma, Jiuding Sun, Eric Todd, David Bau, and Yonatan Belinkov. The quest for the right mediator: Surveying mechanistic interpretability for nlp through the lens of causal mediation analysis. Computational Linguistics, pp. 1–48, 02 2026. ISSN 0891-2017. doi: 10.1162/COLI.a.572. URL https://doi.org/10.1162/COLI.a.572.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, 2023. URL https://arxiv.org/abs/2301.05217.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics, 2025. URL https://arxiv.org/abs/2410.21272.

Kento Nishi, Rahul Ramesh, Maya Okawa, Mikail Khona, Hidenori Tanaka, and Ekdeep Singh Lubana. Representation shattering in transformers: A synthetic study with knowledge editing. arXiv preprint arXiv:2410.17194, 2024.

Core Francisco Park, Andrew Lee, Ekdeep Singh Lubana, Yongyi Yang, Maya Okawa, Kento Nishi, Martin Wattenberg, and Hidenori Tanaka. Iclr: In-context learning of representations. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), International Conference on Learning Representations, volume 2025, pp. 53258–53284, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/83fe5a77502e3d4cfab5960aed0ee6c3-Paper-Conference.pdf.

Kiho Park, Yo Joong Choe, Yibo Jiang, and Victor Veitch. The geometry of categorical and hierarchical concepts in large language models. In The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025. OpenReview.net, 2025b. URL https://openreview.net/forum?id=bVTM2QKYuA.

Michael Pearce, Elana Simon, Michael Byun, and Daniel Balsam. Finding the tree of life in evo 2. Goodfire, August 2025.

Judea Pearl. Probabilities of causation: Three counterfactual interpretations and their identification. Synthese, 121(1):93–149, 1999.

Judea Pearl. Direct and indirect effects, 2001. URL https://arxiv.org/abs/1301.2300.

Mateusz Piotrowski, Paul M Riechers, Daniel Filan, and Adam Shai. Constrained belief updating and geometric structures in transformer representations. In NeurIPS 2024 Workshop on Symmetry and Geometry in Neural Representations, 2024.

Nikhil Prakash, Natalie Shapira, Arnab Sen Sharma, Christoph Riedl, Yonatan Belinkov, Tamar Rott Shaham, David Bau, and Atticus Geiger. Language models use lookbacks to track beliefs, 2025. URL https://arxiv.org/abs/2505.14685.

Lucas Prieto, Edward Stevinson, Melih Barsbey, Tolga Birdal, and Pedro A. M. Mediano. Correlations in the data lead to semantically rich feature geometry under superposition. In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=7akSRQS5Xh.

Philip Quirke and Fazl Barez. Understanding addition in transformers. In The Twelfth International Conference on Learning Representations, 2024.

Philip Quirke, Clement Neo, and Fazl Barez. Understanding addition and subtraction in transformers. arXiv preprint arXiv:2402.02619, 2025.

Shauli Ravfogel, Michael Twiton, Yoav Goldberg, and Ryan Cotterell. Linear adversarial concept erasure, 2022.

Shauli Ravfogel, Yoav Goldberg, and Ryan Cotterell. Log-linear guardedness and its implications. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023, pp. 9413–9431. Association for Computational Linguistics, 2023a. doi: 10.18653/V1/2023.ACL-LONG.523. URL https://doi.org/10.18653/v1/2023.acl-long.523.

Shauli Ravfogel, Francisco Vargas, Yoav Goldberg, and Ryan Cotterell. Kernelized concept erasure, 2023b.

Juan Diego Rodriguez, Aaron Mueller, and Kanishka Misra. Characterizing the role of similarity in the property inferences of language models. CoRR, abs/2410.22590, 2024. doi: 10.48550/ARXIV.2410.22590. URL https://doi.org/10.48550/arXiv.2410.22590.

Paul K. Rubenstein, Sebastian Weichwald, Stephan Bongers, Joris M. Mooij, Dominik Janzing, Moritz Grosse-Wentrup, and Bernhard Schölkopf. Causal consistency of structural equation models. In Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI), 2017.

Andrew M Saxe, James L McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. Proceedings of the National Academy of Sciences, 116(23):11537–11546, 2019.

Or Shafran, Atticus Geiger, and Mor Geva. Decomposing mlp activations into interpretable features via semi-nonnegative matrix factorization, 2025. URL https://arxiv.org/abs/2506.10920.

Adam Shai, Loren Amdahl-Culleton, Casper L. Christensen, Henry R. Bigelow, Fernando E. Rosas, Alexander B. Boyd, Eric A. Alt, Kyle J. Ray, and Paul M. Riechers. Transformers learn factored representations, 2026. URL https://arxiv.org/abs/2602.02385.

Adam S Shai, Sarah E Marzen, Lucas Teixeira, Alexander G Oldenziel, and Paul M Riechers. Transformers represent belief state geometry in their residual stream. Advances in Neural Information Processing Systems, 37:75012–75034, 2024.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeffrey Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Isaac Bloom, et al. Open problems in mechanistic interpretability. Transactions on Machine Learning Research, 2025. URL https://openreview.net/forum?id=91H76m9Z94.

Noam Shazeer. Glu variants improve transformer. ArXiv, abs/2002.05202, 2020. URL https://api.semanticscholar.org/CorpusID:211096588.

Jiajun Song and Yiqiao Zhong. Uncovering hidden geometry in transformers via disentangling position and context. arXiv preprint arXiv:2310.04861, 2023.

Peter Spirtes, Clark Glymour, and Richard Scheines. Causation, Prediction, and Search. MIT Press, 2000.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023, pp. 7035–7052. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.435. URL https://doi.org/10.18653/v1/2023.emnlp-main.435.

Nishant Subramani, Nivedita Suresh, and Matthew E. Peters. Extracting latent steering vectors from pretrained language models. arXiv:2205.05124, 2022. URL https://arxiv.org/abs/2205.05124.

Denis Sutter, Julian Minder, Thomas Hofmann, and Tiago Pimentel. The non-linear representation dilemma: Is causal abstraction enough for mechanistic interpretability?, 2025. URL https://arxiv.org/abs/2507.08802.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=AwyxtyMwaG.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 12388–12401. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/92650b2e92217715fe312e6fa7b90d82-Paper.pdf.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In The Eleventh International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Zhengxuan Wu, Atticus Geiger, Thomas Icard, Christopher Potts, and Noah D. Goodman. Interpretability at scale: identifying causal mechanisms in alpaca. In Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Julian Yocum, Cameron Allen, Bruno Olshausen, and Stuart Russell. Neural manifold geometry encodes feature fields. In NeurIPS Workshop on Symmetry and Geometry in Neural Representations (NeurReps), 2025. URL https://neural-mechanics.baulab.info/papers/yocum-2025-feature-fields.pdf. Proceedings Track.

Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. Interpreting and improving large language models in arithmetic calculation, 2024. URL https://arxiv.org/abs/2409.01659.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks, 2023. URL https://arxiv.org/abs/2306.17844.

Tianyi Zhou, Deqing Fu, Vatsal Sharan, and Robin Jia. Pre-trained large language models use fourier features to compute addition, 2024. URL https://arxiv.org/abs/2406.03445.

Tianyi Zhou, Deqing Fu, Mahdi Soltanolkotabi, Robin Jia, and Vatsal Sharan. Fone: Precise single-token number embeddings via fourier features. arXiv preprint arXiv:2502.09741, 2025.

Fangwei Zhu, Damai Dai, and Zhifang Sui. Language models encode the value of numbers linearly. In Proceedings of the 31st International Conference on Computational Linguistics, pp. 693–709, 2025.

A Related Work

Representation geometry. Neural network representations have been shown to encode abstract concepts along intricate, nonlinearly curved geometries across architectures and modalities (Fel et al., 2025; Csordás et al., 2024; Lubana et al., 2025; Costa et al., 2025; Shafran et al., 2025; Saxe et al., 2019; Maheswaranathan et al., 2019; Park et al., 2025b;a; Shai et al., 2026; Pearce et al., 2025; Gurnee et al., 2025; Yocum et al., 2025; Brenner et al., 2026; Piotrowski et al., 2024; Shai et al., 2024; Kantamneni & Tegmark, 2025; Song & Zhong, 2023; Doshi et al., 2026). As recent work has started to concretize in models of scale (Karkada et al., 2026; Prieto et al., 2026; Korchinski et al., 2025), similar to prior work in toy scenarios (Saxe et al., 2019; Shai et al., 2024; Arora et al., 2018; Park et al., 2025b), these geometric structures are an artifact of data statistics—a model optimally encoding the data distribution ought to organize a concept along specific geometries, as claimed by these works and corroborated empirically by literature. However, despite the richness of this literature, there has been minimal work justifying the causal role this representation geometry plays in behavior, i.e, over the output space of the model. Our work takes a step towards filling this gap by developing a precise account of a task known to show geometrically structured representations in toy scenarios, i.e., modular addition (Nanda et al., 2023; Morwani et al., 2024), by analyzing it as arithmetic over concepts known to show circular geometries in model representations (e.g., weekdays and months) (Engels et al., 2025; Karkada et al., 2026; Park et al., 2025a; Nishi et al., 2024).

Causal analysis of neural networks. We build on prior interpretability research grounded in the theory of causality (Hume, 1748; Pearl, 1999; Spirtes et al., 2000). In particular, causal mediation (Pearl, 2001; Vig et al., 2020; Mueller et al., 2026) has been used to characterize the flow of information in neural networks and causal abstraction (Rubenstein et al., 2017; Beckers & Halpern, 2019; Geiger et al., 2021; 2025c;b) has been used to uncover algorithmic processes. Empirically, we used a variety of different intervention have been applied to neural network hidden representations, including ablations (Li et al., 2017; Cammarata et al., 2020; Elazar et al., 2020; Ravfogel et al., 2022; 2023a;b; Belrose et al., 2023; Geva et al., 2023; Meng et al., 2022; 2023), steering (Giulianelli et al., 2018; Bau et al., 2018a;b; Besserve et al., 2020; Subramani et al., 2022; Antverg & Belinkov, 2022; Marks & Tegmark, 2023), and interchange interventions (Vig et al., 2020; Geiger et al., 2020; Finlayson et al., 2021; Davies et al., 2023; Stolfo et al., 2023; Guerner et al., 2023; Wang et al., 2023; Todd et al., 2024; Arora et al., 2024; Huang et al., 2024; Feng & Steinhardt, 2024; Mueller et al., 2025; Prakash et al., 2025; Gur-Arieh et al., 2025; Grant et al., 2025; Rodriguez et al., 2024; Sutter et al., 2025).

Interpretability on mathematical reasoning in LMs. We build on prior work aimed at understanding how LMs perform arithmetic. Hanna et al. (2023) and Wu et al. (2023) study greater-than operations, while Nanda et al. (2023) and Zhong et al. (2023) study modular arithmetic in small transformers. Several works isolate specific components responsible for arithmetic (Nikankin et al., 2025; Stolfo et al., 2023; Zhang et al., 2024; Quirke & Barez, 2024; Quirke et al., 2025; Bai et al., 2025). Of these works, several have identified base-10 representations of numbers (Levy & Geva, 2025; Kantamneni & Tegmark, 2025; Zhou et al., 2024; 2025), although we provide a new perspective by unifying these representations with non-numerical concepts. More critically, we identify the computational basis, i.e., the concrete neurons that underlie these geometries and the corresponding computation performed over it in a model of scale. While prior work has partially shown such results, to our knowledge, these studies have been limited to toy scenarios that primarily show a correlation between geometry and the precise algorithm a model implements to perform a task (Gopalani et al., 2024; Gopalani & Hu, 2025; Edelman et al., 2023; Morwani et al., 2024; Kunin et al., 2025; Marchetti et al., 2026) or focused primarily on representation geometry, i.e., does not demonstrate how these geometries are implemented or used in computation (Zhou et al., 2025; 2024; Kantamneni & Tegmark, 2025).

B Limitations

Models. In this work, we conduct an in-depth analysis of how Llama-3.1-8B reasons about cyclic concepts. While prior work suggests that some of the phenomena we observe—such as

Fourier features (Zhou et al., 2024; Kantamneni & Tegmark, 2025) or circular representations (Engels et al., 2025)—also appear in other models, we cannot assume that the specific mechanisms identified in this work generalize beyond Llama-3.1-8B.

Data. Our analysis is limited to three types of cyclic concepts (hours, days, and months), while many other cyclic structures exist and remain unexplored. Moreover, we consider only a specific class of reasoning tasks—specifically, addition-based problems. Other forms of cyclic reasoning, such as backward offsets (e.g., "what month is 4 months before July?") or distance queries (e.g., "If it's July and I have an appointment in September, how many months away is that?"), are outside the scope of this study.

Analysis. We focus primarily on how Llama-3.1-8B performs the addition step within cyclic domains. Although we observe that intermediate sums are mapped to the appropriate output concept, we do not analyze *how* the model computes this mapping, and leave this stage for future work.

C Task Setup

C.1 Task Definitions

We define four tasks: three natural modular arithmetic tasks (`weekdays`, `months`, and `hours`), and one control `addition` task. For each task, we evaluate on every combination of input concept (e.g., *Monday*, *January*, *00:00*) and offset (e.g., *one*, *two*, etc.).

Table 1: Task templates and possible values for **input concept** and **offset**. All **output** variables are drawn from the same set as **input concept** variables. We assume that Monday=1 for `weekdays` and January=1 for `months`, and take the integer version of the hour for `hours`.

Task	Template	`concept`	`offset`	n	Acc.
`months`	*Q: What month is {offset} months after {concept}?\nA:*	January ...December	one... twenty-four	288	65.3%
`weekdays`	*Q: What day is {offset} days after {concept}?\nA:*	Monday ...Sunday	one... fourteen	98	71.4%
`hours`	*Q: In 24-hour time, it is now {concept}:00. What time will it be in {offset} hours?\n A: In 24-hour time, it will be*	00, 01, ... 22, 23	one... forty-eight	1152	70.8%
`addition`	*a+b=*	a $\in$ [1, ..., 100]	b $\in$ [1, ..., 100]	10k	97.2%

C.2 Model Performance

Performance for small number ranges is acceptable, but begins to break down as offset increases. We first evaluate Llama-3.1-8B performance on each task broken down by whether the model has to perform a modulo operation, assuming that Monday is the first weekday and January is the first month. Table 2 shows that most failures come from prompts for which the model has to take a modulo, or "wrap" back around the concept circle (e.g., *What month is four months after October*). This suggests that, even if concepts are structured as a circle, the first concept is still privileged in some sense.

Table 2: Llama-3.1-8B accuracy for all tasks, overall and broken down in two manners. We assume that Monday is the first weekday. **By Number Range**: Performance worsens as offset increases. **By Pre-Modulo Sum**: the model has almost perfect accuracy for examples where the answer does not "wrap" around (e.g., *three months after June*, $3 + 6 \leq 12$), but performs poorly for examples where the answer does "wrap" (e.g., *three months after November*, $3 + 11 > 12$).

Task	**By Number Range**		**By Pre-Modulo Sum**		
	$1 \to p$	$p \to 2p$	All	sum $\leq m$	sum $> m$
`months`	81.9%	48.6%	65.3%	100%	55.0%
`weekdays`	91.8%	51.0%	71.4%	95.2%	64.9%
`hours`	97.0%	44.6%	70.8%	100%	60.6%
`addition`	—	—	97.2%	—	—

For a more fine-grained understanding of Table 2, we aggregate prompts by their pre-modulo sum and plot in Figure 11. For example, if January=1, then the pre-modulo value of *January + fourteen* is 15. We observe a clear pattern: as the pre-mod value increases, model accuracy decreases. Interestingly, the model's errors for higher pre-mod values are quite regular, with probability mass placed along diagonal offsets from the true answer. Table 3 summarize these heatmaps.

Discussion. If weekday, month, and hour concepts are represented as circles (Engels et al., 2025), one might expect that Llama-3.1-8B performs calculations over these circles by

Table 3: Llama-3.1-8B accuracy for cyclic tasks broken down by pre-modulo sum range, separately and and cumulatively. The model has difficulty "taking the modulo" of larger numbers. We assume that Monday is the first weekday.

Task	Separate			Cumulative		
	$[1, p]$	$[p, 2p]$	$[2p, 3p]$	$\leq p$	$\leq 2p$	$\leq 3p$
`months`	100.0%	68.1%	30.8%	100.0%	78.1%	65.3%
`weekdays`	95.2%	87.8%	25.0%	95.2%	90.0%	71.4%
`hours`	100.0%	81.1%	17.8%	100.0%	87.6%	70.8%

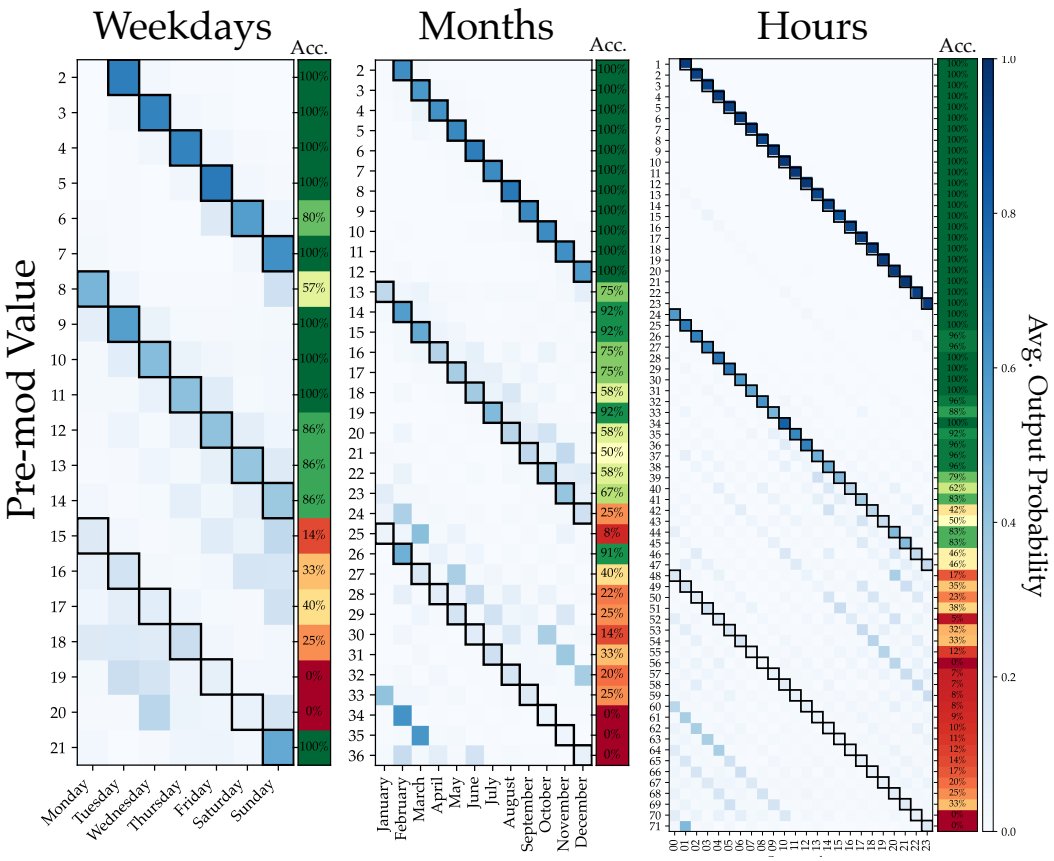


Figure 11: **Model performance for cyclic tasks.** Llama-3.1-8B output probabilities for all prompts in each dataset are displayed, aggregated by pre-modulo sum. Cells for correct answers are outlined in black, and accuracy for each pre-modulo value is displayed beside each row. When the pre-modulo sum is less than or equal to the cycle length for a given concept (e.g., *January* + *four*=5=*May*), model accuracy is perfect, but performance begins to waver once a modulo must be calculated (e.g., *January* + *fifteen*=16=*April* has 75% accuracy). As the pre-modulo sum increases, the model begins to make systematic errors (i.e., probability is placed on the wrong diagonal).

rotating along them. In such a world, there would be no difference between *four months after January* and *four months after November*, as the circular geometry would treat both calculations in the same way. However, the fact that we see such a drastic drop in performance for prompts that must "wrap around" to the canonical start of the sequence tells us that there *is* still a notion of "start" and "end" for these circular concepts.

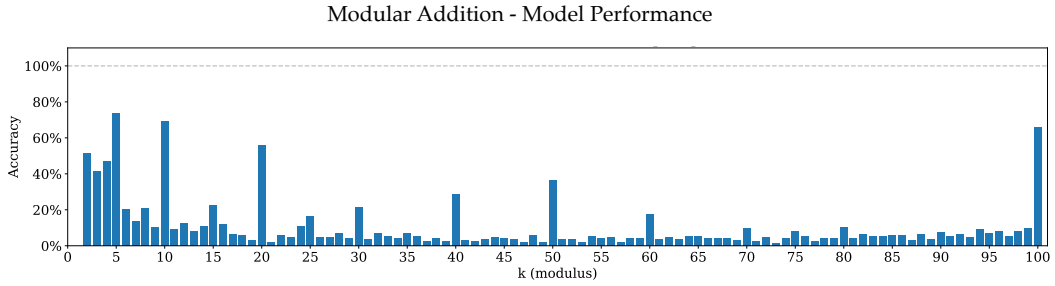


Figure 12: We test whether Llama-3.1-8B can solve prompts of the form `Q: What is ({a} + {b}) mod {k}?\nA:`, where $a, b \in [1, 200]$ and $k \in [2, 100]$. We measure performance on 1,000 **randomly sampled** prompts of this form. Performance is poor for most moduli.

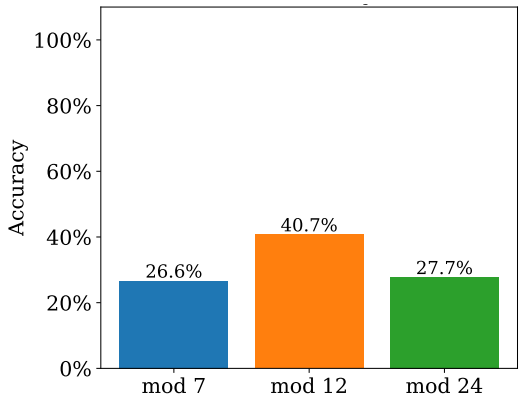


Figure 13: We test whether Llama-3.1-8B can solve prompts of the form `Q: What is ({a} + {b}) mod {k}?\nA:`, where $k \in \{7, 12, 24\}$. For each modulus, we evaluate performance on prompts where $a + b \leq 3k$, which is **equivalent to the range used in the cyclic tasks**.

