WHEN MODELS DON'T MANIPULATE MANIFOLDS: THE GEOMETRY OF A COMPARISON TASK

Anonymous authors
Paper under double-blind review

ABSTRACT

One of the current premises of mechanistic interpretability research is that detailed accounts of the geometry of neural network representations can tell us how models perform computations, and how to effectively intervene on them. While low dimensional manifolds have been observed for multiple concepts in the literature (e.g. numbers encoded on helices, days of the week on a circle, ...), with structure believed to reflect properties of data and tasks, the extent to which models rely on them for computation, and how they manipulate them, remains unclear. We characterize precisely the geometry of computation in a number-comparison task, as an abstraction of comparison for decision making, and how models utilize geometry in an elegant fashion to implement it. Specifically, we study the causal geometry of number comparison in Qwen2.5-7B-Instruct, a capable and widely studied open-weight model, and find Qwen largely uses linear representations of numbers despite the presence of curved geometry. To compare two numbers, the model first encodes each number along a vector and adds the two representations using attention and the residual connection, bringing them into a shared space in the residual stream. Then, the model uses MLP neurons to compare the pair of numbers on local regions in this shared space, which correspond to smaller intervals of input numbers, and combines these to obtain the position of the maximum. In fact, this reliance on linear representations for comparison also persists when the model compares three numbers. Our findings demonstrate that the manifold hypothesis can co-exist with linear representations: while concepts that are ordered may have manifold structure in representations, the model may use an underlying linear structure of the concept in certain computations.

1 INTRODUCTION

How do language models represent concepts internally, and how do they use these representations to perform computations? The linear representation hypothesis (Park et al., 2023; Elhage et al., 2021) posited that models represent ordered concepts as one-dimensional subspaces, which found evidence across diverse applications (Zhu et al., 2024; Marks & Tegmark, 2024; Voynov & Babenko, 2020; Tigges et al., 2023; Lee et al., 2024). Recent work has challenged this perspective and found that these concepts live on low dimensional manifolds – a phenomenon known as the manifold hypothesis: examples include number helices (Kantamneni & Tegmark, 2025), curved manifolds for dates/years (Modell et al., 2025), character count manifolds (Gurnee et al., 2026), as well as age and temperature manifolds (Bhalla et al., 2026). Where does this manifold structure come from? Theoretical analyses suggest that the manifold structure arises from symmetries in data (Karkada et al., 2026) or task symmetries (Hwang & Park, 2026). These insights have also revealed how steering along manifolds can control model behavior (Wurgaft et al., 2026). However, given a specific computation, which aspects of the representation structure the model uses to perform the computation remains an open question. Does the model manipulate nonlinear concept manifolds, as Gurnee et al. (2026) show using character count manifolds on a line-breaking task, or does the model use simpler linear representations for certain tasks despite the presence of manifold structure?

Elucidating how models manipulate internal representations to perform computations has become an active area of research. Framed as such, this question amounts to investigating the *algorithmic* level in Marr's levels of analysis (Marr, 2010), which claim that a system can be understood at the computational (abstract, behavioral), algorithmic (variables and how they are manipulated) and imple-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

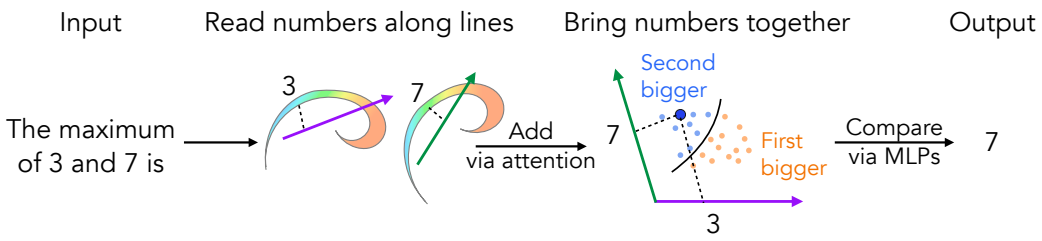

Figure 1: **Linear number representations are used by language model Qwen for comparison**. When asked to compare numbers, the model first represents them on a nonlinear manifold, but uses structure along a single linear direction in computation. It combines information about the two numbers by adding the corresponding linear representations using a single attention head and a residual connection, creating a shared space which enables comparison. On this shared space, the model identifies which number is larger using MLP neurons across two layers, and uses the answer position to read out the answer to the max task.

mentation levels (low-level implementation of the algorithm). *Mechanistic interpretability* research, despite having largely focused on finding circuits of computation, namely the implementation level, has also become interested in the algorithmic level (Geiger et al., 2021) which operates at a higher level of abstraction. Multiple examples of algorithmic understanding of models exist, including trigonometry on circular/ helical structures for addition (Kantamneni & Tegmark, 2025), an addition mechanism using Fourier features for multiple concepts such as week days, months, etc (Feucht et al., 2026), and explaining attention heads with python programs (Hayes et al., 2026). Our work contributes to this burgeoning and exciting line of work on *algorithmic interpretability*.