C.3 Can Llama Perform Standard Modular Addition?

We evaluate whether Llama can perform standard modular addition. To this end, we use prompts of the form `Q: What is ({a} + {b}) mod {k}?\nA:`, where $a, b \in [1, 200]$ and $k \in [2, 100]$, sampling 1,000 prompts in total. As shown in Figure 12, performance is low across most moduli, including 7, 12, and 24.

To enable a more direct comparison with the cyclic tasks, for each modulus $k \in \{7, 12, 24\}$ we also evaluate performance over the same range used in those tasks, namely for prompts satisfying $a + b \leq 3k$. The results are shown in Figure 13. Comparing these results with the model's performance on the cyclic tasks (Table 2) shows that, although the cyclic tasks implicitly require modular addition, the model performs better on those tasks than on standard modular addition.

D Causal Abstraction and Causal Models

Input-output causal models. Define an input-output causal model $\mathcal{A}$ to be a directed acyclic graph where each node is a *variable* V with a *mechanism* $\mathcal{F}_V(\mathbf{p}) = v$ that determines the value v of V from the value $\mathbf{p}$ of its parents $\mathbf{P}$.[5] Input variables $\mathbf{X}$ lack parents and output variables $\mathbf{Y}$ lack children. Denote the input-output behavior of a causal model with $\mathcal{A}(\mathbf{x}) = \mathbf{y}$. We represent both algorithmic hypotheses about neural networks and neural networks themselves as causal models, with neural networks outputting logits.

[5]We use case to distinguish a value v from its variable V. We use bold to distinguish individual values and variables, i.e., v and V, from sets of values and variables, i.e., $\mathbf{p}$ and $\mathbf{P}$.

Interventions. An *intervention* $\mathbf{V} \leftarrow \mathbf{v}$ on $\mathcal{A}$ produces a new model $\mathcal{A}_{\mathbf{V}\leftarrow\mathbf{v}}$ where the mechanisms of $\mathbf{V}$ are fixed to constant functions outputing the values $\mathbf{v}$. Given a counterfactual input c, define an *interchange intervention* $\mathbf{V} \leftarrow \mathcal{A}(c)$ to fix the value of variable $\mathbf{V}$ to the value they would have taken for input c.

Neural network features. An activation vector $\mathbf{h}$ will often not have interpretable dimensions in a standard basis. To solve this problem, we can *featurize* the activation vector using an invertible transformation mapping into a feature space $\mathbb{F}$ where a feature F is a dimension of $\mathbb{F}$.[6] Principal component analysis, sparse autoencoders, and rotation-based distributed alignment search all produce linear features, where a feature value is computed by projecting a hidden vector onto a line. More complex representational schemes involve projecting hidden vectors onto circles or onions (Csordás et al., 2024), where features correspond to angle or magnitude rather than direction.

Causal abstraction. Let $\mathcal{D}$ be a dataset of input-pairs. $\mathcal{A}$ is a causal abstraction of $\mathcal{N}$ if the following holds for all $(o, c) \in \mathcal{D}$:

$$\mathcal{N}_{\mathbf{F}\leftarrow\mathcal{N}(c)}(o) = \mathcal{A}_{\mathbf{V}\leftarrow\mathcal{M}(c)}(o) \tag{7}$$

The *interchange intervention accuracy* is proportion of pairs in $\mathcal{D}$ for which Eq. 8 holds.

Distributed Alignment Search. The method *distributed alignment search* localizes an abstract causal variable V to a low-rank subspace of an activation vector. Concretely, it learns a low-rank orthogonal matrix $\mathbf{R}^\theta \in \mathbb{R}^{d\times k}$ such that interchange interventions on the subspace defined by the matrix produce the outputs of the algorithm under interchange intervention. An interchange intervention on $\mathbf{R}^\theta$ will run the LM on o and fix the k-dimensional subspace to the value it takes on for the input c. The change of basis is optimized (with $\mathcal{N}$ frozen) to maximize cross entropy with the label from $\mathcal{A}$ under an interchange intervention:

$$\mathsf{CrossEntropy}\big(\mathcal{N}_{\mathbf{R}^\theta\leftarrow\mathcal{N}(c)}(o), \mathcal{A}_{\mathbf{V}\leftarrow\mathcal{M}(c)}(o)\big) \tag{8}$$

D.1 Causal Model of the Addition Module

Causal model. Let $\mathcal{M}$ be an input-output causal model with input variables $\mathbf{X} = \{N, C^{\text{in}}\}$ and output variables $\mathbf{Y} = \{C^{\text{out}}\}$. Here, N is the offset (shift amount), and C^{in} is the input concept (day, month, hour-word, or number-token). The intermediate variables are: B (base), K (concept number), S (sum), and M (output number).

Lookup tables. Fix two lookup tables: `con_to_num` maps concepts to number-space, and `num_to_con` is its inverse on each concept-domain. For days and months it is 1-indexed (e.g. `Mon` $\mapsto 1, \ldots,$ `Sun` $\mapsto 7$ and `Jan` $\mapsto 1, \ldots,$ `Dec` $\mapsto 12$); for hour-words it is zero-indexed (e.g. `00:00` $\mapsto 0, \ldots,$ `23:00` $\mapsto 23$); and for number-tokens it is the identity (e.g. `10` $\mapsto 10$). The inverse table maps back accordingly.

Mechanisms.

$$\mathcal{F}_B(c) = \begin{cases} 7 & c \in \{\texttt{Mon}, \ldots, \texttt{Sun}\} \\ 12 & c \in \{\texttt{Jan}, \ldots, \texttt{Dec}\} \\ 24 & c \in \{\texttt{00:00}, \ldots, \texttt{23:00}\} \\ \infty & \text{otherwise (addition)} \end{cases} \qquad \mathcal{F}_K(c) = \mathsf{con\_to\_num}(c),$$

$$\mathcal{F}_S(n, k) = n + k, \qquad \mathcal{F}_M(s) = s,$$

$$\mathcal{F}_{C^{\text{out}}}(m, b) = \mathsf{num\_to\_con}(m \bmod b), \quad \text{where } s \bmod \infty \equiv s.$$

[6] See Geiger et al. (2025a) for a technical definition.

E Coarse Localization with Residual Stream Patching

To determine which token positions are important, we begin with simple residual stream interchange interventions (Geiger et al., 2021). Using the counterfactual dataset from Section F.1, we patch the entire residual stream from a source prompt $\mathbf{h}_{src}$ at layer l into a base prompt and measure how the model's output changes. Figure 14a shows results for the `output` variable in the `weekdays` task: the model's eventual output appears at the last token position at layer 18.

But is this answer actually *computed* at the last token position, or is it copied? The answer appears to be that computation occurs at the last token, because we can intervene on the offset at this position without changing the input concept. Figure 14b shows that the `number` variable can also be localized at the last token position, by patching entire hidden states at layer 15: 81% of the time, patching from e.g. *two days after Monday* → *five days after Thursday* at this position causes the model to output *Saturday* (*Thursday* + 2).

The `input` variable is not as cleanly separable at the last token position (Figure 14c), but this makes sense: if the `number` variable is always copied to the last token position first, then there is no point in the residual stream where `input` can be disentangled via residual stream patching. Patching at the input token position loses its effectiveness after layer 17, suggesting that a head in layer 18 copies the input day to the last token position.

Taken together, these results suggest that for all three tasks, answers are calculated at the last token position. The model first copies `number` information to the last token at layer 15, followed by `input` day/month/hour information, yielding a result around layer 18.

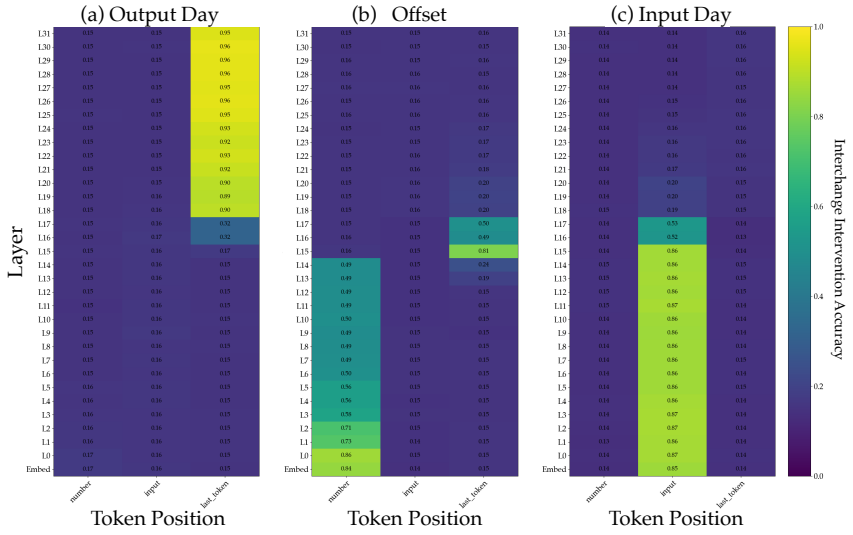


Figure 14: Residual stream patching results for the `weekdays` task for each causal variable. Patching is done for $n = 4096$ counterfactual pairs (Section F.1). (a) The output concept appears at the last token position after approximately layer 18. (b) The offset is moved to the last token position at layer 15. (c) Residual stream patching can only isolate the input concept variable at its own token position. Patching has no effect after layer 15-17, which suggests that the input day is copied to the last token position in layer 18. The template for this task is *Q: What day is {number} days after {input}?\nA:*

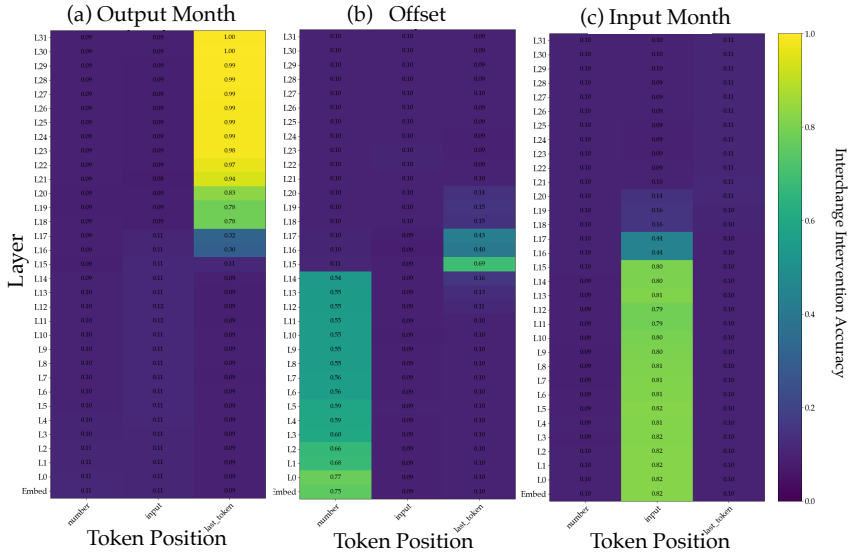


Figure 15: Residual stream patching results for the `months` task for each causal variable. Patching is done for $n = 4096$ counterfactual pairs (Section F.1). See Figure 14 for interpretation of results. The template for this task is *Q: What month is {number} months after {input}?\nA:*

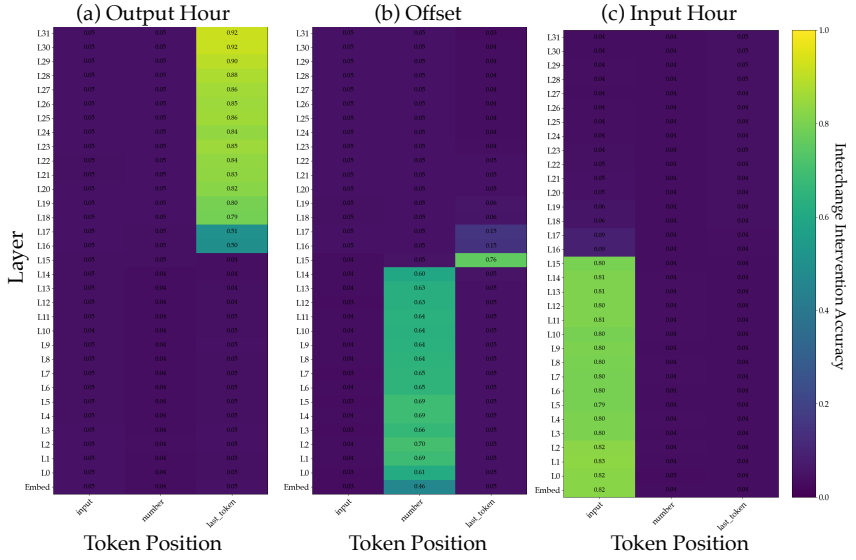


Figure 16: Residual stream patching results for the `hours` task for each causal variable. Patching is done for $n = 4096$ counterfactual pairs (Section F.1). See Figure 14 for interpretation of results. The template for this task is *Q: In 24-hour time, it is now 10:00. What time will it be in twenty-four hours?\nA: In 24-hour time, it will be*

F Distributed Alignment Search

F.1 Training Details

To train distributed alignment search (DAS) on the `months`, `weekdays`, `hours`, and `addition` tasks, we create datasets of 4096 randomly-sampled pairs of prompts per task, sampled from prompts that Llama-3.1-8B answers correctly (Table 1). Setting aside n_{test} random pairs for testing, we train on n_{train} counterfactual pairs, with $n_{\text{test}} = 512$, $n_{\text{train}} = 3584$.

With `weekdays` as a representative task, we run a hyperparameter sweep with $k = 8$ over epochs $\in \{4, 8\}$, learning rate $\in \{0.0001, 0.001, 0.01\}$, and batch size $\in \{16, 32, 64\}$ for the input concept variable at layer 18 after attention and before the MLP. This is a position we know to be important from initial exploratory runs. From this sweep, for `weekdays`, we choose epochs=8, lr=0.0001, and bsz=16. We then run DAS for every task with those particular hyperparameters.

To stabilize optimization, subspace basis vectors are initialized within the space spanned by the top principal components explaining $\geq 90\%$ of the variance in the training set.

F.2 Intervention Details

The simplest causal model we can construct for this task consists of the input concept (e.g., *January*), offset (e.g., *three*), and output concept (e.g., *three months after January is **April***). Our goal is to isolate subspaces of the last token residual stream that "contain" each of these three causal variables:

- Patching the input concept should change the input month: e.g. *three months after **January**→one month after **July*** should yield *one month after **January**=February*.
- Patching the offset should change the input offset: e.g. ***three** months after January→**one** month after July* should yield ***three** months after July=October*.
- Patching the output concept should change the entire output: e.g. *three months after January=**April**→one month after July=**August*** should yield *one month after January=**April***.

Focusing on the last token position, we train DAS for each causal variable at every layer $0 \leq l \leq 31$ and subspace dimension $1 \leq k \leq 8$. If performance does not break 90% at any point for a particular task, we add eight more dimensions, expanding to $1 \leq k \leq 16$ for the `hours` and `addition` tasks. Results are plotted in Figure 17. Immediately before the layer 18 MLP (dotted black line), input variables are cleanly separable in low-dimensional subspaces; immediately after, the output concept becomes patchable. This implies that across all tasks, the layer 18 MLP implements a major piece of computation.

F.3 Best DAS Subspaces

Pre-MLP DAS subspaces are trained on the residual stream immediately after attention has been added, but before LayerNorm. Post-MLP DAS subspaces are trained on the residual stream at the output of the layer, immediately after the MLP output has been added.

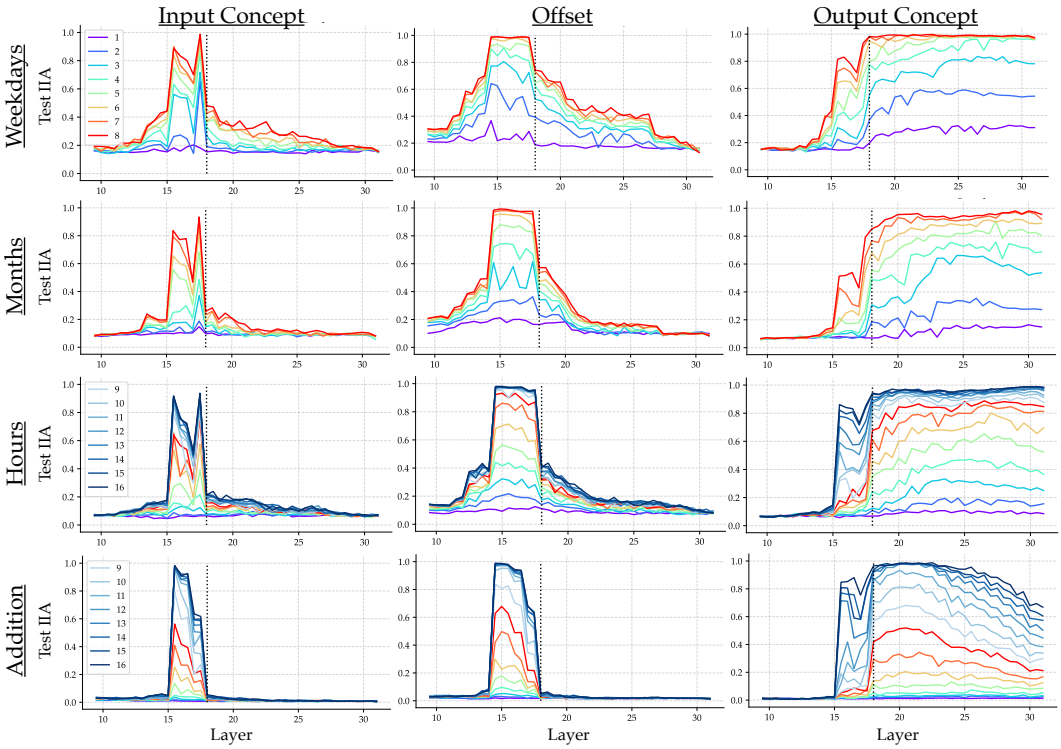


Figure 17: DAS results for every dimension across layers, alternating between sublayer (post attention) and layer output (post MLP). We include $8 < k \leq 16$ for `hours` and `addition` in blue, as $k \leq 8$ does not achieve maximum IIA.

Table 4: Best DAS runs for every task $\times$ causal variable at layer 18. All subspaces are matrices $\mathbf{R} \in \mathbb{R}^{d \times k}$. We use these subspaces for all experiments throughout the paper.

Task	Variable	Pre/Post-MLP	Best Dim. k	Test IIA
Months	input concept	Pre-MLP	$k = 8$	93.4%
	offset	Pre-MLP	$k = 8$	97.5%
	output concept	Post-MLP	$k = 8$	85.2%
Weekdays	input concept	Pre-MLP	$k = 8$	98.7%
	offset	Pre-MLP	$k = 8$	98.5%
	output concept	Post-MLP	$k = 8$	98.3%
Hours	input concept	Pre-MLP	$k = 16$	94.0%
	offset	Pre-MLP	$k = 14$	95.6%
	output concept	Post-MLP	$k = 16$	94.3%
Addition	input concept	Pre-MLP	$k = 16$	61.7%
	offset	Pre-MLP	$k = 15$	63.8%
	output concept	Post-MLP	$k = 16$	96.6%

F.4 Overlap Between DAS Subspaces

To get a measure of how much the best subspaces for each task/causal variable overlap, we use principal angles. Let $\mathbf{A} \in \mathbb{R}^{d \times m}$ be a matrix with orthonormal columns spanning an m-dimensional subspace, and $\mathbf{B} \in \mathbb{R}^{d \times n}$ be a matrix with orthonormal columns spanning an n-dimensional subspace. We can calculate the principal angles between these subspaces

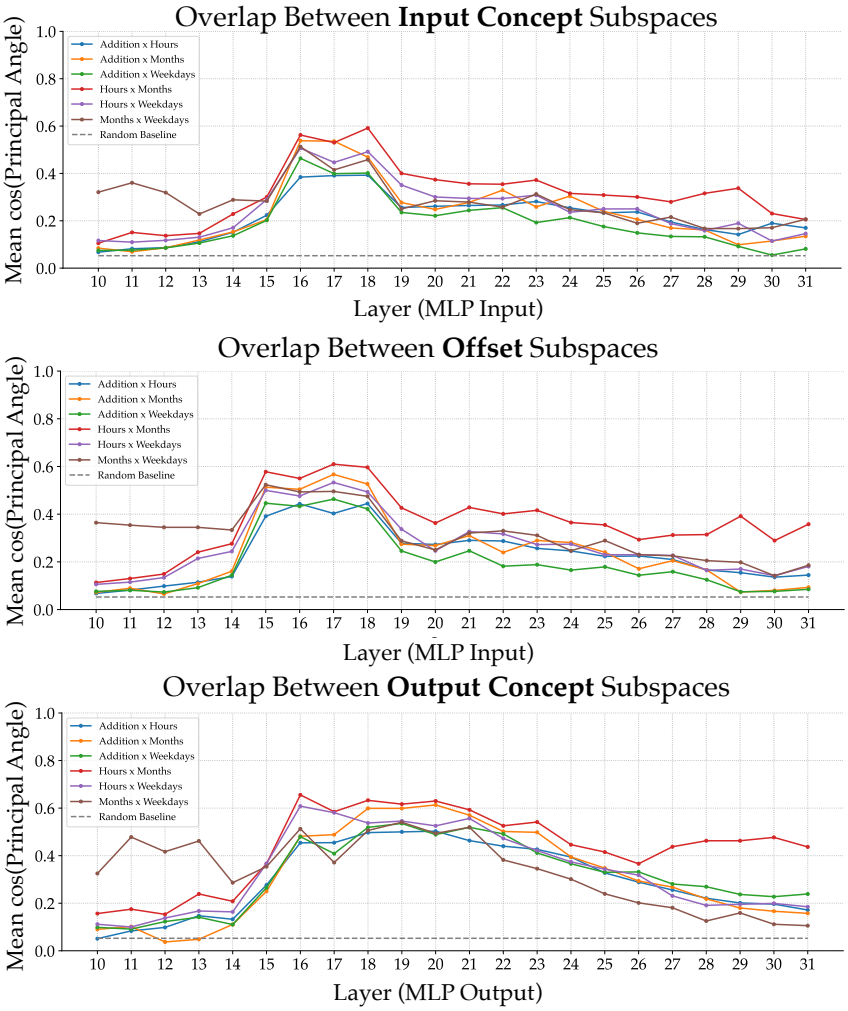


Figure 18: Average cosines of principal angles between subspaces for each causal variable in different tasks. Right before the important computation at layer 18, input concept and offset overlap significantly above change. Subspaces for output concept also increase in overlap in these layers.

by taking the singular values $\sigma_1 ... \sigma_{\min(m,n)}$ of $\mathbf{A}^T\mathbf{B}$. The first singular value σ_1 corresponds to the cosine of the smallest angle between the closest vectors in each subspace; σ_2 is the next smallest angle orthogonal to the first, and so on.

We take the average of all of these singular values and report them in Figure 18. If this metric is 0, the subspaces are orthogonal; if it is 1, they are identical (or the smaller one is fully contained in the larger one). We observe that similarity for output concept subspaces increases around layers 16-20, much above a baseline between 1000 pairs of random subspaces with $m, n = 16$. Subspaces also overlap for input concept and offset variables.

G Weekday Alignment with Numbers

Although we find strong evidence that addition is used in computation for the `weekdays` task, the exact alignment between weekdays and numbers is unclear. We hypothesize that this is due to a lack of a "canonical" enumeration of weekdays, which causes noise/misalignment in the embedding and unembedding spaces. Figure 19 shows cosine similarities between weekday tokens and number tokens up to 30: we observe that alignment in the embedding

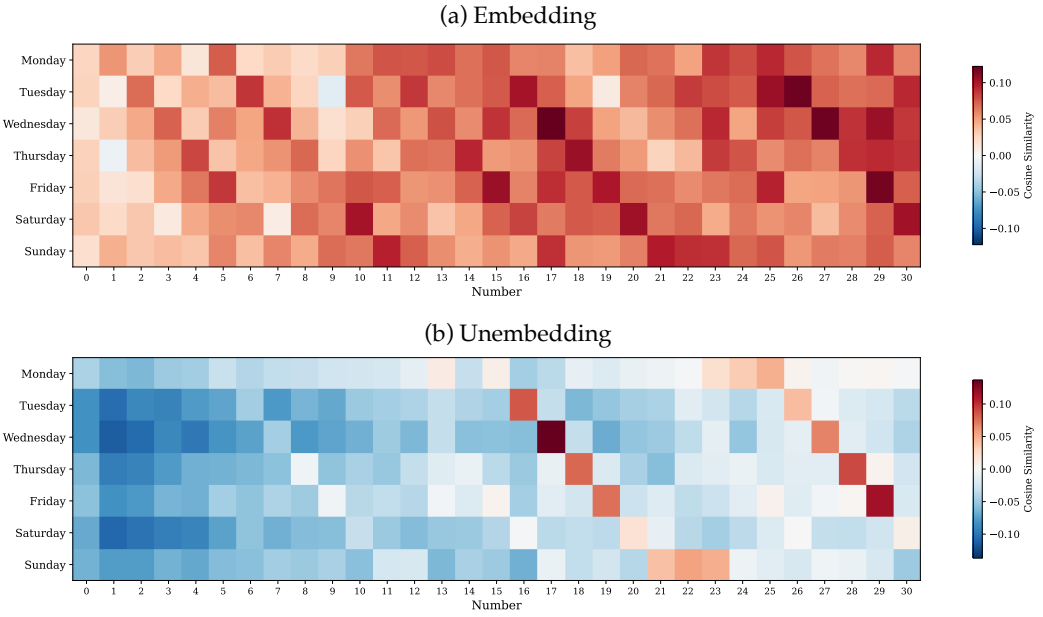


Figure 19: Alignment between weekday/number tokens for Llama-3.1-8B with (a) token embeddings and (b) token unembeddings.

space is very noisy. Although we do observe diagonal patterns in weekday alignment with numbers, there are odd offsets: for example, Wednesday is most aligned with the number 17 in the unembedding space. In our experiments, we account for this misalignment in a few ways.

When patching from `weekdays` into `addition` (Figure 20c, Figure 24), we find that probability heatmaps have a consistent diagonal pattern, but shifted with an offset of +4 (i.e., when the `addition` sum is 6, the model outputs 10, and so on). This means that when we patch the other way, from `addition` to `weekdays`, we must shift our expectations: instead of assuming that 1 will map to Monday, we assume that the model computes sums for the weekdays task with an offset of four, meaning that 1+4=5 will actually map to Monday. Although this assumption is imperfect, it appears to predict model behavior for this experiment reasonably well (Figure 26).

This +4 shift also appears consistent with the fact that the unembedding vector for, e.g., Tuesday is most similar to the number token 16 (Figure 19). If Tuesday was the number 2 under our initial incorrect assumptions, then a shift of +4 to match the model would give us 6. Due to the fact that Llama-3.1-8B mainly uses periodicities $T \in \{2, 5, 10\}$ for the `weekdays` task (Tab. 5, Figure 43), the model would not be able to distinguish between 6, 16, 26, etc.

For Fourier steering, we show results when defining Saturday as 0, which we empirically find to have the most consistent results. We do not have a good explanation, other than the fact that this would possibly imply that Sunday is 1, which appears to be a plausible enumeration. If the model is ignoring the tens place in these calculations, then this would also be consistent with Saturday being similar to 20 and 30 in the unembedding space (Figure 19).

We acknowledge that this is messy, but reiterate that the addition neurons we find *are* causally important for the `weekdays` task (Tab. 6)—we are quite sure that these neurons (which perform addition cleanly for all other tasks) are being used to compute, e.g., *three days after Friday*, but do not claim to understand Llama-3.1-8B's internal logic for how it enumerates weekdays.

H Cross-Task Patching

When we "patch within the union of subspaces," this simply means that we concatenate the columns spanning the source and target subspaces to obtain a new, larger subspace. For example, for a source subspace $\mathbf{A} \in \mathbb{R}^{d_{\text{model}} \times m}$ and a target subspace $\mathbf{B} \in \mathbb{R}^{d_{\text{model}} \times n}$, we build a new matrix $\mathbf{C} = [\mathbf{A}\ \mathbf{B}] \in \mathbb{R}^{d_{\text{model}} \times (m+n)}$. We then orthogonalize the columns of $\mathbf{C}$ and use it to patch from source to target prompts following Eq. 2.

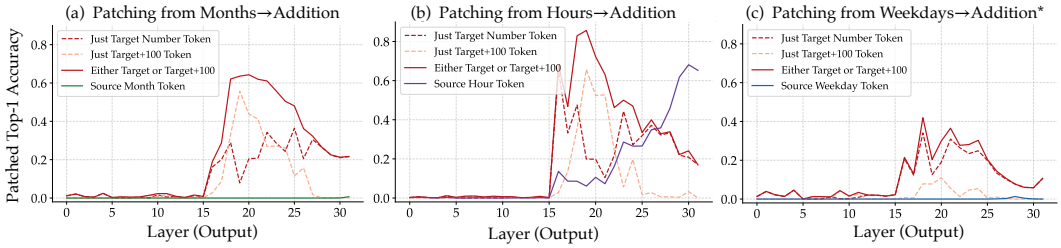


Figure 20: Accuracy when patching **from cyclic tasks into `addition`**: this intervention "exposes" a sum computed in the forward pass of a cyclic prompt (e.g., *three months after November=3+11=14*). (a) Patching from `months`→`addition`. More than 60% of the time, this intervention exposes the pre-modulo sum, but sometimes the highest probability is that sum + 100. See Figure 22 for heatmaps. (b) Patching from `hours`→`addition`. In intermediate layers, this exposes the pre-modulo sum, but patching too late brings over the modded hour token. See Figure 23 for heatmaps. (c) Patching from `weekdays`→`addition`. Weekdays are misaligned—these accuracies are based on an offset of +4, where Monday=5.

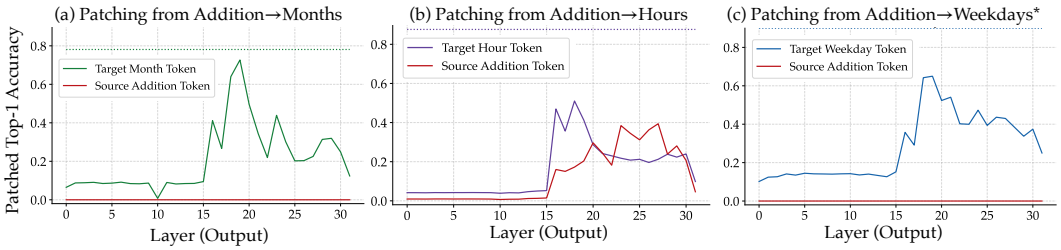


Figure 21: *[Duplicate of Figure 4 for easy cross-reference.]* Accuracy when patching **from `addition` into cyclic tasks**: this intervention forces the model to "take the modulo" of a sum calculated in an addition setting, mapping that value to a particular output concept (e.g., *5+9=14→February*). We patch within the union of subspaces for both tasks. Because the clean model's accuracy begins to break down for larger numbers (Figure 11), we limit `addition` prompts to those with sum $\leq 2p$. (a) See heatmaps in Figure 25. (b) See heatmaps in Figure 27. (c) See heatmaps in Figure 26. Again, as weekdays are misaligned, we use an offset of +4.

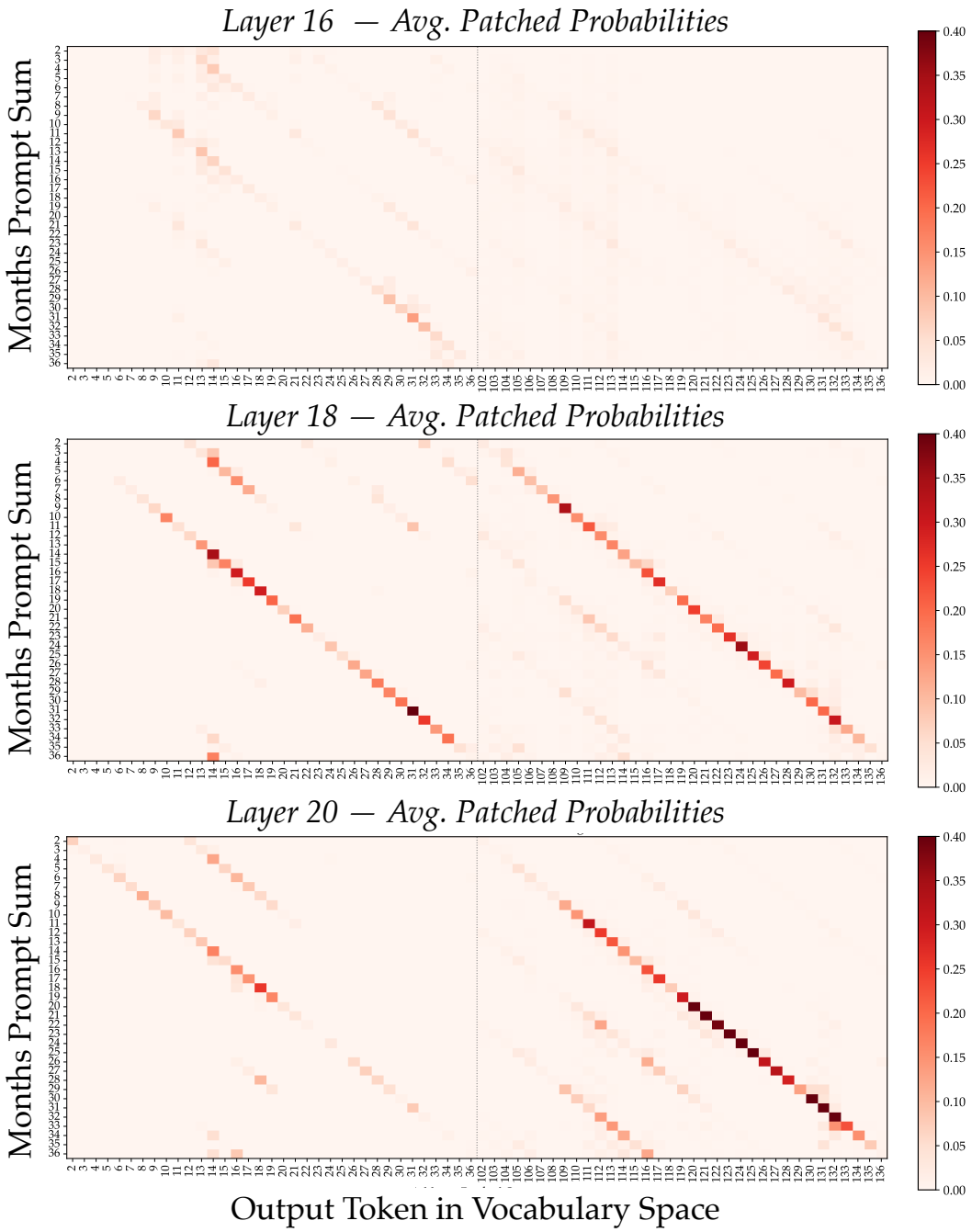


Figure 22: Patching from `months`→`addition` at layers 16, 18, and 20. Note that patching is most consistent at layer 18; this plot is the same as Figure 5. See Figure 20(a) for performance across all layers.

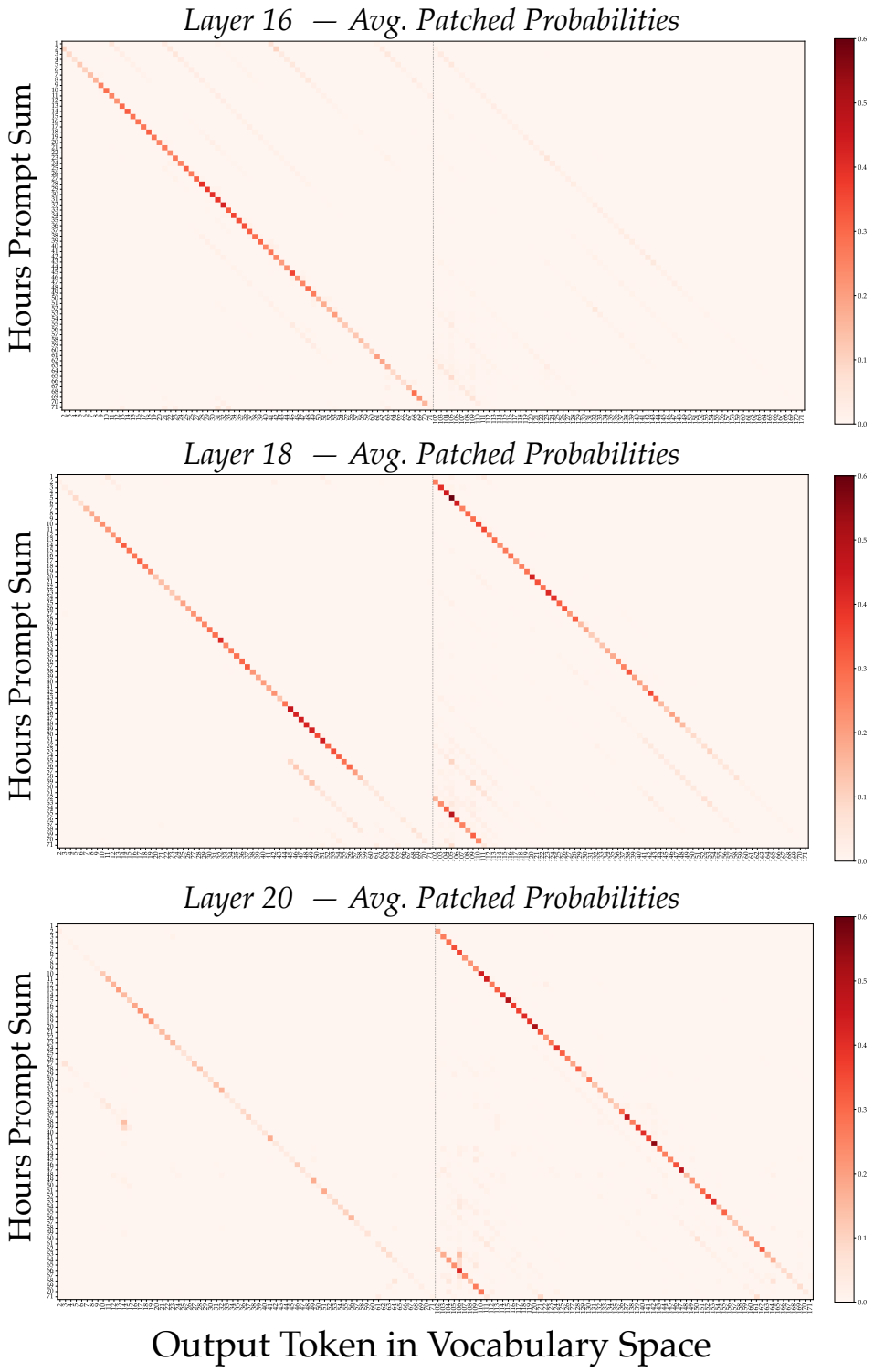


Figure 23: Patching from `hours`→`addition` at layers 16, 18, and 20. Surprisingly, patching is also quite effective at layer 16 for this task, implying that the input representations for `hours` transfer well to `addition`; this may be because `hours` uses literal number tokens. See Figure 20(b) for performance across all layers.

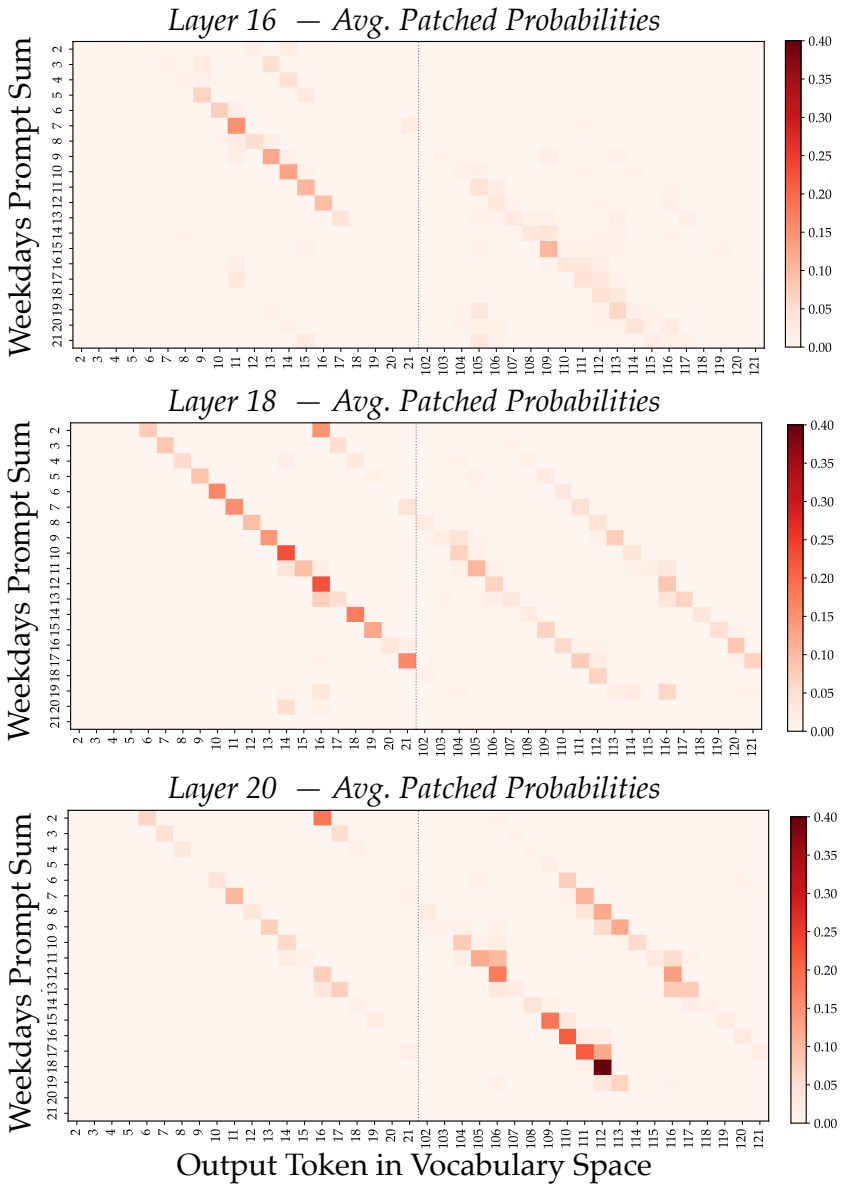


Figure 24: Patching from `weekdays`→`addition` at layers 16, 18, and 20. Note that patching is most consistent at layer 18, and that we observe a strange +4 offset for `weekdays`: see App. G for discussion. See Figure 20(b) for performance across all layers.

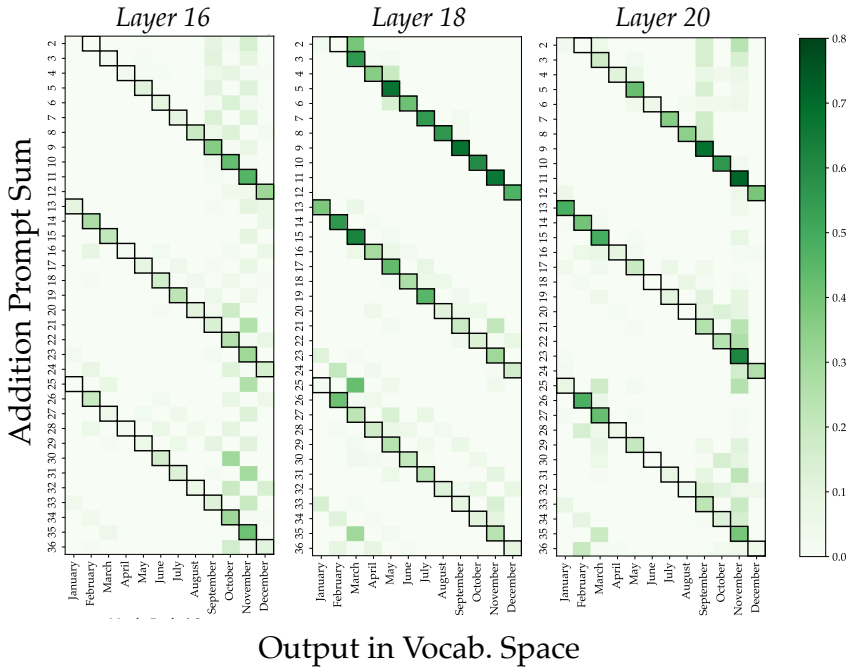


Figure 25: Patching from addition→months at layers 16, 18, and 20. Note that patching works best at layer 18, and that performance begins to break down as the sum increases, closely matching errors in a clean forward pass in Figure 11. See Figure 21(a) for performance across all layers.