Motivated by the above lines of inquiry, we ask how language models implement comparison using representation geometry. Comparison is a widely useful operation for intelligent systems, and language models in particular. It is an integral component of weighing options and making decisions, especially using numerical or ordered concepts (which can be represented on a line; see (Gärdenfors, 2000)). For example, queries like *"Which of these apartments is closest to my office?"*, *"Which quarter had the worst sales in this five-year report?"* require performing comparison of the specified objects using the specified attributes. Despite reports of nonlinear, helical representations of numbers (Kantamneni & Tegmark, 2025), the aspects of number representations models use for the purpose of comparison, and how do they manipulate them to implement the operation remains poorly understood.

Previous works have studied how models perform comparison. (Hanna et al., 2023) characterize in great detail a comparison circuit in GPT-2. However, they do not establish causal role of representation geometry in comparison by acknowledging that "GPT2's structured number representations may be relevant to its greater-than ability. However, our experiments struggle to prove this causally" (quoted from Hanna et al. (2023)). (El-Shangiti et al., 2025) found a linear subspace which causally affects model outputs, but their analysis does not concern the mechanism by which the model compares values represented in this subspace. (Yuchi et al., 2026) study mixed notation number comparison and compare behavioral accuracy with classifier performance. Taken together, these works have focused on circuit discovery, model performance, or probing accuracy: they however do not study causal representation geometry and how it is used by the model to perform comparison. We provide a detailed geometric account of the algorithm language models employ to compare numbers (Fig. 1). In addition to being descriptive, our account is causal: we can predict how causal interventions will affect model behavior.

Concretely, we make the following contributions:

- **Reconciling linear representations and manifolds**: We demonstrate how linear representations of numbers are causally involved in the model's number comparison, despite the presence of underlying curved manifold structure.
- **Algorithm for number comparison using linear representations**: We further show how the model manipulates number representations to perform comparison: by additive mixing of individual number directions, followed by local comparisons (for specific intervals of input numbers) which are then combined to give the global comparison answer.
- **Extension of the comparison algorithm to longer sequences of numbers**: We show how the algorithm using linear number representations extends to three-length sequences.

Algorithm 1 Pairwise Number Comparisons

Require: numbers y_1, y_2 at times $t = 1, 2$ resp.
  if $t = 1$ **then**
    store y_1 as $\boldsymbol{x}_l^1 = \mathbf{u} f_1'(y_1)$ ▷ Fig. 2 (a, b)
  else if $t = 2$ **then**
    store y_2 as $\boldsymbol{x}_l^2 = \boldsymbol{v}_2 f_2(y_2)$ ▷ Fig. 2 (c)
    copy $\boldsymbol{x}_l^1$, transform as $\boldsymbol{v}_1 f_1(y_1)$, ▷ Done by Attention Head(s), Fig. 2(b, d)
    add to residual $\boldsymbol{x}_{l+1}^2 = \boldsymbol{x}_l^2 + \boldsymbol{v}_1 f_1(y_1)$ ▷ Residual Stream, Fig. 3 (a–c)
    divide span$(\boldsymbol{v}_1, \boldsymbol{v}_2)$ into regions $\{\mathcal{R}_i\}$ ▷ MLP neurons, Fig. 4(a, b)
    compare y_1, y_2 in $\{\mathcal{R}_i\}$ as $C_i = \mathbb{I}(y_1 > y_2).\mathbb{I}(\boldsymbol{x}_{l+1}^2 \in \mathcal{R}_i)$ ▷ MLP neuron outputs, Fig. 4(b)
    combine local comparisons $\{C_i\}$ to get $ans = \mathbb{I}(y_1 > y_2)$ ▷ later MLP neurons, Fig. 4(c, d)
  end if

2 Pairwise Comparison of Numbers

In this section, we first describe the pairwise comparison mechanism in an LLM. We state the algorithm explicitly, and discuss the main steps involved. Furthermore, we provide evidence describing how the model implements this algorithm in subsequent subsections. We extend the algorithm to three number comparisons and include evidence in Section 3.

Notation. Computationally relevant subspaces within model activations are denoted by $\boldsymbol{x}_l^t$, where l denotes the layer index within the model and t denotes time (token position). Numbers present in the input prompt are denoted by $\{y_t\}$. $\{\boldsymbol{v}_i\}$ are directions in model activations, which belong to the same space (same layer and token position). These directions encode numbers, with $\boldsymbol{v}_i$ encoding $\boldsymbol{y}_i$ as $\boldsymbol{v}_i f_i(y_i)$, where f_i may be a nonlinear function of y_i.