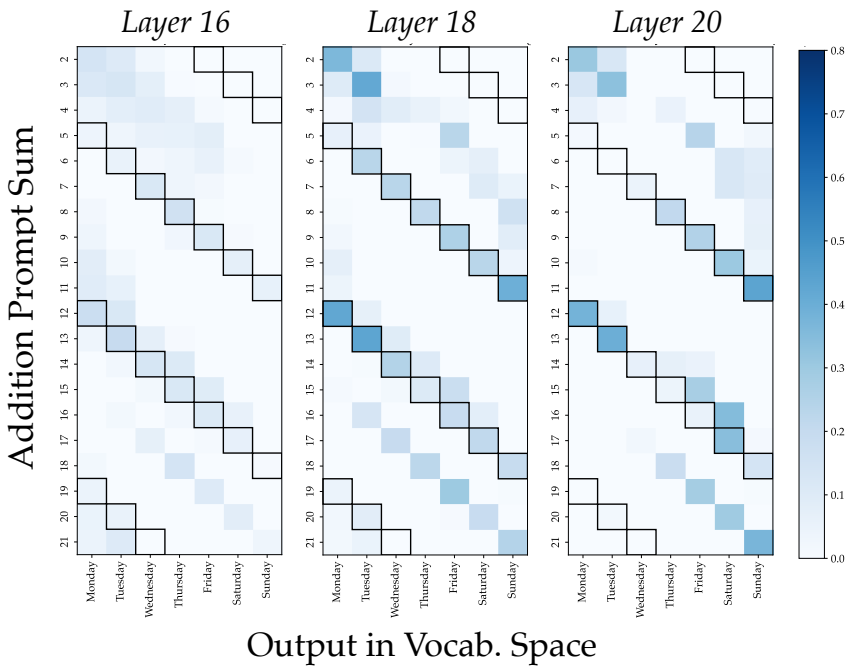


Figure 26: Patching from addition→weekdays at layers 16, 18, and 20. Note that patching works best at layer 18, and that performance begins to break down as the sum increases, closely matching errors in a clean forward pass in Figure 11. See Figure 21(c) for performance across all layers. We define Monday=5; see App. G for discussion.

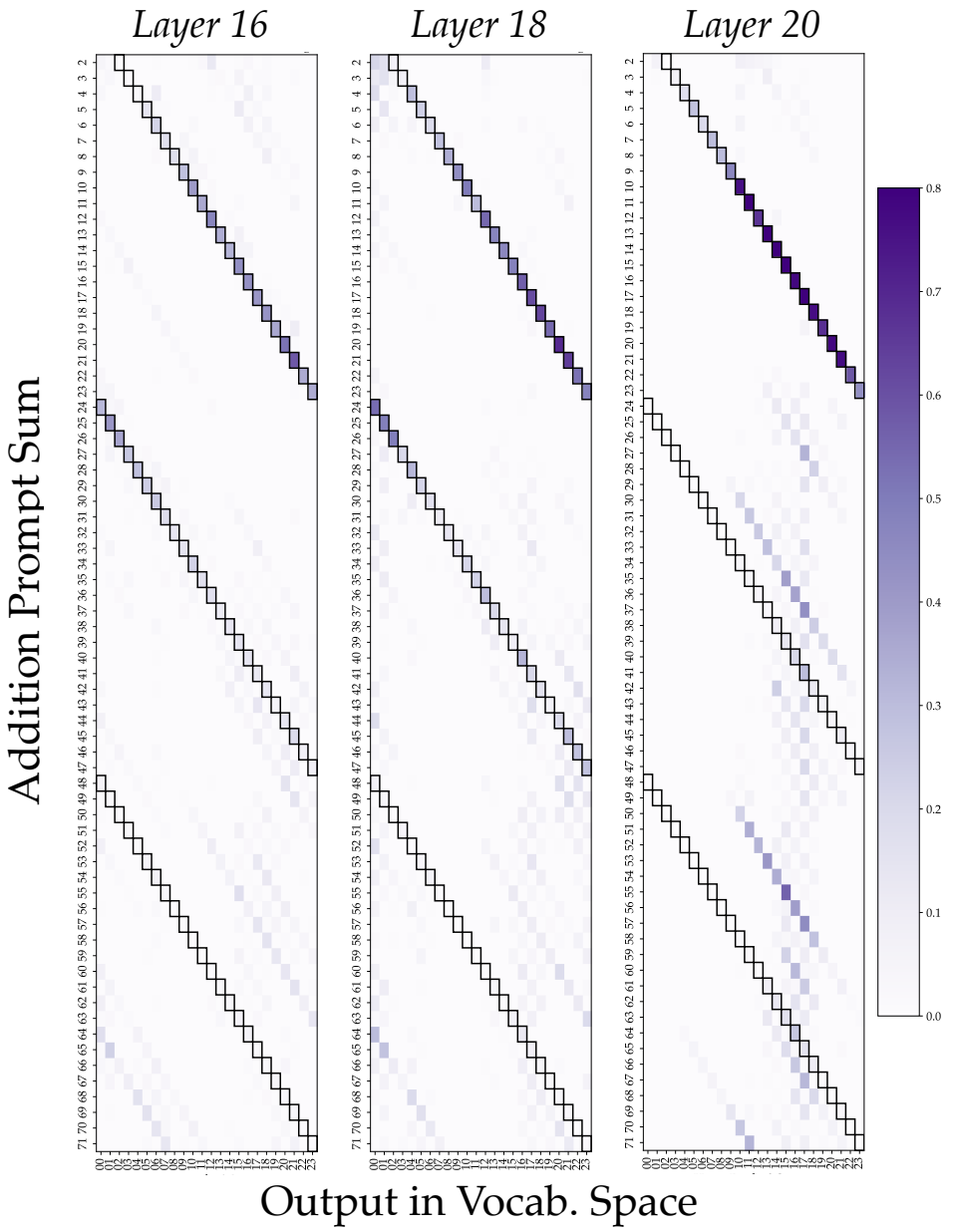


Figure 27: Patching from addition→hours at layers 16, 18, and 20. See Figure 21(b) for performance across all layers.

I Fourier Probes

I.1 Fourier Probes Training

The probes are trained on the addition task using the template `"{a}+{b}="`, where $a, b \in \{1, \ldots, 199\}$.

Let $\mathbf{h}_{a+b}^{(l)} \in \mathbb{R}^{4096}$ denote the hidden state at layer l at the final tokenn position. For each period $1 \leq T \leq 150$, we train two affine probes to follow sine and cosine harmonics by minimizing the following losses:

$$\mathsf{MSE}\left(\langle \mathbf{w}_{\sin}^{(l,T)}, \mathbf{h}_{a+b}^{(l)} \rangle + b_{\sin}^{(l,T)}, \sin\left(\tfrac{2\pi(a+b)}{T}\right)\right), \quad \mathsf{MSE}\left(\langle \mathbf{w}_{\cos}^{(l,T)}, \mathbf{h}_{a+b}^{(l)} \rangle + b_{\cos}^{(l,T)}, \cos\left(\tfrac{2\pi(a+b)}{T}\right)\right). \tag{9}$$

Each probe was trained independently for 500 epochs using Adam with learning rate 10^{-3}.

Figure 39 shows the R^2 scores across layers and periods T. Fourier features begin to appear around layer 15, which, according to the DAS experiments in Section 2, is where the model passes operand a and operand b to the final token position and begins computing the output sum. We observe significant R^2 values for $T \in \{2, 5, 10, 20\}$. For larger periods, the signal remains strong but becomes more diffuse. Following Kantamneni & Tegmark (2025), we also focus on $T \in \{50, 100\}$, as these periods exhibit high R^2 and align with the inductive bias of a base-10 number system.

When applying the trained probes to the layer 18 residual activations at the final token position, the outputs follow a clear sinusoidal pattern, as expected (Figure 28). Taking the two probe outputs corresponding to the sine and cosine components for a given period T yields a circular pattern with that period (Figure 29). Although the probes are trained only on the addition task, we observe similar patterns when applying them to activations from the hours task (Figures 30 and 31), the months task (Figures 32 and 33), and the weekdays task (Figures 34 and 35).

I.2 Fourier Probe Analysis

Fourier Probes are Orthogonal to Each Other Figure 40 shows the cosine similarity between all probes for $T \in \{2, 5, 10, 20, 50, 100\}$ at layer 18. As can be seen, almost all probes are orthogonal to one another, with the exception of the $\cos(20)$ and $\sin(20)$ probes, whose cosine similarity is -0.21.

Fourier Probes Overlap with DAS Subspaces We measure the overlap between the Fourier probes and the output concept DAS subspaces identified in Section 2 for both the addition and cyclic tasks. We define the overlap score ω between a probe $\mathbf{w}^{(T)}$ and a DAS subspace $\mathbf{S}$ as:

$$\omega(\mathbf{w}^{(T)}, \mathbf{S}) = \frac{\|\mathbf{S}\mathbf{S}^\top \mathbf{w}^{(T)}\|}{\|\mathbf{w}^{(T)}\|} \tag{10}$$

For each period T, we report the average overlap across the sine and cosine probes. The results are shown in Figure 38. Notably, the Fourier probes exhibit substantial overlap with the DAS subspaces, despite being trained to find sinusoidal directions in the full residual stream and were trained only on the addition task. Interestingly, each task exhibits overlap with probes of different periods. While most tasks show overlap with probes for $T \in \{2, 5, 10, 20, 50\}$, only the addition task also overlaps with probes at period $T = 100$. This can be explained by the fact that the addition task is the only task with prompts where the output can exceed 100.

Table 5: Steering targets and selected Fourier periods for each task. Periods are chosen based on overlap with DAS output subspaces (Section 2, Figure 38).

Task	**Steering Targets**	**Steered Periods** T
Weekdays	$\{0, \ldots, 6\}$	$\{2, 5, 10\}$
Months	$\{1, \ldots, 12\}$	$\{2, 5, 10, 20, 50\}$
Hours	$\{0, \ldots, 23\}$	$\{2, 5, 10, 20, 50\}$
Addition	$\{0, \ldots, 23\}$	$\{2, 5, 10, 20, 50, 100\}$

I.3 Steering With Fourier Probes

To test whether the model uses the Fourier features, we evaluate whether the directions learned by the probes can be used to steer its predictions on the addition, month, hour, and weekday tasks. More concretely, we use the probes to steer the model's prediction from a value n to a counterfactual value n' (pre-modulo). For example, if we steer the model on the months task to $n' = 6$, we expect it to predict "June", regardless of the prompt that we are steering on.

For the full steering algorithm, see Algorithm 1. For the cyclic tasks, we use the same dataset described in Table 1. For the addition task, we use a subset in which $a, b \in [0, 10]$. Table 5 provides details on the steering targets and selected periods for each task.

For each task, we steer every prompt in the dataset to each target value in turn. The results are shown in Figure 7. Each row corresponds to the average output probabilities after steering all prompts toward a given target. A strong diagonal pattern indicates that the target token receives high probability after steering, while other tokens receive low probability.

When projecting the layer 18 residual activations at the final token position onto the probe directions, the outputs follow a clear sinusoidal pattern, as expected (Figure 28). Furthermore, projecting onto the pair of directions corresponding to the sine and cosine components for a given period T yields a circular pattern with that period (Figure 29). Although the probes are trained only on the addition task, we observe similar patterns when projecting activations from the hours task (Figures 30 and 31), the months task (Figures 32 and 33), and the weekdays task (Figures 34 and 35).

Steering Performance Across Individual Prompts. Figure 37 presents the steering results when applying the intervention to individual prompts in the months task. While steering performs well on average across targets, the per-prompt results are noticeably noisier. For example, steering the prompt "Three months after January" (top row, third column) largely fails, whereas steering "Seven months after January" (second row, left) performs well across most targets. Together with the need for a scaling factor, this suggests that the Fourier features at layer 18 alone do not fully override downstream computation.

I.4 Circular Probes

Engels et al. (2025) suggested training circular probes for cyclic tasks by first reducing activations with PCA ($d_{\text{PCA}} = 5$) and then fitting the probes via least squares, an approach that is particularly effective for small datasets.

The target in this approach is the same as in I.1:

$$\text{MSE}\left(\langle \mathbf{w}_{\text{sin}}^{(l,T)}, \mathbf{h}_{a+b}^{(l)} \rangle, \sin\left(\tfrac{2\pi(a+b)}{T}\right)\right), \quad \text{MSE}\left(\langle \mathbf{w}_{\cos}^{(l,T)}, \mathbf{h}_{a+b}^{(l)} \rangle, \cos\left(\tfrac{2\pi(a+b)}{T}\right)\right). \tag{11}$$

Here, $\mathbf{h}_{a+b}^{(l)} \in \mathbb{R}^{d_{\text{PCA}}}$ denotes the PCA-reduced hidden state at layer l corresponding to the final token, and the probe parameters are given by $\mathbf{w}_{\text{sin}}^{(l,T)}, \mathbf{w}_{\cos}^{(l,T)} \in \mathbb{R}^{1 \times d_{\text{PCA}}}$.

Algorithm 1 Activation Steering via Fourier Probes

Require: Base residual stream activation $\mathbf{h} \in \mathbb{R}^d$; steering target k; set of Periods $\mathcal{T}$; steering factor α; learned probe parameters $\{(\mathbf{w}_{\sin}^{(t)}, b_{\sin}^{(t)}, \mathbf{w}_{\cos}^{(t)}, b_{\cos}^{(t)})\}_{t \in \mathcal{T}}$

$\tilde{\mathbf{h}} \leftarrow \mathbf{h}$ {Copy of base activation}
for each period $t \in T$ **do**
  ▷ *Compute original Fourier radius*
  $\hat{s}_t \leftarrow \mathbf{w}_{\sin}^{(t)} \cdot \mathbf{h} + b_{\sin}^{(t)}; \quad \hat{c}_t \leftarrow \mathbf{w}_{\cos}^{(t)} \cdot \mathbf{h} + b_{\cos}^{(t)}$
  $r_t \leftarrow \sqrt{\hat{s}_t^2 + \hat{c}_t^2}$
  ▷ *Compute target Fourier coefficients*
  $\theta_t^* \leftarrow \frac{2\pi k}{t}$ {Target angle on Fourier circle}
  $s_t^* \leftarrow \alpha\, r_t \sin(\theta_t^*); \quad c_t^* \leftarrow \alpha\, r_t \cos(\theta_t^*)$
  ▷ *Patch sine direction*
  $\hat{s}_t \leftarrow \mathbf{w}_{\sin}^{(t)} \cdot \tilde{\mathbf{h}} + b_{\sin}^{(t)}$
  $\tilde{\mathbf{h}} \leftarrow \tilde{\mathbf{h}} + \frac{s_t^* - \hat{s}_t}{\|\mathbf{w}_{\sin}^{(t)}\|^2}\, \mathbf{w}_{\sin}^{(t)}$
  ▷ *Patch cosine direction*
  $\hat{c}_t \leftarrow \mathbf{w}_{\cos}^{(t)} \cdot \tilde{\mathbf{h}} + b_{\cos}^{(t)}$
  $\tilde{\mathbf{h}} \leftarrow \tilde{\mathbf{h}} + \frac{c_t^* - \hat{c}_t}{\|\mathbf{w}_{\cos}^{(t)}\|^2}\, \mathbf{w}_{\cos}^{(t)}$
end for
Replace $\mathbf{h}$ with $\tilde{\mathbf{h}}$ at the hook layer's last-token position
return $\mathrm{softmax}\big(\mathrm{Model}(\text{input};\ \mathbf{h} \to \tilde{\mathbf{h}})\big)$

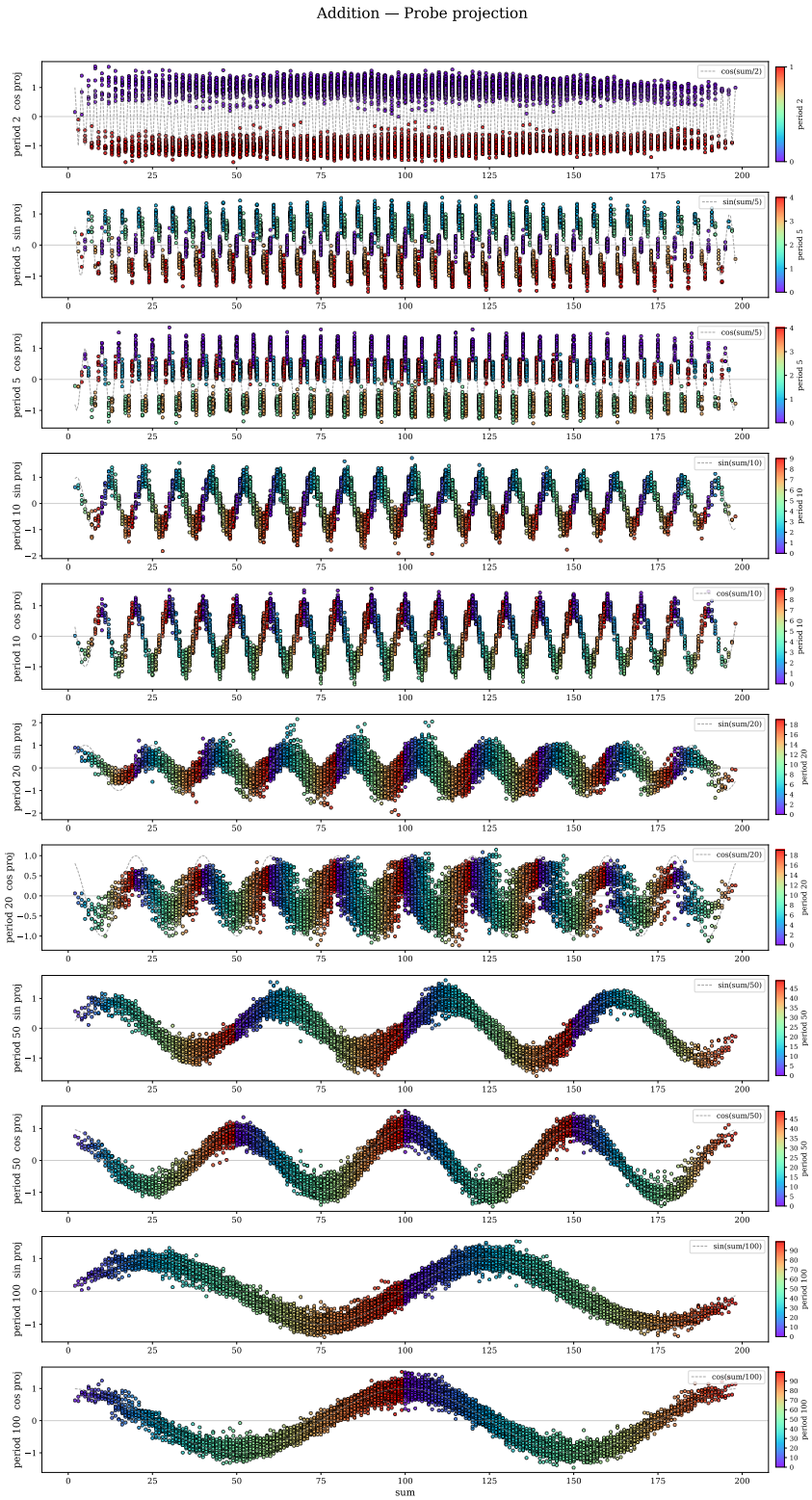


Figure 28: Projection of the layer 18 residual activations at the final token position onto the Fourier probe directions.

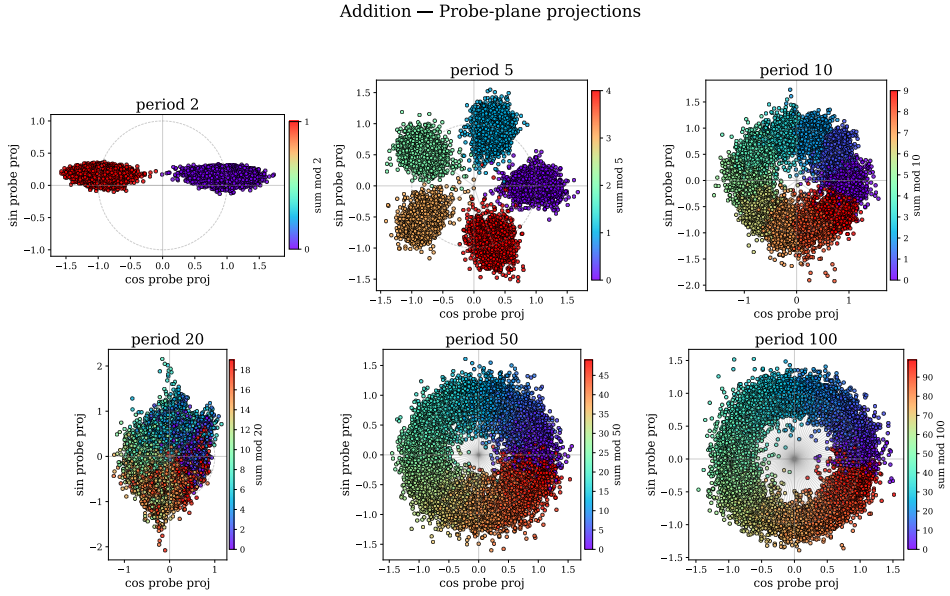


Figure 29: Projection of the layer 18 residual activations at the final token position onto the Fourier probe planes.