2.1 Algorithm for Pairwise Comparisons

The model encodes each number's magnitude y_i as a nonlinear function $f_i(y_i)$ along a single direction $\boldsymbol{v}_i$, which is different for each number position. Since the two numbers y_1, y_2 are provided as inputs at different times (distinct token positions), the model then creates a *shared* representation from the two numbers by (1) copying information about y_1 into y_2's position, and (2) adding together the single number representations. The shared representation is expressed as:

$$\boldsymbol{x} = \boldsymbol{v}_1 f_1(y_1) + \boldsymbol{v}_2 f_2(y_2) \tag{1}$$

This representation spans a two-dimensional plane $\text{span}(\boldsymbol{v}_1, \boldsymbol{v}_2)$. Moving along certain directions in this plane (e.g., along $\alpha\boldsymbol{v}_1 - \beta\boldsymbol{v}_2$ for any $\alpha > 0, \beta \geq 0$) changes the probability of the model answering y_1 as the greater number. The comparison $y_1 > y_2$ can be linearly decoded on this plane. However, the model implements comparison in two stages, as described below.

From this shared representation, MLP neurons first perform local comparisons, identifying and comparing the two numbers y_1, y_2 for specific ranges of individual numbers or their combinations. This occurs because of the gating-based nonlinearity of MLP neurons (SwiGLU for Qwen2.5-7B), whose sigmoid gate and overall expression leads to local regions of activation on intersection with span$(\boldsymbol{v}_1, \boldsymbol{v}_2)$. The outputs of the neurons are nonlinear on local regions, a consequence of approximate quadratic behavior of the SwiGLU nonlinearity on active regions.

Subsequent MLP neurons then combine these local comparisons to create global comparator neurons, which nearly perfectly capture $\mathbb{I}(y_1 > y_2)$. These neurons then construct a single direction in the residual stream which encodes the comparison's answer, and causally affects the model's outputs.

The algorithm is stated in Alg. 1, visualized in App. Fig. 9, along with evidence demonstrating each step.

Evidence for the pairwise comparison algorithm. We perform experiments using the open-weight model Qwen2.5-7B-Instruct (Yang et al., 2024). We ask the model to compare pairs of numbers, and provide the model with a one-shot example for output format. The prompt is "*Answer in the*

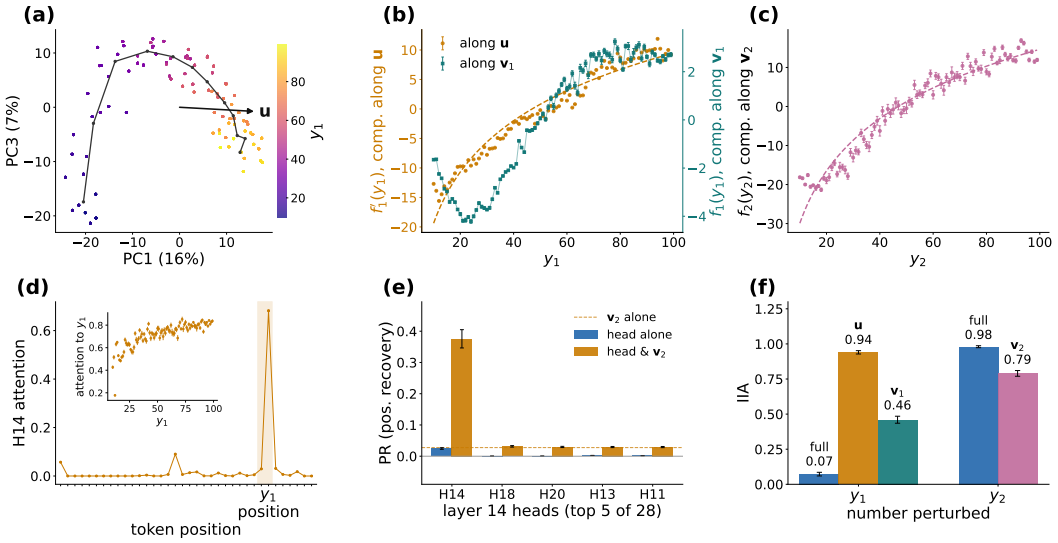

Figure 2: **A single causal direction for numbers controls model comparison. (a)** A causal direction $\mathbf{u}$ found in Layer 13's residual stream causally affects model behavior, despite the presence of curved geometry in the activations, as observed in principal components 1, 3. **(b, c)** Projection of activations onto the obtained causal directions $\mathbf{u}, \boldsymbol{v}_1, \boldsymbol{v}_2$ encode the number magnitude for a wide range of values. While $\mathbf{u}$ is in layer 13 residual stream at the first number y_1's position, $\boldsymbol{v}_1, \boldsymbol{v}_2$ are causal directions encoding y_1, y_2 resp. at y_2's position. $\boldsymbol{v}_1$ is obtained using DAS at an attention head H14's outputs. **(d)** Attention head 14 in layer 14 attends to the position of the first number y_1, irrespective of the number value, serving as a copy head. **(e)** Among all the attention heads in Layer 14, head H14 is causally involved in the model's computation, showing significantly higher position recovery than others. **(f)** Patching along $\mathbf{u}, \boldsymbol{v}_1, \boldsymbol{v}_2$ has significant causal effects on model behavior, as shown by interchange intervention accuracy (IIA).

following format with a single answer. The maximum of 12 and 4 is 12. The maximum of y1 and y2 is ". Our causal analyses involve activation patching using interchange interventions, where we patch specific component activations from a model running on a 'clean' prompt to when the model is processing another 'corrupt' prompt ((Meng et al., 2022)). For example, suppose the 'clean' prompt has inputs $(60, 10)$. The corrupt prompt then uses $(3, 10)$ as inputs, and patching activations from clean to corrupt changes the model's outputs on the 'corrupt' prompt. In this example, since the first number is changed between the clean and corrupt prompts, we refer to this as 'y_1 perturbed' in subsequent figures Fig. 2, 3, 4, 5. Further insights into our experimental setup is included in App. A.