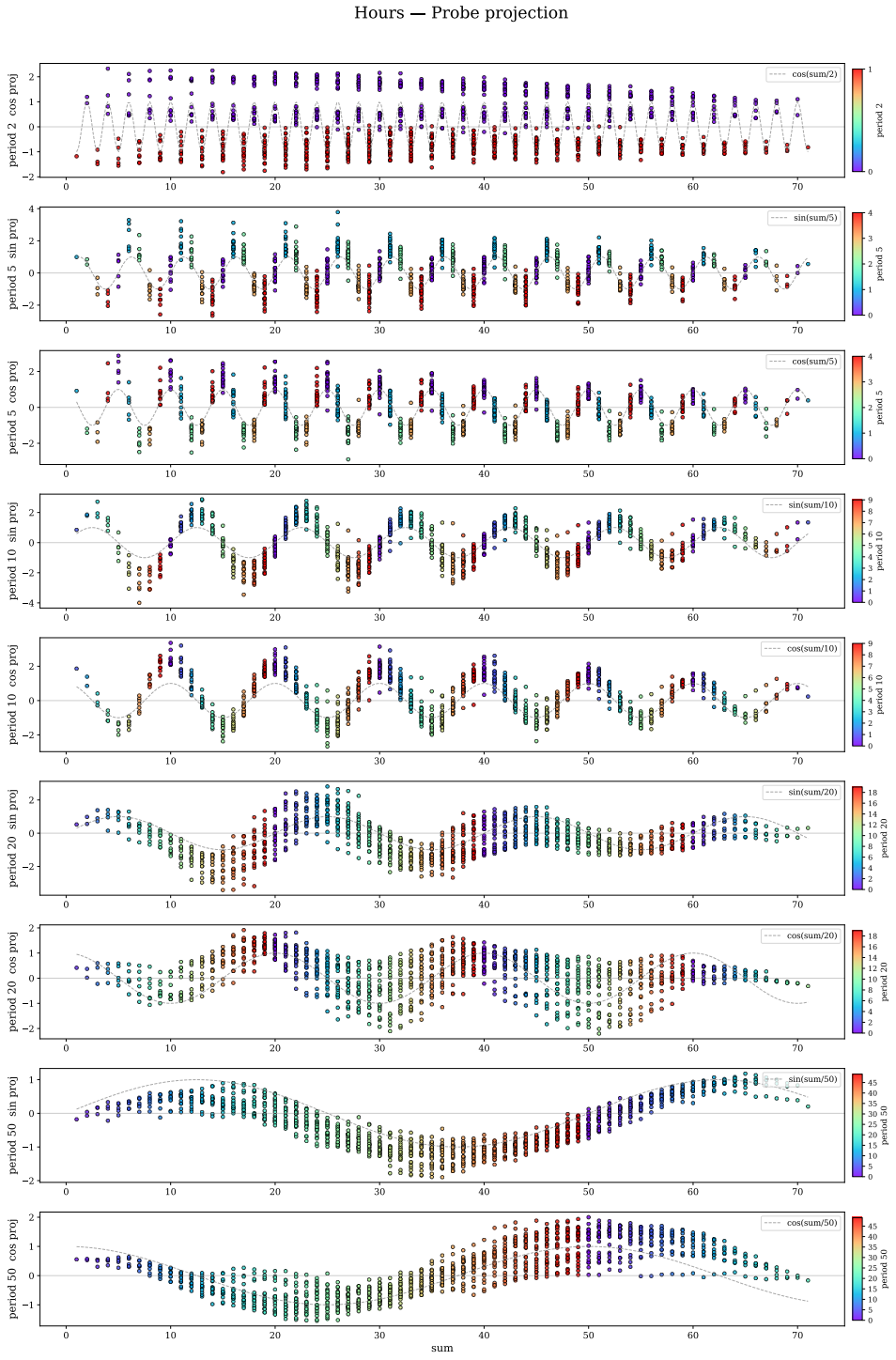


Figure 30: Projection of the layer 18 residual activations at the final token position for the hours task onto the Fourier probe directions learned from the addition task.

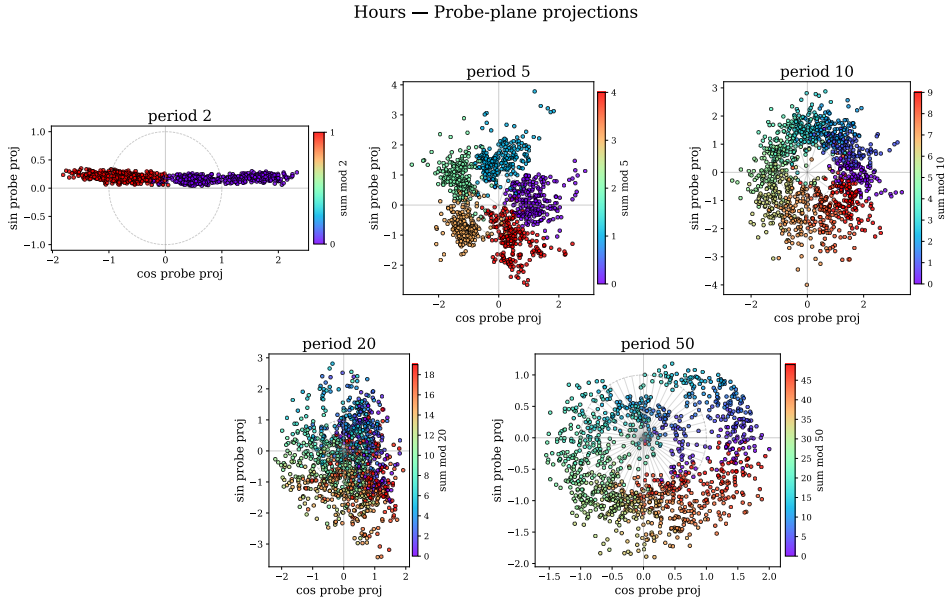


Figure 31: Projection of the layer 18 residual activations at the final token position for the hours task onto the Fourier planes learned from the addition task.

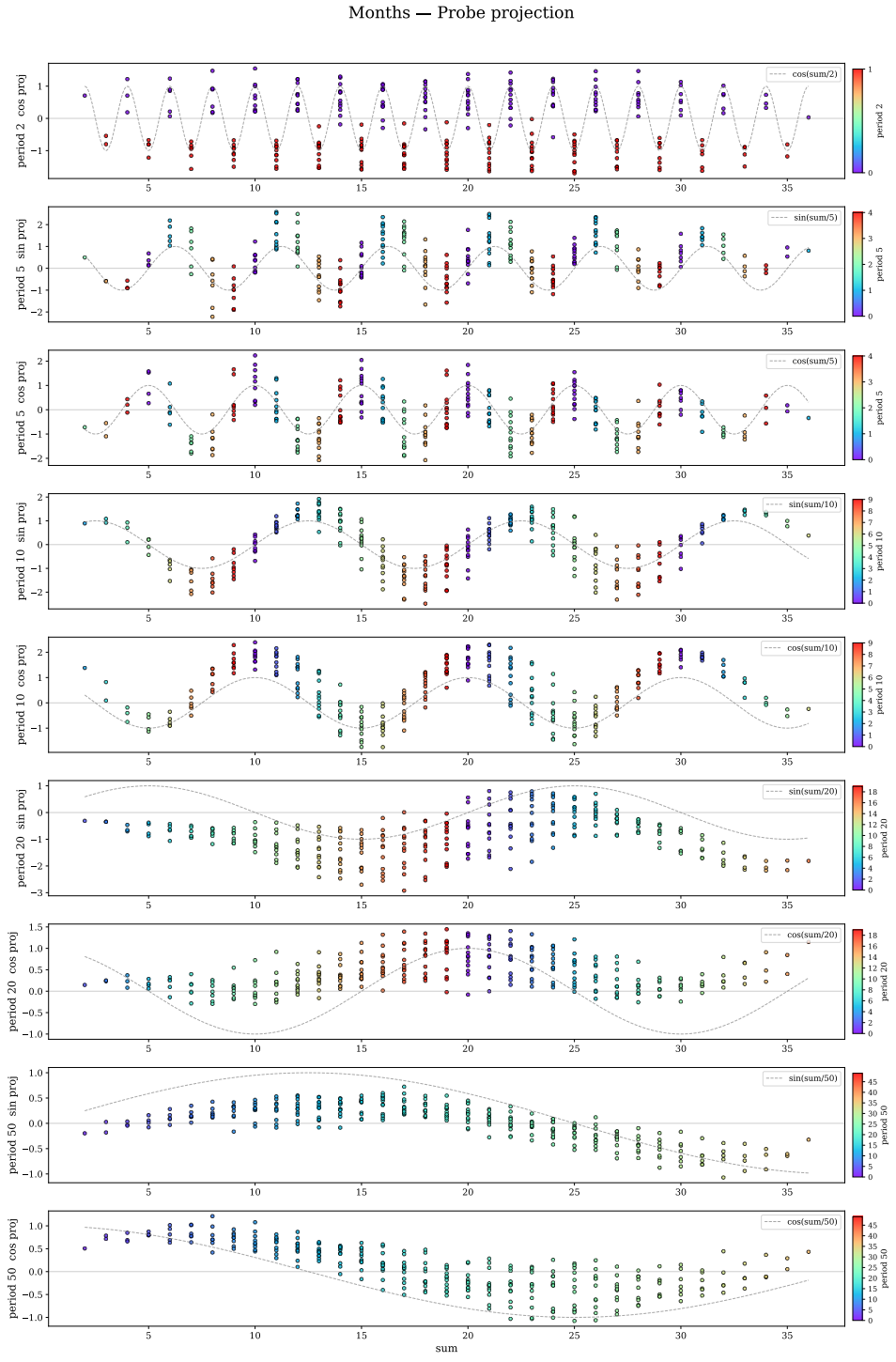


Figure 32: Projection of the layer 18 residual activations at the final token position for the months task onto the Fourier probe directions learned from the addition task.

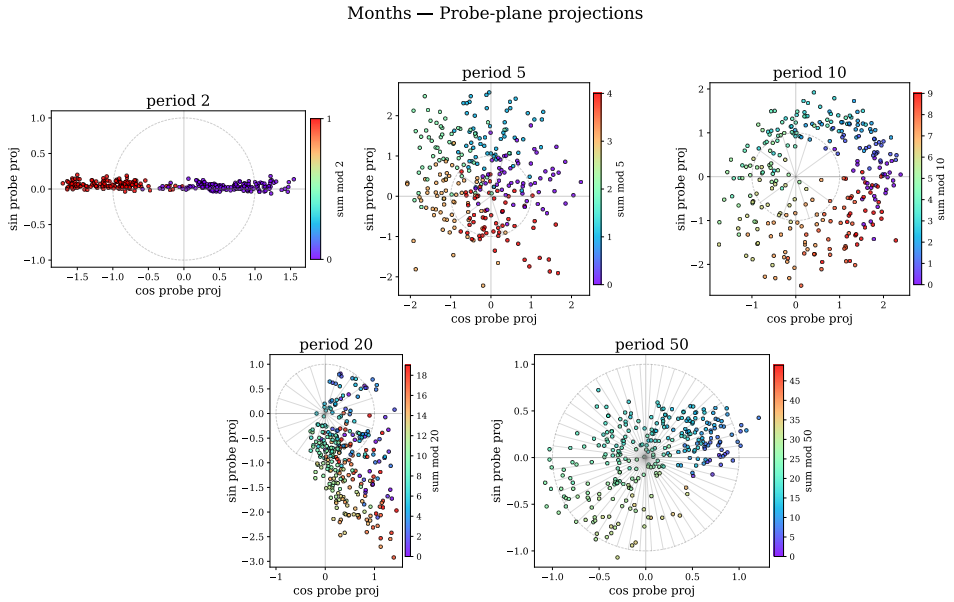


Figure 33: Projection of the layer 18 residual activations at the final token position for the months task onto the Fourier planes learned from the addition task.

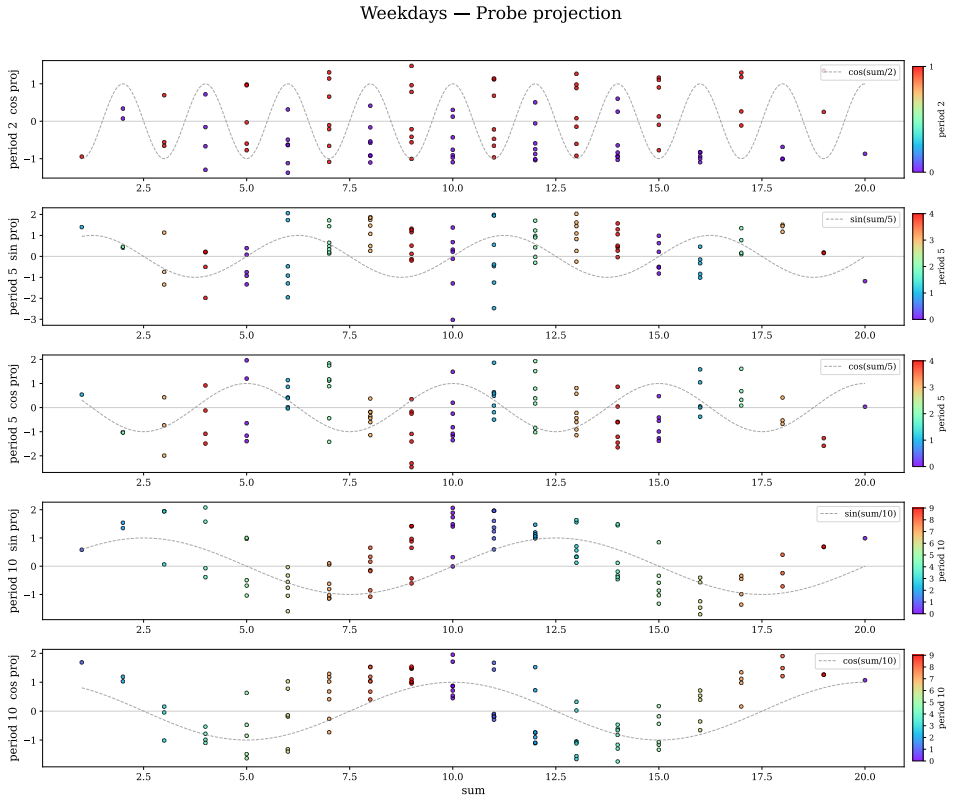


Figure 34: Projection of the layer 18 residual activations at the final token position for the weekdays task onto the Fourier probe directions learned from the addition task.

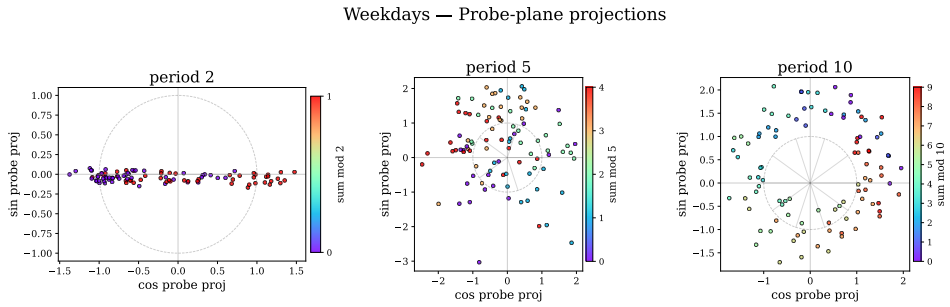


Figure 35: Projection of the layer 18 residual activations at the final token position for the weekdays task onto the Fourier planes learned from the addition task.

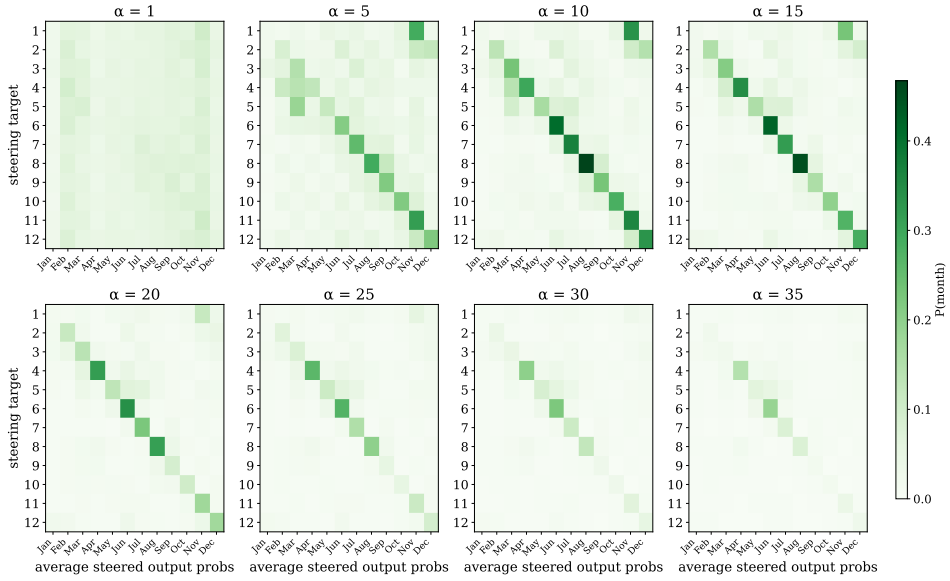


Figure 36: Average output probabilities after steering with Fourier probes at the output of layer 18 for different steering factors α. For each original prompt, we steer toward a numeric target n' by modifying its Fourier features to encode that value, and average the resulting output distributions across targets. A strong diagonal indicates a successful intervention.

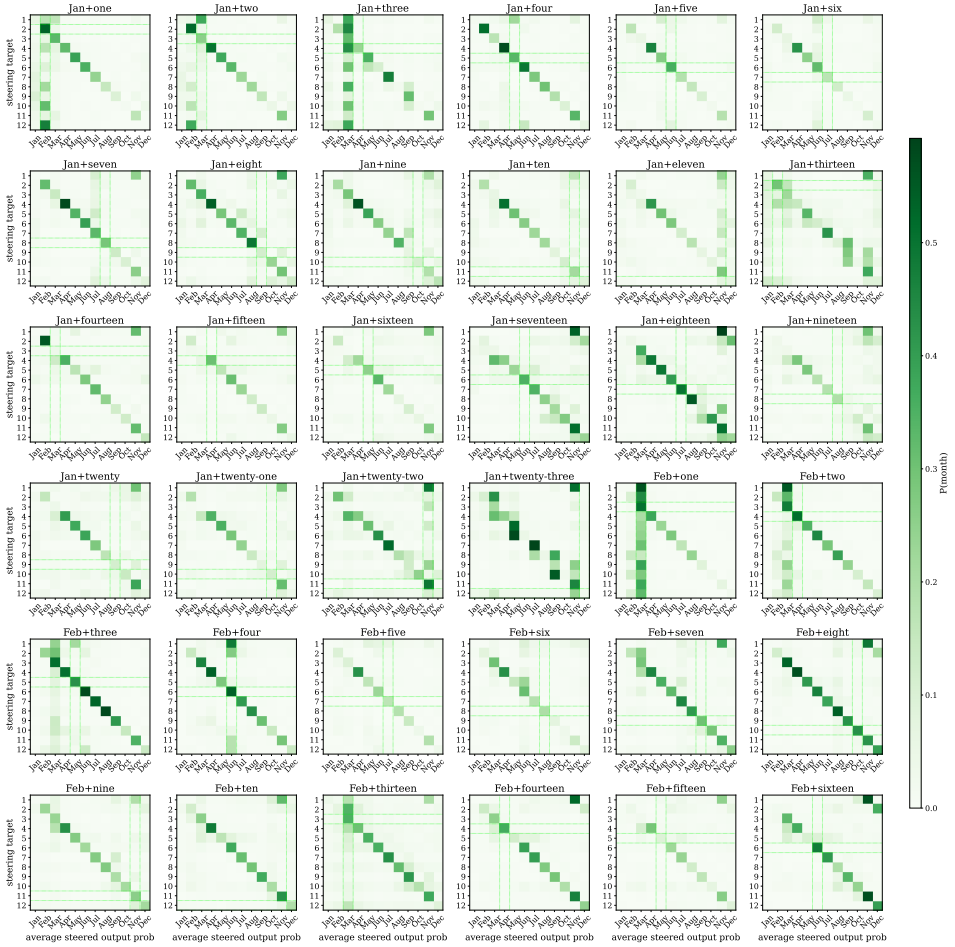


Figure 37: Output probabilities after steering each prompt with Fourier probes at the output of layer 18, using a steering factor of $\alpha = 10$. Each heatmap corresponds to a specific prompt and shows the output probabilities after applying the steering intervention to that prompt, with each row showing the probabilities after steering toward a particular target. A strong diagonal indicates a successful intervention. As shown, the effectiveness of steering varies across prompts. For example, steering the prompt "Three months after January" (top row, third column) largely fails, whereas steering "Seven months after January" (second row, left) performs well across most targets.

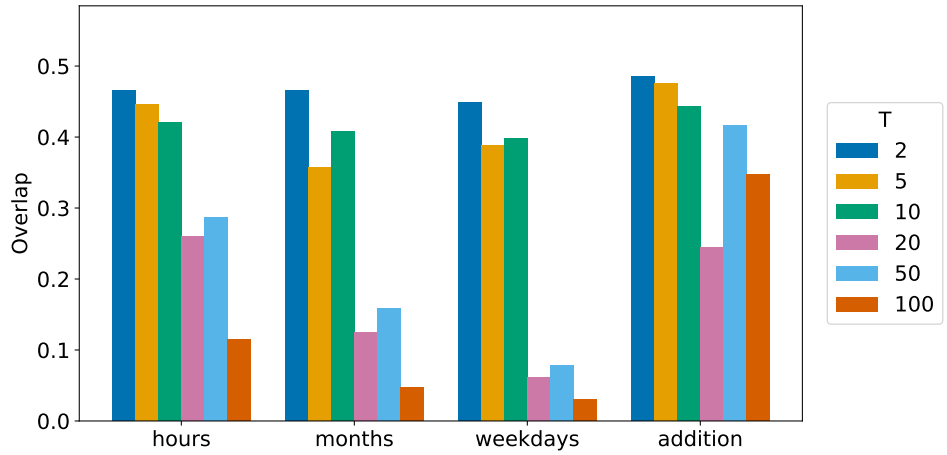


Figure 38: Overlap between addition Fourier probes and DAS output concept subspaces at layer 18. For each period T, we report the average overlap of $\mathbf{w}_{\cos}^{(T)}$ and $\mathbf{w}_{\sin}^{(T)}$ with each DAS subspace, similar to Eq. 6 (where the subspace is defined by the span of these two probes). For `months` and `weekdays`, only Fourier probes for $T \leq 10$ overlap significantly with DAS output subspaces. This is likely because the maximum sums we train on for these tasks are 36 and 21 respectively. Even for `hours`, which has a larger output range (up to 72), overlap with $T = 100$ is much lower than it is for `addition`, for which the largest sum is 200.

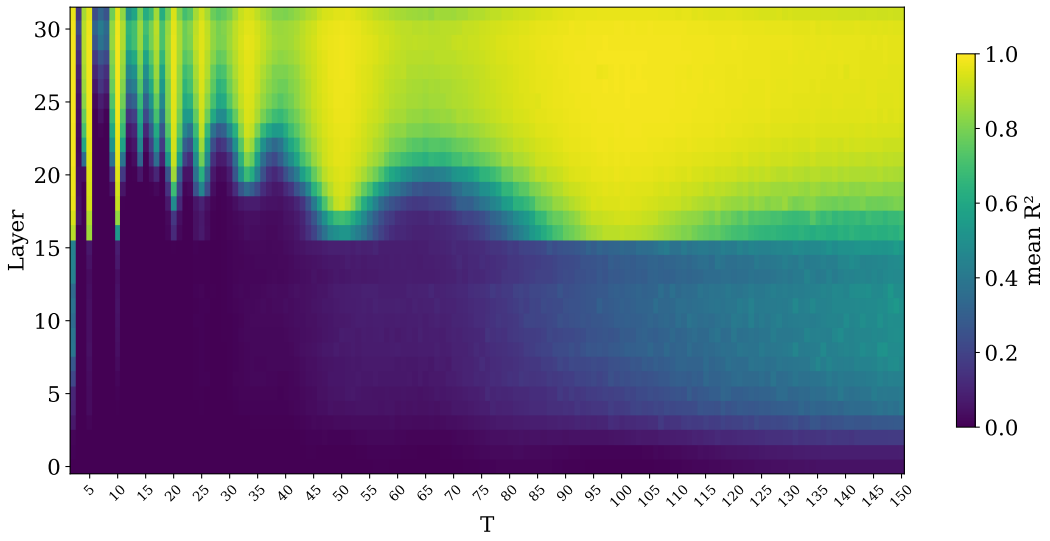


Figure 39: R^2 scores for Fourier probes across layers and for each $T \in \{2, \ldots, 150\}$. For each T, we train sine and cosine probes and report the average R^2.

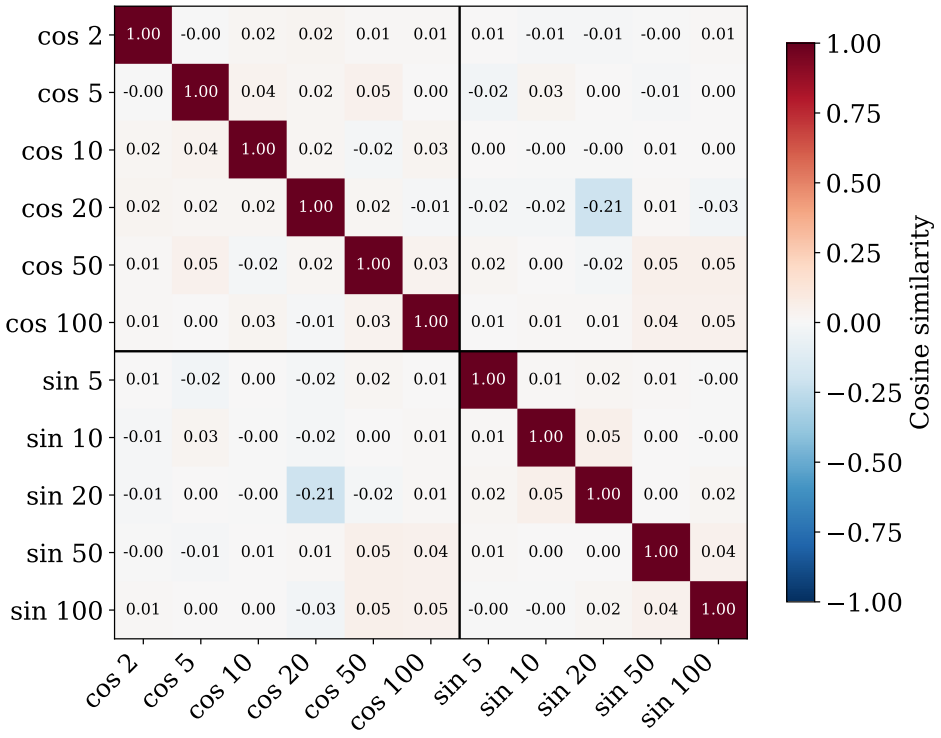


Figure 40: Cosine similarity scores for Fourier probes at layer 18. All probe directions are orthogonal, except for $\cos(20)$ and $\sin(20)$, which have cosine similarity -0.21.

J The Shared MLP Addition Module

J.1 Identifying Addition Neurons

Figure 41 shows the distribution of write scores for all layer 18 MLP neurons (Section 5.1): this score measures the proportion of a neuron's down projection row $\mathbf{d}_i$ that is within the best DAS output subspace at layer 18. We choose a threshold $\tau = 0.4$ by eye based on the `addition` task, and find that neurons in all other tasks are a subset of these 28 `addition` neurons. Table 6 gives change in performance when this set of 28 neurons is ablated, as well as when all other neurons at layer 18 are ablated. These addition neurons can explain most of the important computation at the layer 18 MLP.

Neuron ablations. We show ablation results for our canonical tasks in Table 6. We also test ablation on unseen templates in Tables 7,8,9, and 10, finding that these neurons are also important for rephrasings of the same task. Figure 42 shows errors for Llama-3.1-8B on the `addition` task, compared to errors when all MLP neurons at layer 18 are zero-ablated except for $\mathcal{N}_{\text{add}}$; failure for larger numbers suggests that we may be missing larger period neurons.

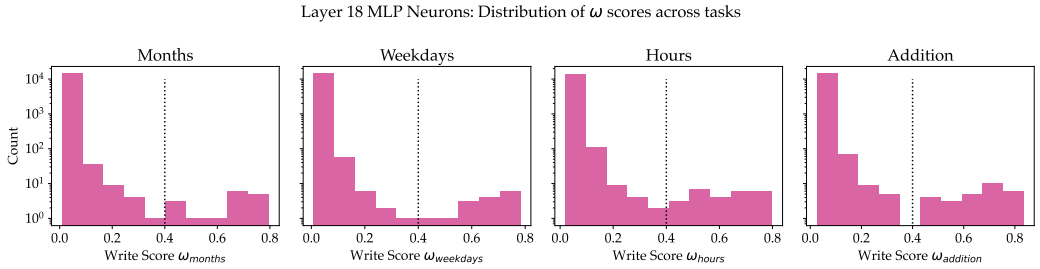


Figure 41: Distribution of write scores ω for layer 18 MLP neurons. A higher score means that this neuron's down projection row $\mathbf{d}_i$ is within the best output DAS subspace for this task. We pick the threshold $\tau = 0.4$ by eye based on the `addition` task. There are 28 neurons with $\omega > 0.4$ for the `addition` task, 16 for `months`, 15 for `weekdays`, and 26 for `hours`. All neurons for cyclic tasks are contained within the set of `addition` neurons, except for a single `hours` neuron.

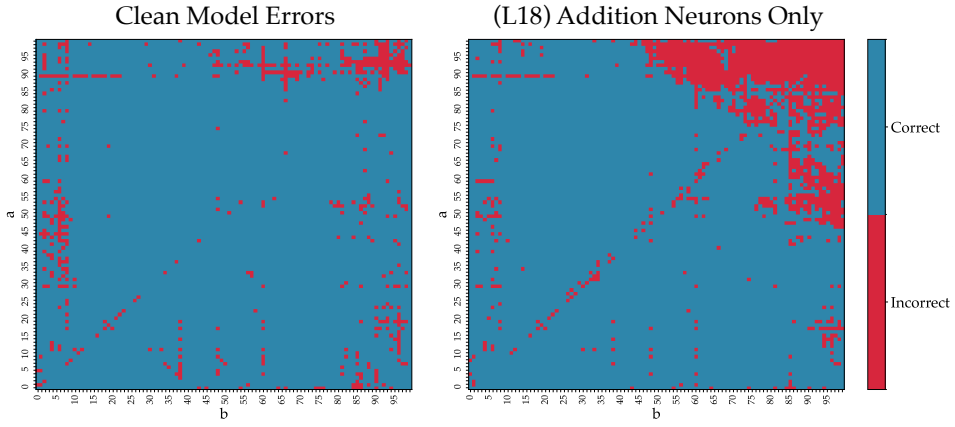


Figure 42: Llama-3.1-8B errors on the `addition` task. We show errors for a clean model run (95% accuracy), as well as errors when all neurons at the L18 MLP are zero-ablated except for our 28 addition neurons $\mathcal{N}_{\text{add}}$ (86% accuracy). Most errors come from higher number ranges, suggesting that $\mathcal{N}_{\text{add}}$ excludes some neurons with larger periods.

Table 6: Accuracy (%) intervening on the set of 28 `addition` neurons $\mathcal{N}_{\text{add}}$. **Clean**: Llama-3.1-8B accuracy on this task. **Only $\mathcal{N}_{\text{add}}$**: accuracy when all other neuron activations at the layer 18 MLP are set to 0, except for neurons in $\mathcal{N}_{\text{add}}$. Despite these neurons making up 0.2% of layer 18 neurons, accuracy remains high. **Zeroed**: accuracy drops significantly when these neurons' activations are set to zero. **Flipped**: reversing the sign of these neurons' activations is destructive, likely because they represent periodic functions that oscillate between negative and positive values across inputs.

Task	Number Range	Clean	Only $\mathcal{N}_{\text{add}}$	Zeroed	Flipped	*Chance*
Weekdays	1→7	91.84	89.80	40.82	20.41	*14.29*
	8→14	68.37	58.16	28.57	17.35	*14.29*
Months	1→12	81.25	77.78	36.81	14.58	*8.33*
	13→24	64.93	70.83	23.61	16.32	*8.33*
Hours	1→24	93.06	80.56	50.69	16.32	*4.17*
	24→48	69.18	51.65	29.77	11.63	*4.17*
Addition	$a, b \in [1..100]$	94.60	86.36	24.11	2.97	*0.50*

Table 7: Neuron intervention results for the `addition` task across alternative prompt templates, for one cycle each. **Input / Offset Format** columns show examples of strings input to model. **Clean**: unmodified accuracy for Llama-3.1-8B. **Only**: accuracy when all neurons at layer 18 except for $\mathcal{N}_{\text{add}}$ are zero-ablated. **Zeroed**: accuracy when all 28 $\mathcal{N}_{\text{add}}$ addition neurons are zero-ablated. **Flipped**: accuracy when 28 $\mathcal{N}_{\text{add}}$ neurons have their signs flipped. **Random**: accuracy when 28 random non-addition neurons at layer 18 are zero-ablated (averaged across 10 seeds).

Template	Input Format	Offset Format	Clean	Only ↑	Zeroed ↓	Flipped ↓	Random ↑
(default prompt)	1, 2...	1, 2...	94.33%	86.11%	23.80%	2.92%	94.47% ± 0.17
The sum of {input} and {offset} equals	1, 2...	1, 2...	100.00%	98.87%	42.74%	5.09%	100.0% ± 0.00
Math question: {input} plus {offset} equals	1, 2...	1, 2...	96.76%	76.98%	29.30%	4.22%	96.66% ± 0.32
Q: What is the sum of {input} and {offset}? A:	1, 2...	1, 2...	99.70%	97.79%	43.93%	5.41%	99.68% ± 0.06
Q: If I have {input} apples and buy {offset} more, how many apples do I have? Total apples:	1, 2...	1, 2...	75.67%	40.63%	12.32%	1.43%	74.92% ± 2.47

Table 8: Neuron intervention results for the weekdays task across alternative prompt templates, for one cycle each. **Input / Offset Format** columns show examples of strings input to model. **Clean**: unmodified accuracy for Llama-3.1-8B. **Only**: accuracy when all neurons at layer 18 except for $\mathcal{N}_{\text{add}}$ are zero-ablated. **Zeroed**: accuracy when all 28 $\mathcal{N}_{\text{add}}$ addition neurons are zero-ablated. **Flipped**: accuracy when 28 $\mathcal{N}_{\text{add}}$ neurons have their signs flipped. **Random**: accuracy when 28 random non-addition neurons at layer 18 are zero-ablated (averaged across 10 seeds).

Template	Input Format	Offset Format	Clean	Only ↑	Zeroed ↓	Flipped ↓	Random ↑
(default prompt)	Monday, Tuesday...	one, two...	91.84%	89.80%	40.82%	20.41%	92.45% ± 1.31
Let's do some weekday math. {offset} days after {input} is what day? Answer:	Monday, Tuesday...	one, two...	73.47%	73.47%	30.61%	16.33%	73.47% ± 1.83
Question: Today is {input}. In {offset} days, what day will it be? Answer: It will be	Monday, Tuesday...	1, 2...	97.96%	85.71%	44.90%	24.49%	95.31% ± 1.84
If today is {input}, and I have a deadline in {offset} days, what day is my deadline? My deadline is on	Mon, Tues...	one, two...	83.67%	71.43%	40.82%	20.41%	86.53% ± 1.63
Let's do some calendar math. Today is {input}. What day will it be in {offset} days? Answer:	Mon, Tues...	1, 2...	85.71%	83.67%	38.78%	30.61%	83.47% ± 1.70

Table 9: Neuron intervention results for the `months` task across alternative prompt templates, for one cycle each. **Input / Offset Format** columns show examples of strings input to model. **Clean**: unmodified accuracy for Llama-3.1-8B. **Only**: accuracy when all neurons at layer 18 except for $\mathcal{N}_{\text{add}}$ are zero-ablated. **Zeroed**: accuracy when all 28 $\mathcal{N}_{\text{add}}$ addition neurons are zero-ablated. **Flipped**: accuracy when 28 $\mathcal{N}_{\text{add}}$ neurons have their signs flipped. **Random**: accuracy when 28 random non-addition neurons at layer 18 are zero-ablated (averaged across 10 seeds).

Template	Input Format	Offset Format	Clean	Only ↑	Zeroed ↓	Flipped ↓	Random ↑
(default prompt)	January, February...	one, two...	81.25%	76.39%	36.81%	14.58%	81.32% ± 1.00
Let's do some calendar math. If the current month is {input}, then in {offset} months, what month will it be? Answer: It will be	January, February...	one, two...	97.22%	96.53%	59.03%	22.22%	96.74% ± 0.82
Question: I have a big presentation in {offset} months. If it is currently {input}, then what month is my presentation? Answer: My presentation is in	January, February...	1, 2...	70.83%	64.58%	46.53%	25.00%	70.97% ± 0.52
Q: A pregnant woman will give birth in {offset} months. If it is currently {input}, what month will she give birth? Answer: She will give birth in	Jan, Feb...	one, two...	91.67%	89.58%	49.31%	17.36%	91.88% ± 1.28
Let's figure out some scheduling. I just got my hair cut, and it is currently {input}. If I have to book another appointment in {offset} months, what month will it be? Answer: My appointment is in	Jan, Feb...	1, 2...	81.25%	78.47%	53.47%	34.03%	80.90% ± 1.33

Table 10: Neuron intervention results for the hours task across alternative prompt templates, for one cycle each. **Input / Offset Format** columns show examples of strings input to model. **Clean**: unmodified accuracy for Llama-3.1-8B. **Only**: accuracy when all neurons at layer 18 except for $\mathcal{N}_{\text{add}}$ are zero-ablated. **Zeroed**: accuracy when all 28 $\mathcal{N}_{\text{add}}$ addition neurons are zero-ablated. **Flipped**: accuracy when 28 $\mathcal{N}_{\text{add}}$ neurons have their signs flipped. **Random**: accuracy when 28 random non-addition neurons at layer 18 are zero-ablated (averaged across 10 seeds).

Template	Input Format	Offset Format	Clean	Only ↑	Zeroed ↓	Flipped ↓	Random ↑
(default prompt)	00, 01...	one, two...	97.57%	84.55%	50.87%	16.15%	97.24% ± 0.28
Let's do some clock math. If it is {input}:00 right now in 24-hour time, then what time will it be in {offset} hours? Answer:	00, 01...	one, two...	90.45%	68.40%	47.40%	13.37%	90.31% ± 1.30
When I was in the military, I learned 24-hour time. So if right now it is {input}:00, then in {offset} hours, it will be	00, 01...	one, two...	91.32%	83.68%	50.52%	18.06%	91.15% ± 0.28
Q: If it is {input} o' clock in military time, what time will it be in {offset} hours? A: In 24-hour time, it will be	00, 01...	one, two...	81.08%	73.26%	47.57%	23.26%	81.23% ± 0.31
I need to work out when to get to the airport. Q: It's {input}:00 right now in 24-hour time. My flight is in {offset} hours. What time is my flight? Answer: My flight will be at	00, 01...	1, 2...	86.46%	64.58%	34.20%	12.67%	86.22% ± 0.59

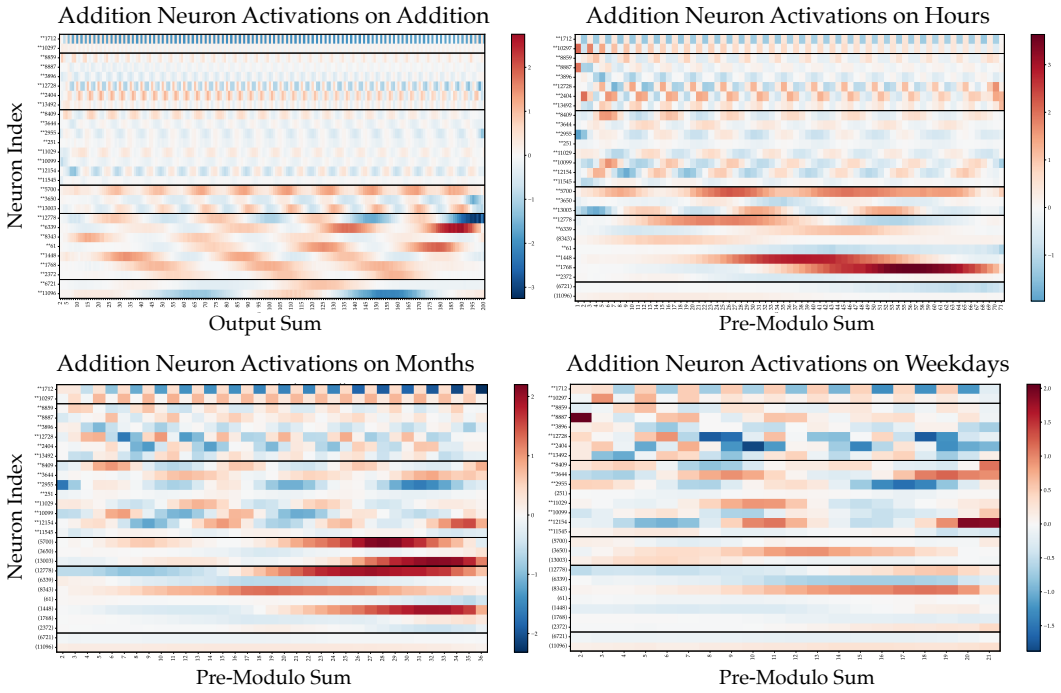


Figure 43: Addition neuron activations $\mathcal{N}_{\text{add}}$ across all examples for all tasks: `addition`, `hours`, `months`, and `weekdays`. We observe the same periodic structure across sums for all four tasks, although it is more difficult to see for tasks with smaller output ranges. Neurons that are also within the set for a respective task are **starred (e.g., n_{1712} is starred in all plots, so is used for all tasks). Addition neurons that are irrelevant for this task are marked in parentheses. Notably, `addition` neurons that are irrelevant for `months` and `weekdays` all correspond to larger m. In fact, the largest $m = 100$ neurons, n_{6721} and n_{11096}, are *only* relevant for the `addition` task.

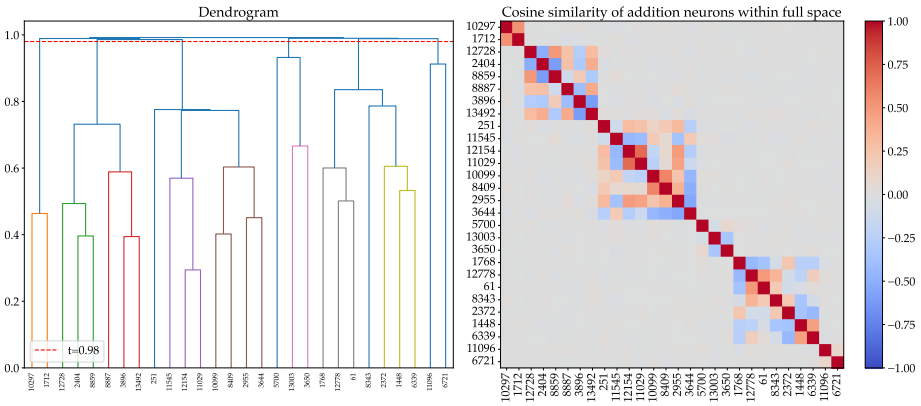


Figure 44: Simple hierarchical clustering of addition neurons with $1 - |\text{cosine\_sim}|$ as a distance metric recovers the same neuron clusters we observe in Figure 43.

J.2 Addition Neurons Group by Fourier Frequency

Figure 43 shows neuron activations averaged across output sums for all addition neurons. Averages are taken across output sums for the `addition` task and pre-modulo sums for `hours`, `months`, and `weekdays`.

J.3 Neuron Activations Across Tasks

We show activations across all tasks for three neurons: **N12728**, which is a period 5 neuron (Figure 45), **N1712**, the even parity neuron (Figure 46), and **N8409**, which is a period 10 neuron (Figure 47). The first neuron is a "split" neuron that dedicates its gate activations to the input concept and its up activations to the offset, whereas the latter two neurons read equally from both inputs. Activations are consistent across tasks, with e.g. `weekdays` being a more "zoomed in" version of tasks with larger number ranges.