The model computes the argmax position and uses that to produce the answer. First, we observe that when patching is successful, patching model activations from a 'clean' run to a 'corrupted' run leads to changing the model's answer position, instead of the value (see App. C.1). For instance, patching activations from $(60, 10)$ into a model processing $(3, 10)$ will make the model answer 3, and not 60. Therefore, internal model activations compute the $\arg\max$ position and use that to produce the answer. We restrict our analysis to the model's computation of the $\arg\max$ position.

We demonstrate the pairwise comparison algorithm by analyzing individual number representations (Fig. 2), which combine to form the shared representation (Fig. 3). MLP neurons then perform the comparison in two stages, first locally and then globally (Fig. 4).

2.2 Individual Number Representations

Setup. Using 2000 ordered pairs of two-digit numbers, which include 1000 unique randomly chosen pairs (a, b) and their reflections (b, a), we collect model activations at all layers and all token positions while processing the input prompt. The pairs a, b are chosen to have distinct leading digits, so that the effect of patching is visible at the model logits since tokens are individual digits (two digit numbers having different first digits are a large fraction of all possible pairs, $\sim 90\%$). We use both PCA and Distributed Alignment Search (DAS) (Geiger et al., 2024) to find causally relevant directions

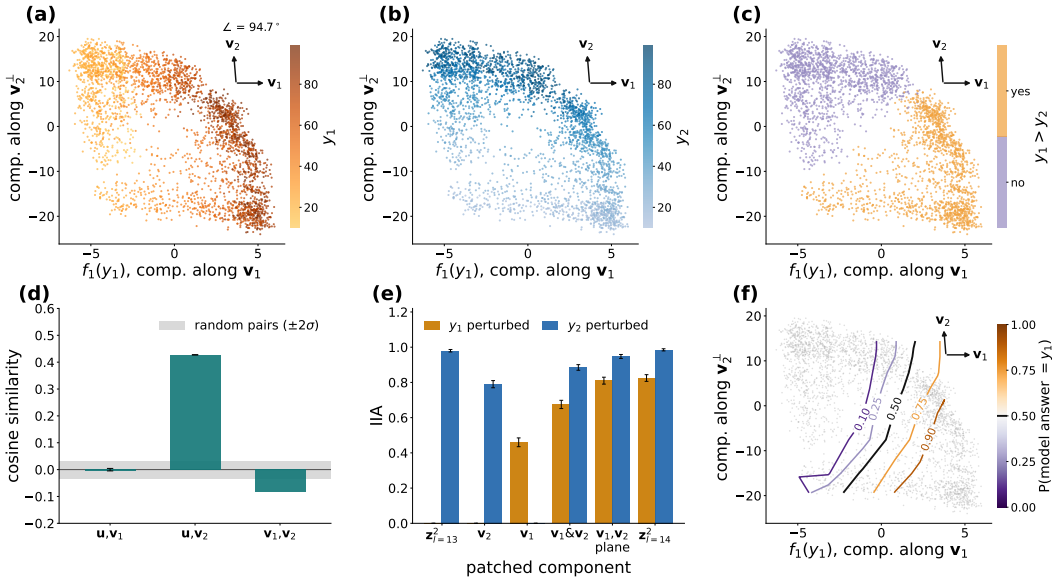

Figure 3: **The model uses causal number directions v_1, v_2 to construct a shared two-dimensional representation encoding the two inputs y_1, y_2. (a)–(c)** The linear span of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ shows each number is encoded along its own direction, and the answer to comparison is linearly separable in this shared representation space. **(d)** By copying y_1 from $\mathbf{u}$ to $\boldsymbol{v}_1$, the model reduces the alignment between the two number representations: $|\cos(\boldsymbol{v}_1, \boldsymbol{v}_2)| << |\cos(\mathbf{u}, \boldsymbol{v}_2)|$. **(e)** While the residual stream at layer 13, denoted $\boldsymbol{z}^2_{l=13}$, encodes y_2, $\boldsymbol{v}_1$ brings in causally useful information into the y_2 position. The shared representation is additive: Patching $\boldsymbol{v}_1 \& \boldsymbol{v}_2$ together (two rank-one patches) nearly matches the IIA of the rank-two patch onto the $\boldsymbol{v}_1, \boldsymbol{v}_2$ plane (i.e., span$(\boldsymbol{v}_1, \boldsymbol{v}_2)$). The plane itself captures as much information as the next layer 14 residual stream. **(f)** Moving around the $\boldsymbol{v}_1, \boldsymbol{v}_2$ plane, which is a two-dimensional plane in the 3,584-dimensional residual stream, is sufficient to change model behavior predictably.

encoding each number y_1, y_2: we employ DAS whenever the principal components are not causally relevant.

Observations. Fig. 2a shows that despite nonlinear manifold structure of number representations (shown in 3D PCA), there exists a direction $\mathbf{u}$ in layer 13's residual stream which causally affects the model's answer, as measured by Interchange Intervention Accuracy (IIA, (Geiger et al., 2021)) (Fig. 2f). Note that we compute IIA using the immediate next token generated by the model. An extension to the model's entire work is left to future work. A single attention head H14 of layer 14 copies information about y_1 from layer 13 in token position y_1 to y_2. It consistently attends to the y_1 position (Fig. 2 d), and patching this head's output has the highest effect on the model's logits, as measured using position recovery (a modified version of recovery (Meng et al., 2022) that patching changes the model's answer position instead of value).

$$PR = \frac{LD_{patch} - LD_{corrupt}}{LD_{clean} - LD_{corrupt}} \tag{2}$$

where LD is the logit difference between the pair $(\min(y_1, y_2), \max(y_1, y_2))$ where y_1, y_2 are inputs on the corrupt prompt, since upon patching from clean to corrupt prompts, models output $\min(y_1, y_2)$ as the answer to the corrupt max-prompt. Note that we call this position recovery to observe how well the model reorganizes its logits to patching and answers with the patched position. The denominator is only meant to provide a rough scale of logit difference. $\boldsymbol{v}_1$ is then obtained as the DAS direction at the output of H14. $\boldsymbol{v}_2$ is obtained as the top principal component from the layer 13 residual stream. Therefore, obtained causal directions $\mathbf{u}, \boldsymbol{v}_1, \boldsymbol{v}_2$ encode the values of y_1, y_1, y_2 respectively, as shown in panels (b, c). These directions are causal, as shown by their IIA scores in (Fig. 2f).