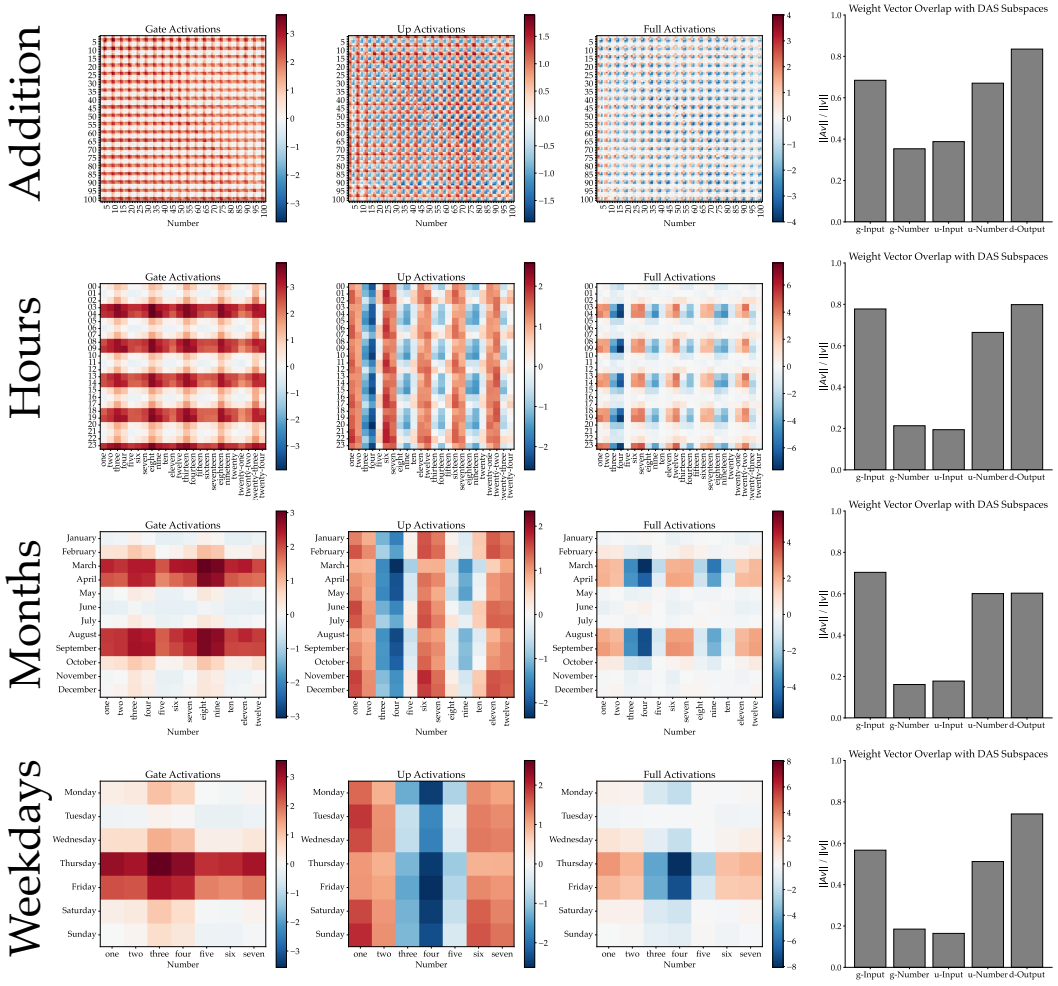


Figure 45: **N12728** activations across prompts for all four tasks, organized by input variables. This is a period 5 neuron. We also show read/write scores for $\mathbf{g}_i, \mathbf{u}_i, \mathbf{d}_i$ with input and output spaces. For all cyclic tasks, we can see that this neuron's gate vector $\mathbf{g}_{12728}$ has a much higher read scores from the input subspace (high first bar, horizontal stripes), whereas its up vector $\mathbf{u}_{12728}$ reads more heavily from the number subspace (high fourth bar, vertical stripes). Interestingly, activations are not as "split" for the `addition` task; this may be related to the fact that variables for a and b in a+b= are not causally separable by DAS, as results from Figure 17 show.

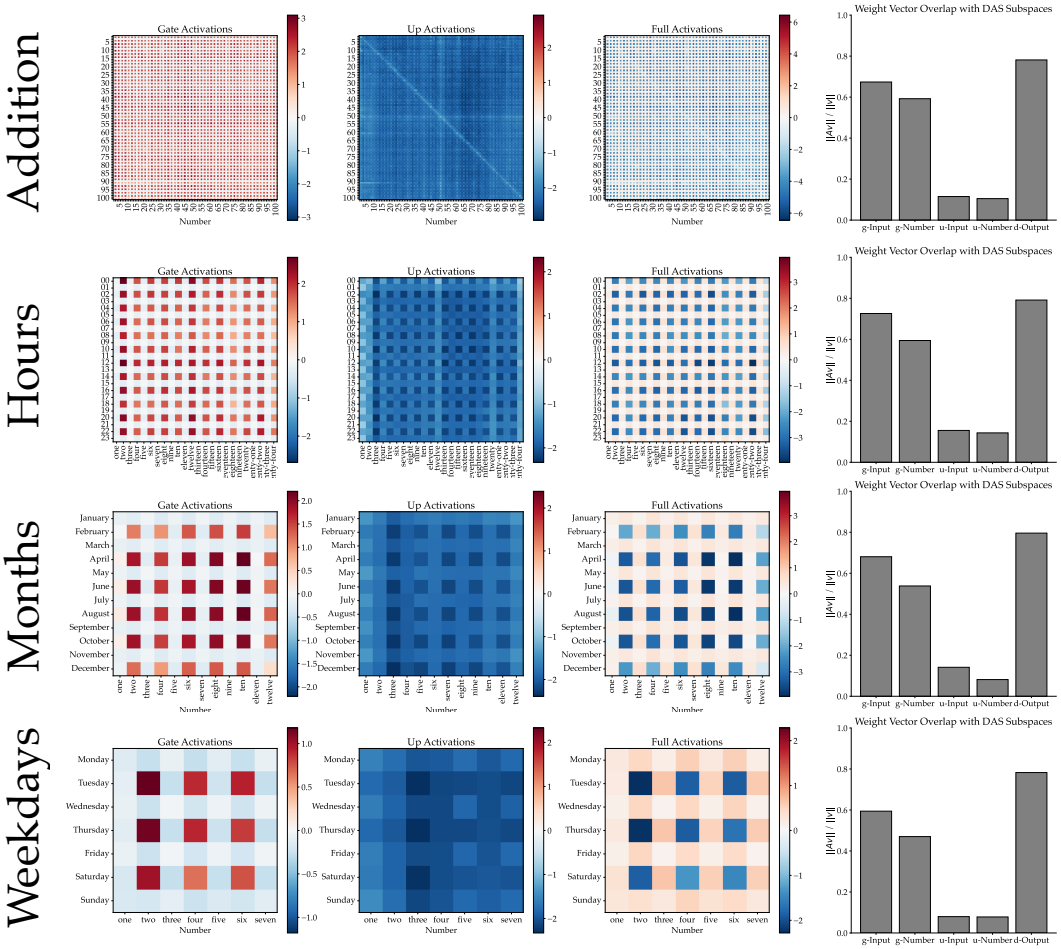


Figure 46: **N1712** activations across prompts for all four tasks, organized by input variables. This is a parity neuron. We also show read/write scores for $\mathbf{g}_i, \mathbf{u}_i, \mathbf{d}_i$ with input and output spaces. We can see that this neuron's gate vector $\mathbf{g}_{1712}$ reads equally from both input subspaces (checkered pattern across examples, and the first two bars are similar heights), whereas its up vector $\mathbf{u}_{1712}$ is mostly negative across examples, only slightly reading from the input subspaces.

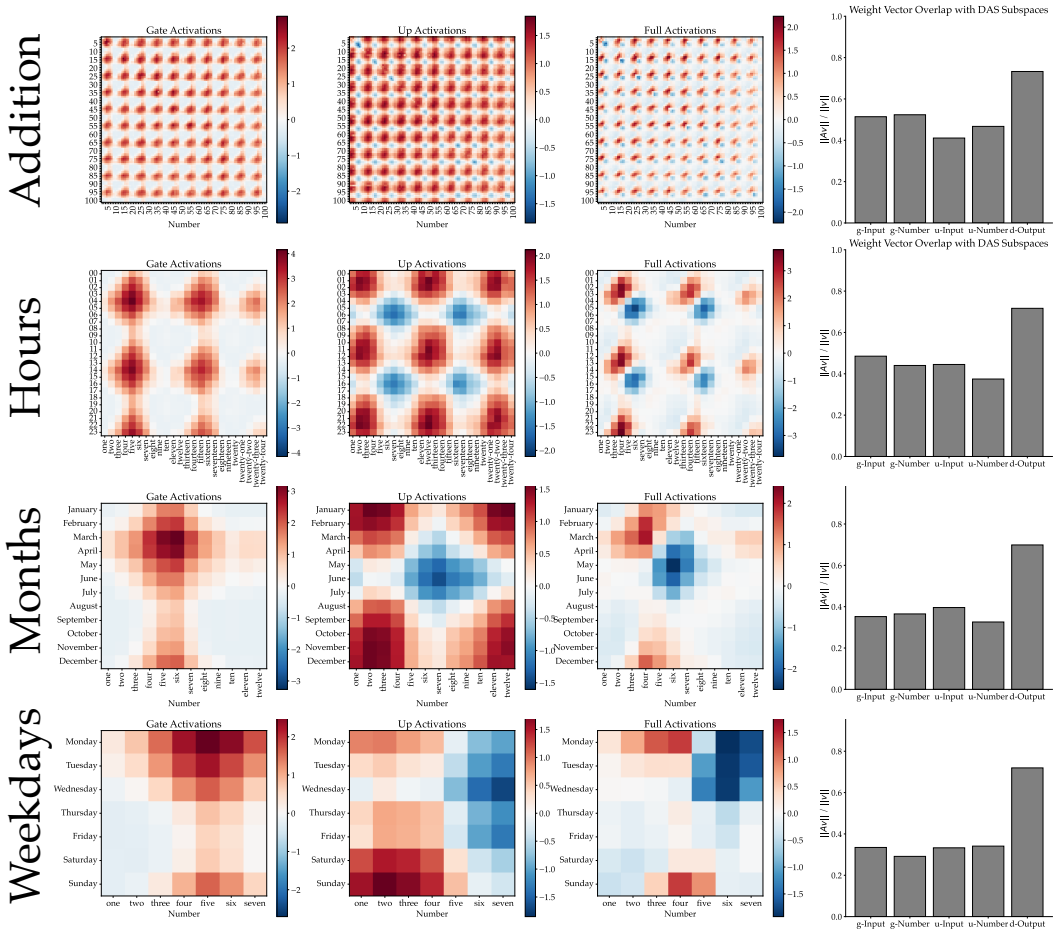


Figure 47: **N8409** activations across prompts for all four tasks, organized by input variables. This is a period 10 neuron. We also show read/write scores for $\mathbf{g}_i, \mathbf{u}_i, \mathbf{d}_i$ with input and output spaces. We can see that this neuron's gate vector $\mathbf{g}_{8409}$ reads equally from both input subspaces (checkered pattern across examples, and the first two bars are similar heights), as well as its up vector $\mathbf{u}_{8409}$.

J.4 All Addition Neurons at Layer 18 $\mathcal{N}_{\text{add}}$

Identifying split neurons. For each addition neuron, we compute "read scores" (overlap with input subspaces) following Eq. 6 with input concept and offset subspaces for $\mathbf{g}_i, \mathbf{u}_i$. We say an activation pattern is *split* for this task if, for both the gate and up projections, one of the input concept or offset overlap scores is at least 50% greater than the other score. That is, both of the below conditions should be true:

$$\max(\omega_{inputconcept}(\mathbf{g}_i), \omega_{offset}(\mathbf{g}_i)) > 1.5 \cdot \min(\omega_{inputconcept}(\mathbf{g}_i), \omega_{offset}(\mathbf{g}_i)) \tag{12}$$

$$\max(\omega_{inputconcept}(\mathbf{u}_i), \omega_{offset}(\mathbf{u}_i)) > 1.5 \cdot \min(\omega_{inputconcept}(\mathbf{u}_i), \omega_{offset}(\mathbf{u}_i)). \tag{13}$$

We find that under this threshold, 17/28 neurons in $\mathcal{N}_{\text{add}}$ have *split* activations for the `hours` task.

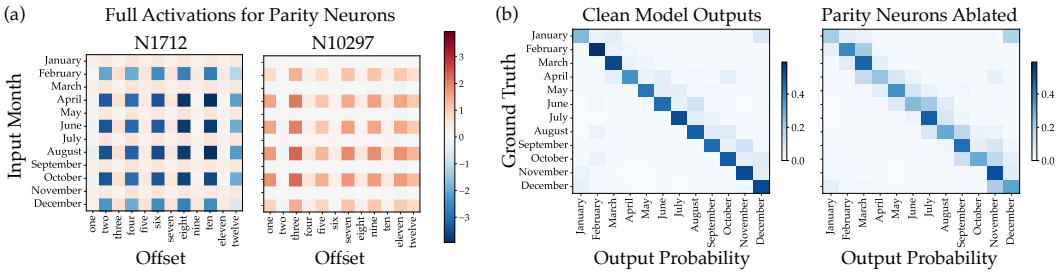


Figure 48: Parity neurons ($T = 2$) sharpen model predictions. (a) Full activations (gate · up) for both parity neurons on the `months` task. These neurons write in opposite directions for alternating prompts. (b) Zero-ablating just these two neurons shifts probability towards neighboring outputs with the wrong parity, "blurring" the model's predictions.

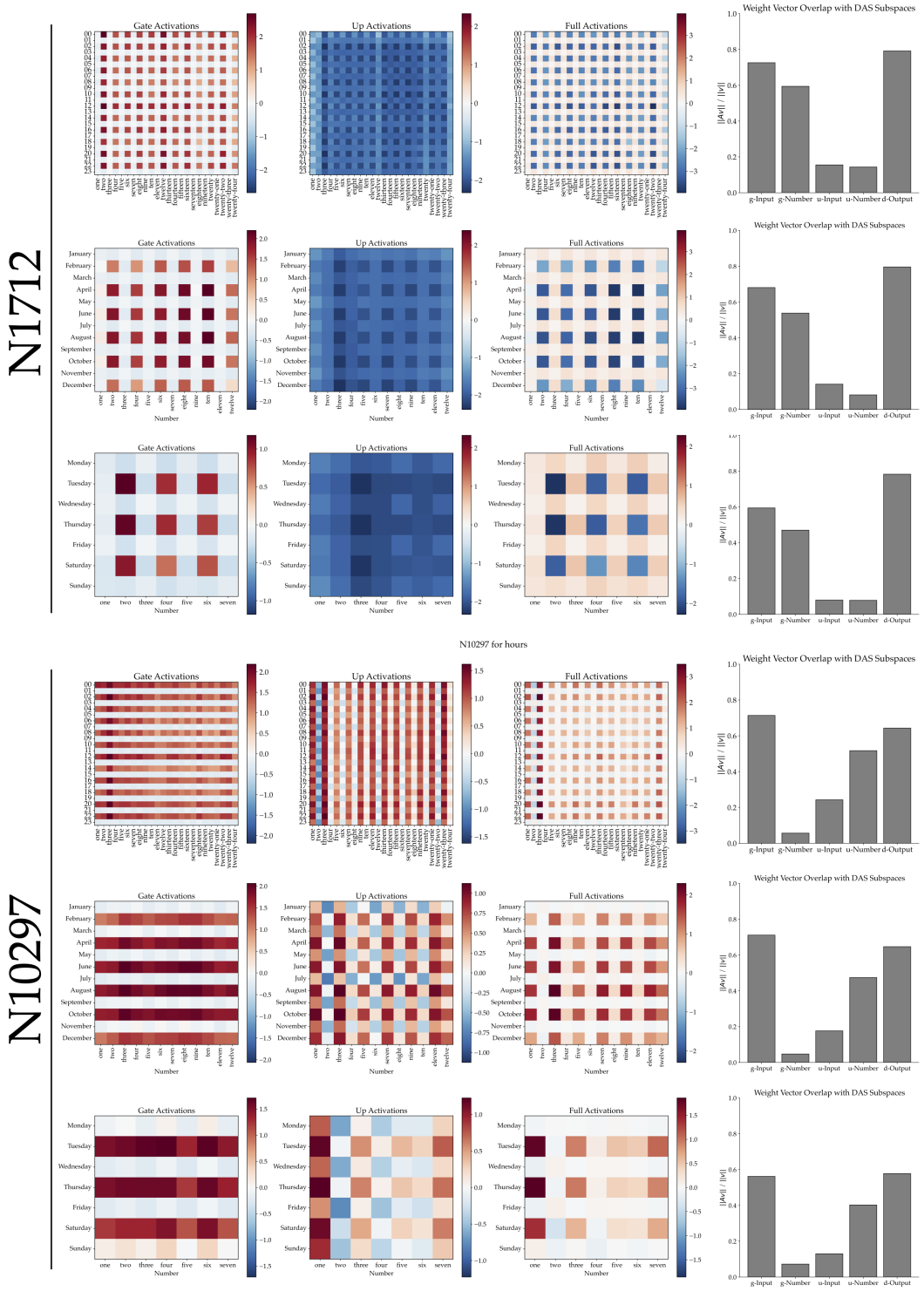


Figure 49: **All period 2 neurons**, activations for the `hours` and `months` tasks.

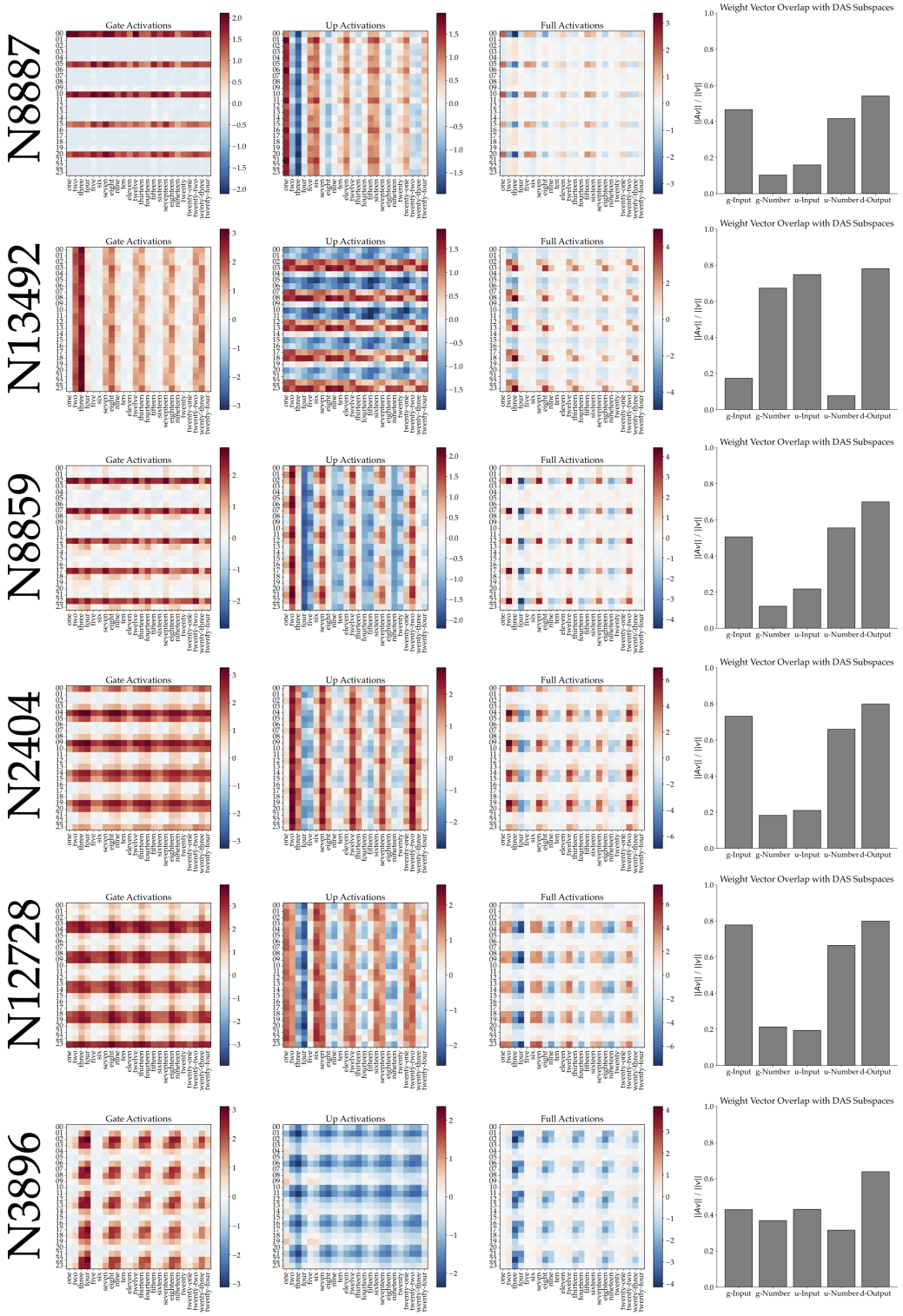


Figure 50: **All period 5 neurons**, activations for the `hours` task.

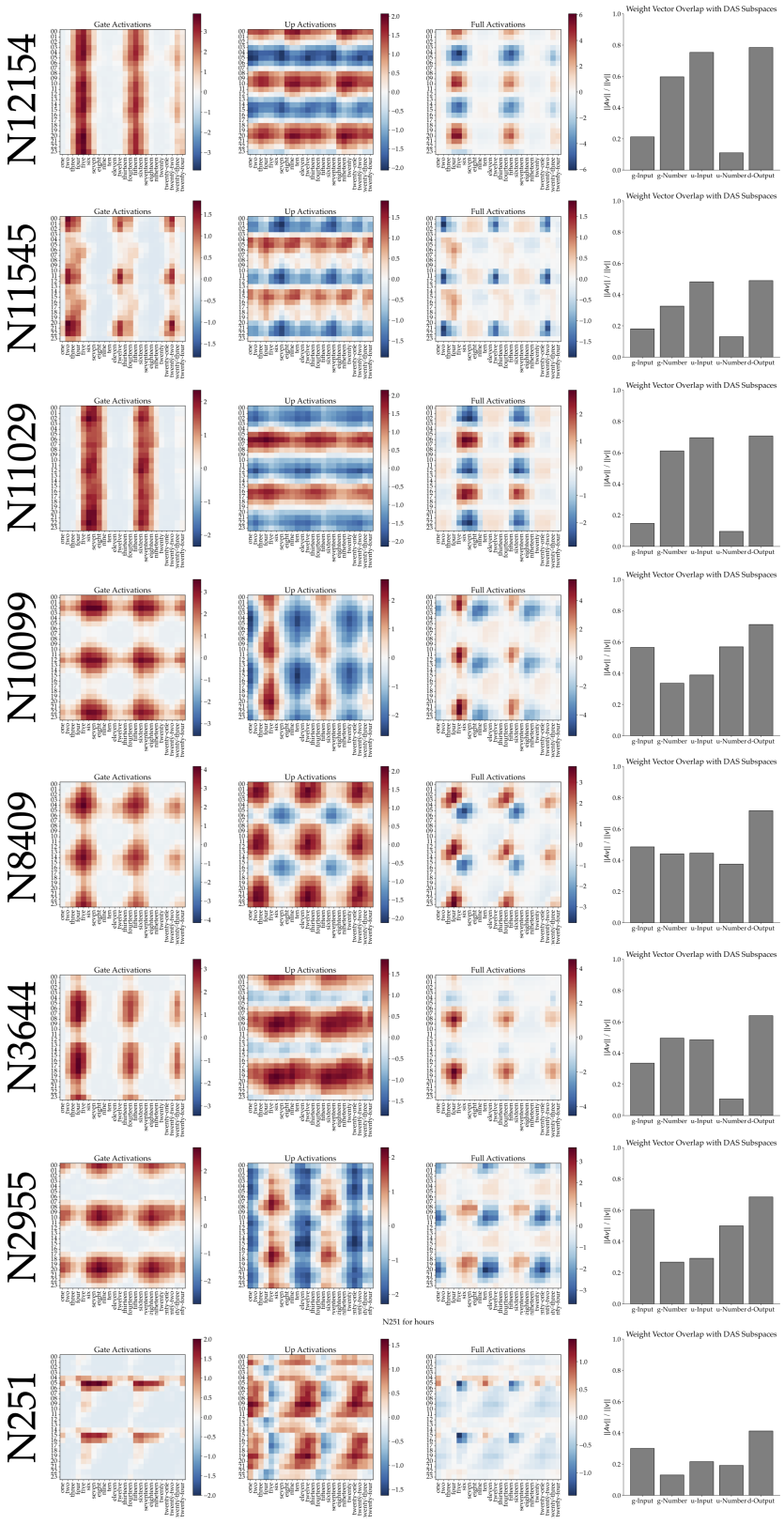


Figure 51: **All period 10 neurons**, activations for the hours task.