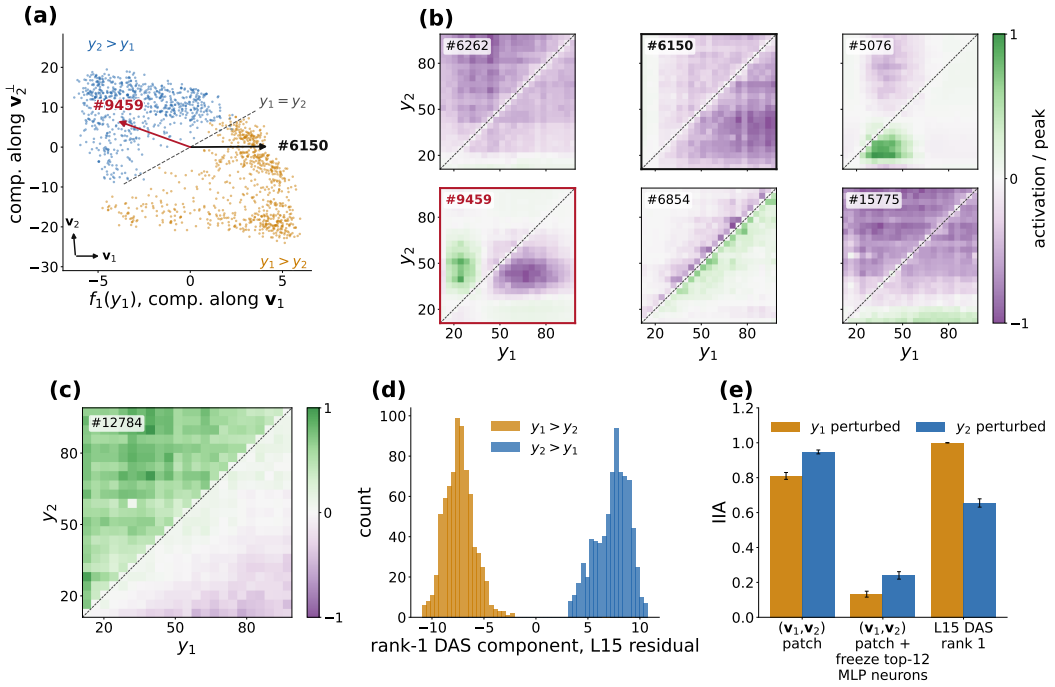

Figure 4: **The model compares numbers y_1, y_2 by combining local comparisons on the shared $\boldsymbol{v}_1, \boldsymbol{v}_2$ plane**. **(a)** Individual neuron weights in layer 14 MLP are specific directions in the $\boldsymbol{v}_1, \boldsymbol{v}_2$ plane. The y-axis is $\boldsymbol{v}_2^\perp$, the component of $\boldsymbol{v}_2$ orthogonal to $\boldsymbol{v}_1$ (since $\boldsymbol{v}_1, \boldsymbol{v}_2$ are not exactly orthogonal) **(b)** Neurons in layer 14 MLP (ranked by attribution scores) localize specific regions of the inputs y_1, y_2 (like neuron $\#6150$), or perform comparisons in localized regions (neuron $\#9459$). **(c)** Select neurons in layer 15 MLP, which combine the outputs of layer 14 MLP neurons, are global comparators: they respond positively when $y_2 > y_1$ and negatively otherwise. **(d)** A single direction in the layer 15 residual stream (after layer 15 MLP), which is formed by inputs from global comparator neurons from layer 15 MLP, encodes the position of the answer $\arg\max(y_1, y_2)$. **(e)** There are 12 comparator neurons (6 each in layer 14, 15 MLPs) which perform comparison: freezing these neurons significantly degrades the IIA achieved by patching in the (v_1, v_2) plane . The one-dimensional comparison direction in layer 15 (panel (d)) controls model behavior.

2.3 Shared Representation Enables Comparison

Using the causal directions of individual numbers $\boldsymbol{v}_1, \boldsymbol{v}_2$, the model creates a shared representation of both numbers, which is a two-dimensional plane in the pre-MLP residual stream of layer 14. We visualize the projection of model activations onto the linear span of these directions (Fig. 3). This shared representation lives in the pre-MLP residual stream of layer 14. Fig. 3 (a)–(c) show activations in this space encoding each number along its own direction, while allowing linear separation of the two cases $y_1 > y_2, y_1 < y_2$. The probability contours of the model's answer are approximately parallel to the decision boundary (Fig. 3 f). The directions $\boldsymbol{v}_1, \boldsymbol{v}_2$ are nearly orthogonal (Fig. 3 d) and less aligned than their counterparts which lived at different token positions $(\mathbf{u}, \boldsymbol{v}_2)$. Patching these directions using two rank-one patches (i.e., by projecting along $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$) shows a similar causal effect (IIA) as patching the rank-two shared representation space (by projecting onto $\text{span}(\boldsymbol{v}_1, \boldsymbol{v}_2)$) (Fig. 3 e), indicating linear combination of individual number representations create this shared space. This observation is nontrivial because $\boldsymbol{v}_2$ is obtained in layer 13's residual stream and $\boldsymbol{v}_1$ is from head H14's output (which itself uses the layer 13 residual stream in computation), making nonlinear interactions possible.

2.4 Global Comparison is a Combination of Local Comparisons

Setup. We first identify which layer MLPs are involved in the maximum computation by performing freezing experiments. Here, we patch the $\text{span}(\boldsymbol{v}_1, \boldsymbol{v}_2)$ space, but freeze the downstream MLP outputs to remain the same as the no-patch case. This allows us to check the contribution of individual layer MLPs in the subsequent computation: we expect that freezing important MLPs will result in a

Algorithm 2 Three-Number Comparisons

Require: numbers y_1, y_2, y_3 at times $t = 1, 2, 3$ resp., T is the last token.
  if $t = 1$ **then**
    store y_1 as $\boldsymbol{x}_l^1 = \mathbf{u}_1 f_1(y_1)$
  else if $t = 2$ **then**
    store y_2 as $\boldsymbol{x}_l^2 = \mathbf{u}_2 f_2(y_2)$
    compute $\alpha_2 = \mathbb{I}(y_2 = \max(y_1, y_2))$ and store along $\mathbf{d}_2$ ▷ Use Alg. 1; Fig. 5 (d), App. Fig. 26
  else if $t = 3$ **then**
    store y_3 as $\boldsymbol{x}_l(3) = \boldsymbol{v}_3 f_3(y_3)$ ▷ Fig. 5(a)
    copy $\boldsymbol{x}_l^1, \boldsymbol{x}_l^2$, transform as $\boldsymbol{v}_1 f_1(y_1), \boldsymbol{v}_2 f_2(y_2)$ ▷ Done by Attention Heads, App. Fig. 26
    add to residual $\boldsymbol{x}_{l+1}(3) = \boldsymbol{x}_l(3) + \boldsymbol{v}_1 f_1(y_1) + \boldsymbol{v}_2 f_2(y_2)$ ▷ Residual Stream, App. Fig. 26
    divide span$(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3)$ into regions $\mathcal{R}_i$ ▷ Fig. 5(c)
    compare in $\mathcal{R}_i$ as $C_i = \mathbb{I}(y_3 = \max(y_1, y_2, y_3)).\mathbb{I}(\boldsymbol{x}_{l+1}(3) \in \mathcal{R}_i)$ ▷ MLP neurons, Fig. 5(c)
    combine $\{C_i\}$, get $\alpha_3 = \mathbb{I}(y_3 = \max(y_1, y_2, y_3))$, store on $\mathbf{d}_3$ ▷ Later MLP neurons, Fig. 5(d)
  else if $t = T$ **then**
    combine $\alpha_2 \mathbf{d}_2, \alpha_3 \mathbf{d}_3$ to get $\boldsymbol{x}_{\ell'}(T) = \sum_t w_t \alpha_t \mathbf{d}_t$ ▷ Attention Head, Fig. 5(e), App. Fig. 26
    Read $\arg\max(y_1, y_2, y_3)$ from $\boldsymbol{x}_{\ell'}(T)$
  end if

significant drop in the patching effectiveness. For relevant MLPs, we then identify neurons of interest using first-order attribution scores from the model logits (App. B.5, App. C.5). The obtained neurons are tested for causal relevance by further patch-and-freeze experiments (see App. B for details).