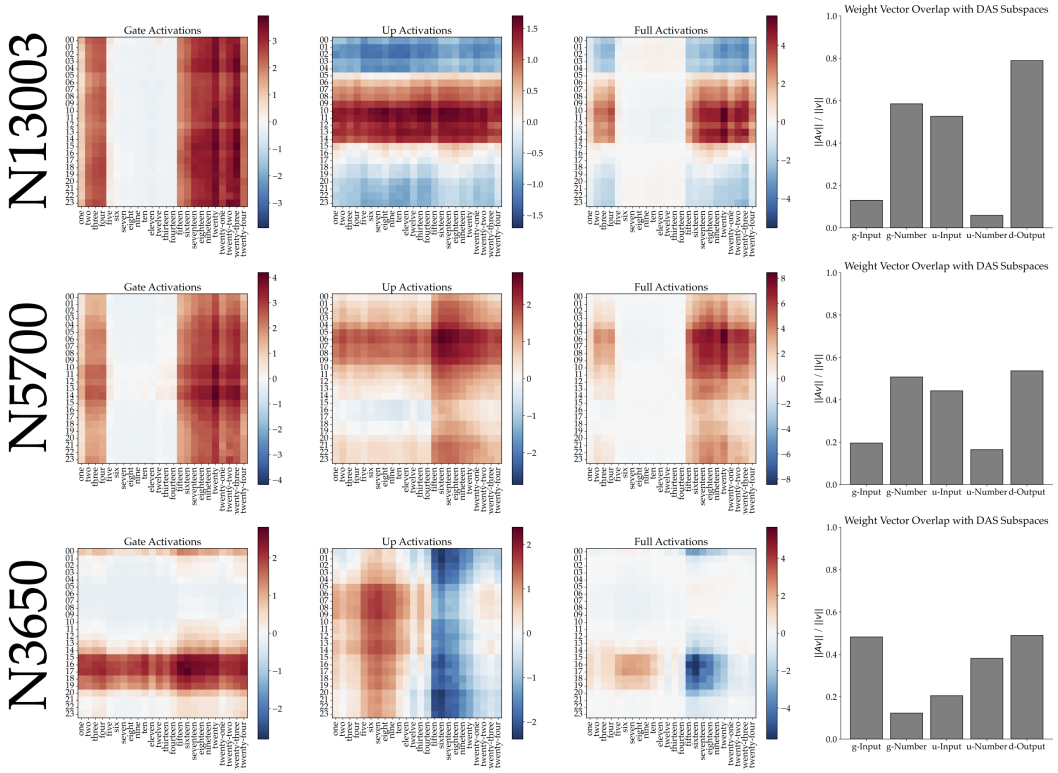


Figure 52: **All period 20 neurons**, activations for the hours task.

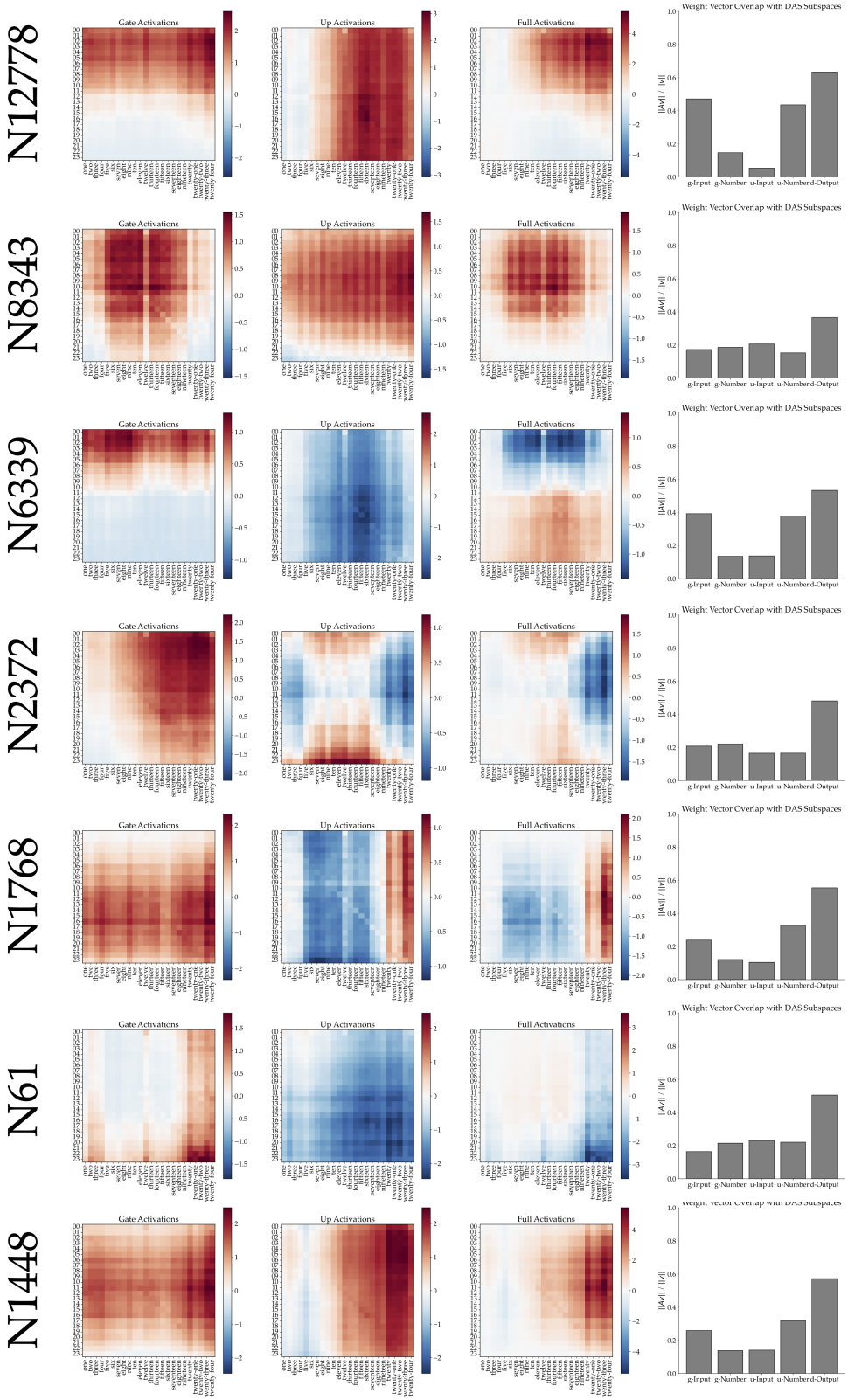


Figure 53: **All period 50 neurons**, activations for the hours task.

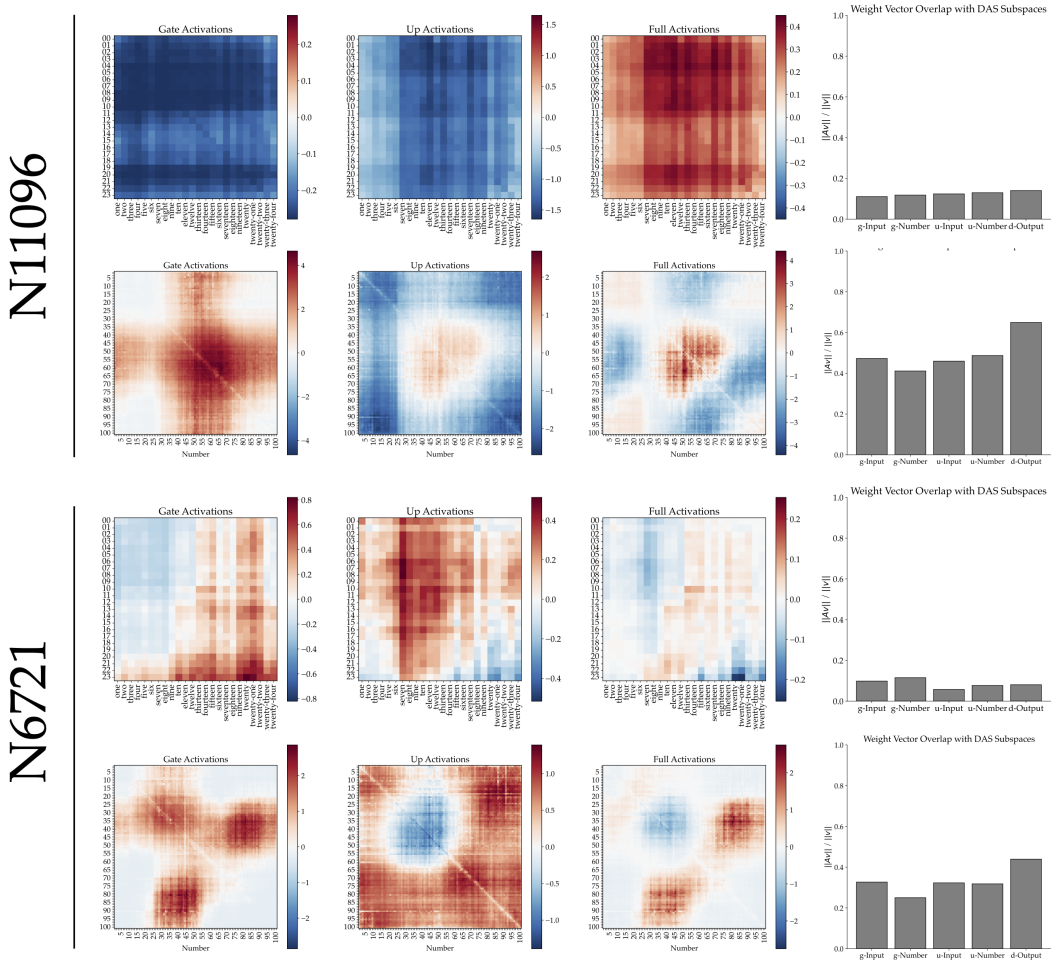


Figure 54: **All period 100 neurons**, activations for the hours task.

J.5 Addition Neuron Down Projection Analysis

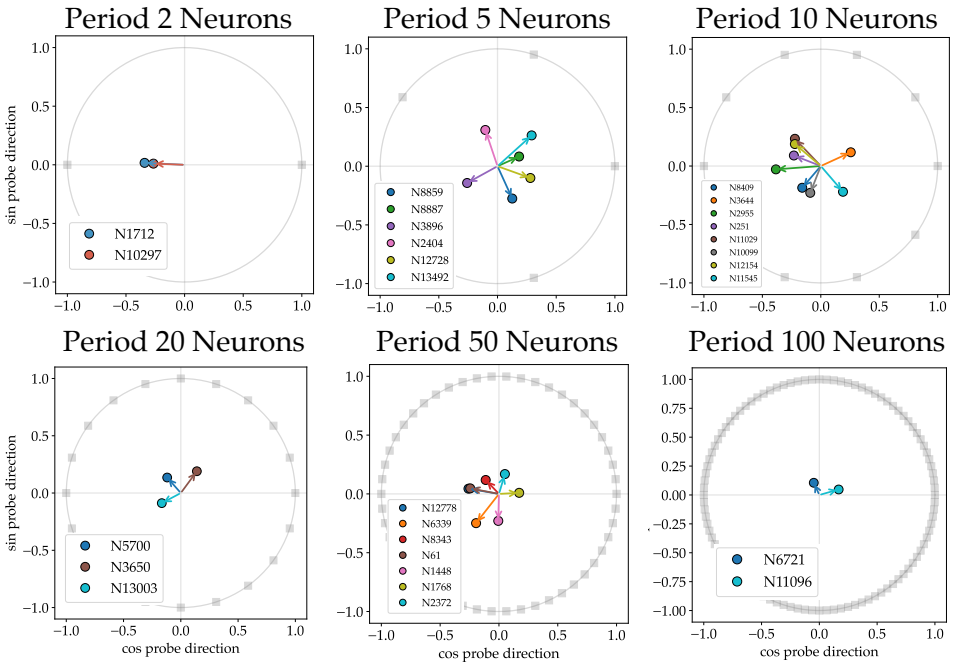


Figure 55: Down projection rows $\mathbf{d}_i$ for all addition neurons, projected onto the Fourier plane that corresponds to their activation period from Figure 43. We normalize probes before projecting. We do not orthogonalize cosine and sine probes for each period, as most are orthogonal (see Figure 40.

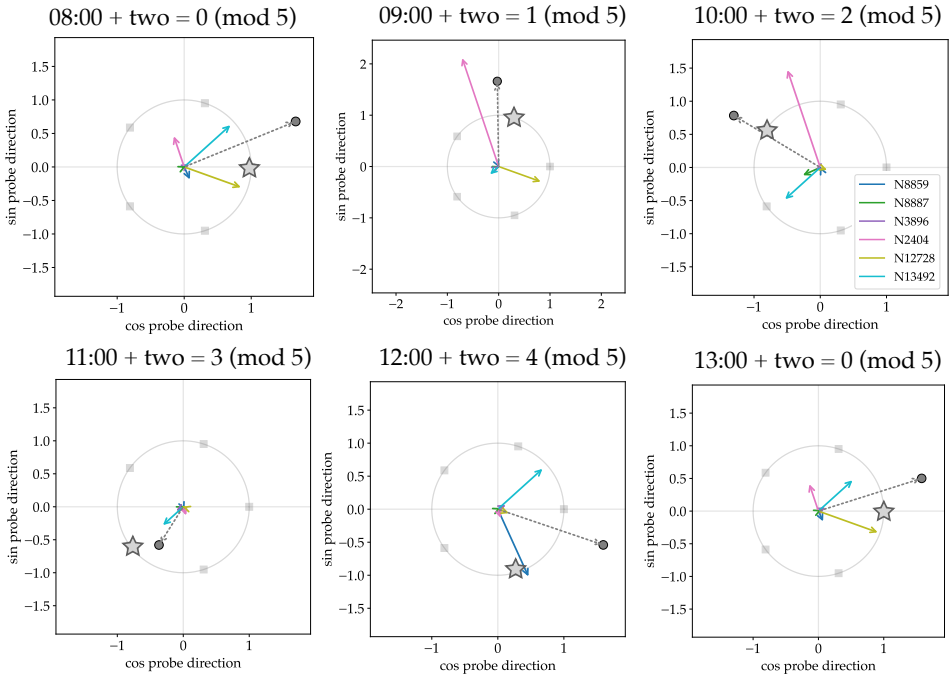


Figure 56: Period 5 neuron behavior when scrubbing across the input concept for the `hours` task, holding offset constant. The gray dashed line indicates the sum of all the period five neuron activations, and the gray star indicates the ideal location this vector should point to. From left to right, the sum of our six neurons creates a vector with an angle roughly corresponding to the sum of input concept + offset modulo 5.

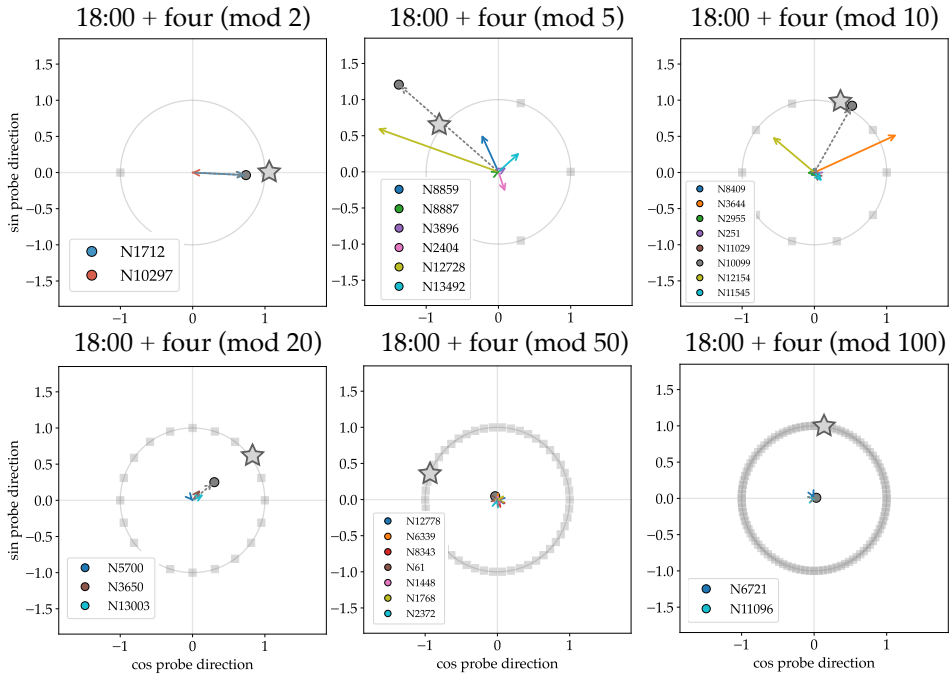


Figure 57: Neuron outputs for the `hours` prompt *four hours after 18:00 = 22:00* projected onto the Fourier plane for each period. $T \in [2, 5, 10, 20]$ activate strongly in the correct directions, while $T \in [50, 100]$ activations are close to zero for this prompt.

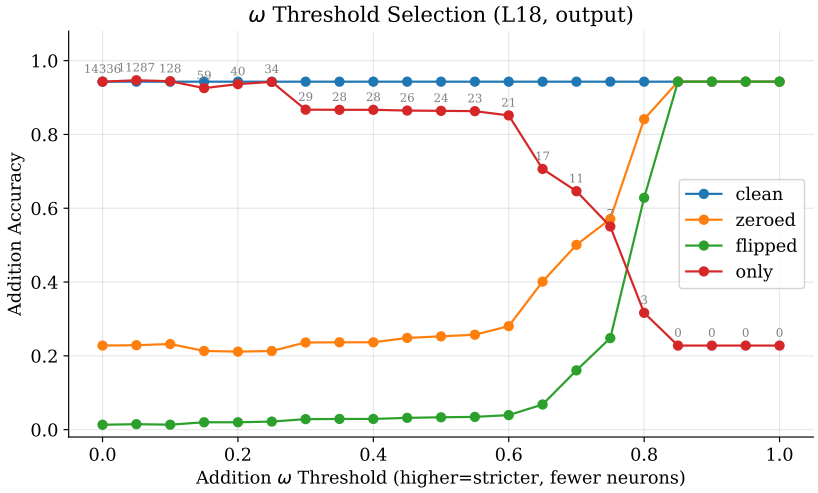


Figure 58: Sweeping over ω for the `addition` task. While we choose $\omega > 0.4$, keeping more neurons ($\omega > 0.2$) allows for cleaner results with still only 40 addition neurons. Gray text indicates the number of neurons included for this threshold. We include analysis on the additional neurons not in $\mathcal{N}_{add}$ in App. J.6.

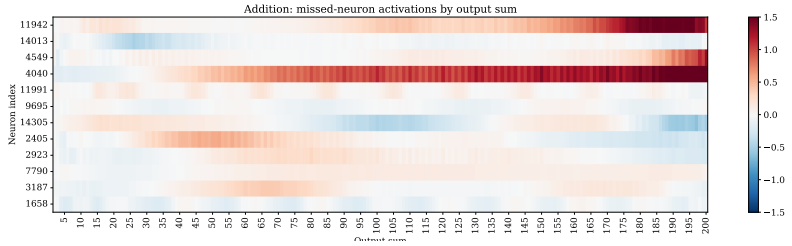


Figure 59: Activations for twelve extra neurons that are included when $\omega > 0.2$ for the `addition` task. These neurons also show wavelike patterns when activations are averaged across output sums.

J.6 Extra Addition Neurons

After sweeping over possible thresholds for ω for the `addition` task in Figure 58, we notice that cleaner ablation results can be achieved with a more lax threshold, including layer 18 MLP neurons with $\omega > 0.2$.

We plot the activation patterns for the additional twelve neurons (40 - 28) not yet analyzed in this paper in Figure 59, observing that they also have periodic patterns, with activations oscillating along output sums. In fact, neurons 11991 and 1658 appear to be missing period 20 neurons, while others appear to encode larger periods, as hypothesized in Figure 42 (where we observed that errors with our original set of 28 neurons were mostly for larger numbers).

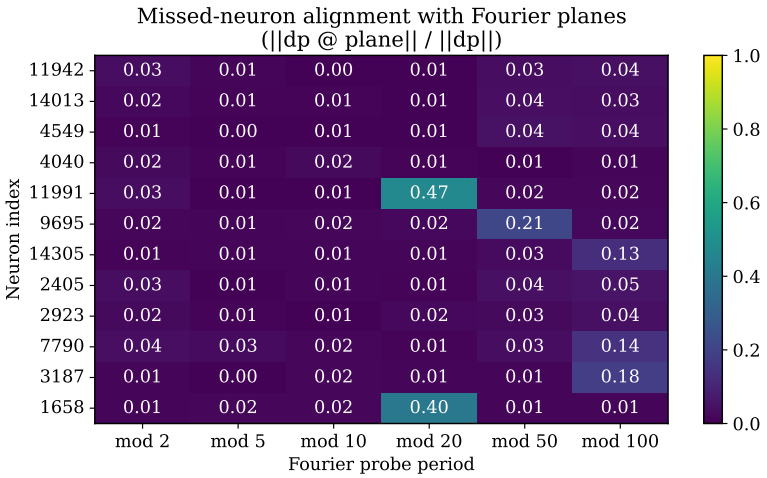


Figure 60: Alignment of twelve extra neurons from Figure 59 that are included when $\omega > 0.2$ for the `addition` task. Two neurons (11991, 1658) appear to be period 20 neurons that could be included in $\mathcal{N}_{add}$. Neuron 9695 appears aligned with $T = 50$, and neurons 14305, 7790, and 3187 are slightly aligned with $T = 100$. For other neurons, we see low alignment—we hypothesize that this is because these neurons write to larger periods. We can directly observe this in Figure 59: e.g., neuron 2405 is positive at 50 and negative at 200, suggesting that it has a period of $T = 300$, and neurons 11942 and 4040 appear to gradually activate for a large range of sums. Also see Figure 42.