Observations. MLP neurons in layers 14 and 15 operate on the shared representation space span$(\boldsymbol{v}_1, \boldsymbol{v}_2)$ (Fig. 4(a)). We find a set of 12 neurons (6 neurons in MLP of layer 14 and 6 in layer 15's MLP) which together contribute to the comparison computation, and freezing these neurons significantly reduces the IIA from patching the $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ space (see Fig. 4 (e) and App. C.5). They perform the comparison in two stages: first, neurons in the MLP of layer 14 perform localized comparisons: they respond to specific ranges of values of y_1 or y_2, or a combination thereof (e.g., $|y_1 - y_2| < \eta$) (see Fig. 4 (b) which shows the receptive fields of neurons, i.e., their activations as a heatmap in the (y_1, y_2) space). While the geometry of the span$(\boldsymbol{v}_1, \boldsymbol{v}_2)$ space makes comparison possible by linear readouts (Fig. 3c, MLP neurons seem to use their linear transform followed by nonlinearity to perform local comparisons. Some neurons perform the comparison $y_1 < y_2$ on these bounded ranges (like neuron #9459 in Fig.4(b)) while others serve to identify the ranges (like neuron #6150 in Fig. 4(b)). These neurons are then combined in the second stage (MLP of layer 15), into neurons which serve as global comparators, encoding the answer position over the entire range of values of both numbers (Fig. 4(c)). We find a single direction (using DAS) in layer 15's residual stream (after MLP of layer 15), which is written to by several global comparator neurons, and encodes the answer to the comparison as a binary value (Fig.4(d)). The importance of these twelve MLP neurons, as well as the causal relevance of the DAS direction obtained in layer 15, are shown using IIA in panel (Fig. 4(e)).

3 Multiple Comparisons: The Case with Three Numbers

Having stated and established evidence for the pairwise comparisons algorithm in Qwen, we now extend it to multiple number comparisons: does the model continue to use linear representations of numbers for multiple comparisons? The model is very good at the multiple comparison task: comparing long sequences of numbers in a single forward pass (see App. C.1). Using three numbers as a case study, we state the model's algorithm and describe evidence for this algorithm in model activations.

3.1 Algorithm for Multiple Comparisons

To compare longer sequences of numbers, language models employ an interesting strategy: at every time t, they compute a binary flag which answers the following question: is y_t is the largest number seen so far? To achieve this, they construct an additive sum of linear representations of all numbers up to and including y_t. Individual neurons then perform local comparisons on this

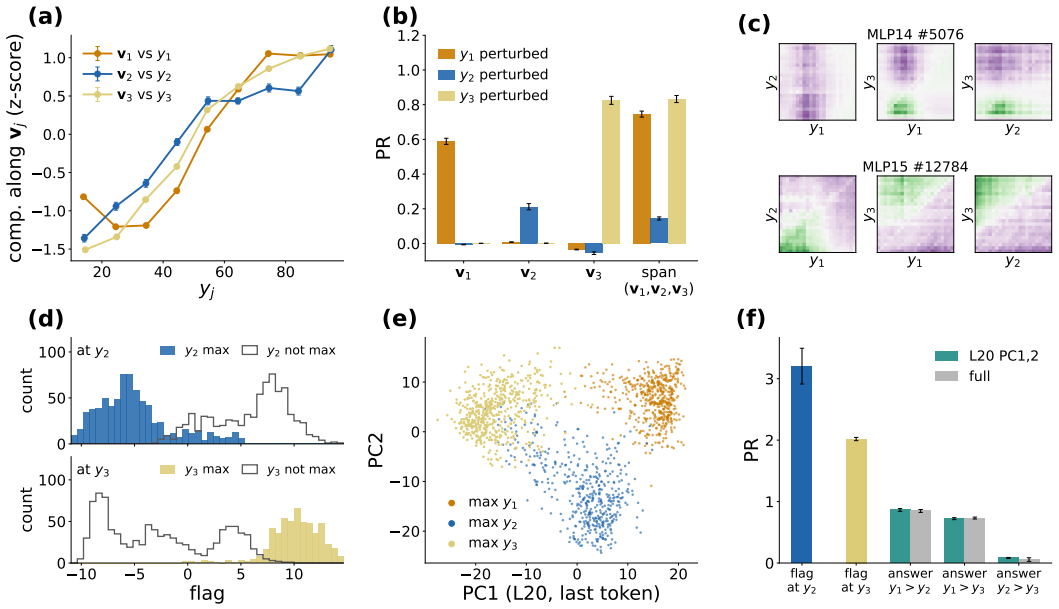

Figure 5: **The model continues to use linear number representations for three-number comparisons.** **(a)** We observe that the causal directions $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ (obtained at the third number y_3's position) encode the numbers y_1, y_2, y_3, through nearly monotonic components $f_i(y_i)$. **(b)** The three directions $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ have causal effects on the model outputs (logits, measured using position recovery, PR). The effect on y_2 is lower at this position (y_3 position). **(c)** Receptive fields from layer 14 MLP neurons show local comparisons now being performed in the y_1, y_2, y_3 space (whose three two-dimensional views are shown). **(d)** The model represents its answer *flag* along a single direction in the layer 15 residual stream at both the y_2 and the y_3 position. The flag identifies if y_t is the largest upto time t. **(e)** The $\arg\max$ position is represented in the top two principal components of layer 20 residual stream, at the last token position (after the third number y_3, and right before the model answers). **(f)**. The flags at y_2 and y_3 positions, as well as the answer position from (e), are causal and affect model logits.

shared representation, which is combined to give a global comparator that identifies whether y_t is the largest entry seen so far. This information is stored along a single direction $\mathbf{d}_t$, as a binary variable $\alpha_t = \mathbb{I}(y_t = \max(y_1, \ldots, y_t))$.

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{v}_1 f_1(y_1) + \boldsymbol{v}_2 f_2(y_2) + \boldsymbol{v}_3 f_3(y_3), \\ \text{at } t, \quad \alpha_t &= \mathbb{I}(y_t = \max(y_1, \ldots, y_t)) \,. \end{aligned} \tag{3}$$

At this stage of computation, if y_t is not the maximum upto time t, i.e., the flag $\alpha_t = 0$, the position of the maximum is not otherwise stored (i.e., the model simply knows that the maximum is not at t, but not if the maximum is at an intermediate position). Information about whether the first number y_1 (which is a special case since there are no numbers before it) is the maximum is also computed at the y_3 position. At the final token before generating the answer, the model combines these directions using a single attention head which attends to the answer position (i.e., w_t is high at $\max(y_1, \ldots, y_T)$)) to obtain $\boldsymbol{x}_{\ell'}(T) = \sum_{t=1}^{T} w_t \alpha_t \mathbf{d}_t$ (App. Fig. 26). This final representation encodes the answer position which is then read off by the model. While our results describe how individual number representations are used to create causally relevant flags, we empirically find the above attention head. We leave an investigation into the complete mechanism of how the head works, and how the model decodes the answer position to future work.

We hypothesize that the model may read this information by exploiting the order of numbers, checking sequentially (in decreasing t) if $\alpha_t = 1$ and identifying the first time this occurs. We state the algorithm in Alg. 2.

3.2 EVIDENCE FOR THE THREE NUMBER COMPARISON ALGORITHM

Setup. We extend our analysis to studying comparison of three numbers. We choose 1500 triples and 400 held-out examples. In this case, IIA does not show enough signal for analysis, so we employ position recovery (Eq. 2) to test for causal effects of activation patching on the model's logits.

Observations. First, we note that when given three numbers, the model seems to perform two parallel computations (see App. C.6). Fig. 5's first row shows the computation occurring at the third number position. Using three directions $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ which encode the magnitudes of the three numbers y_1, y_2, y_3 (panel a), the model creates a three-dimensional space span$(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3)$. Fig. 5(b) shows that patching each direction affects the model outputs when the corresponding number is perturbed, and that patching the span affects multiple numbers. The effect of perturbing y_2 is minimal here, indicating that the model uses different computation for determining y_2 is the largest. Representative MLP neurons in layer 14 MLP perform local comparisons in the (y_1, y_2, y_3) space (Fig. 5(c)). These neurons construct a single direction in layer 15's residual stream space which encodes the answer $\mathbb{I}(y_3 = \max(y_1, y_2, y_3))$ (Fig. 5(d), bottom). At the second number y_2's position, the model compares y_2 and y_1 (using Alg. 1), and creates a single direction from DAS which encodes the answer position in layer 15's residual stream (Fig. 5(d), top). At the last token position, layer 20's residual stream combines the flag information (using an attention head, see App. Fig. 26) encodes the position of the answer, as visible on a two dimensional PCA Fig. 5(e). Position recovery scores in Fig. 5(f) show that the flags at the y_2 and y_3 positions, as well as the two-dimensional subspace encoding the overall answer in layer 20 (from Fig. 5(e)) are causal. The two principal components in layer 20 have the same position recovery as the entire residual stream at that position (Fig. 5(f)).

4 DISCUSSION AND LIMITATIONS

Our work is a concrete illustration of how a language model (Qwen2.5-7B-Instruct in our case) uses linear representations and additive mixing for specific model computations, number comparison in our case. The model uses linear representations for each number position despite the presence of a nonlinear manifold in its activations. We demonstrate how the model methodically combines information from both numbers by adding the number representations together, constructing a two-dimensional plane in its residual stream. The model then employs MLP neurons to focus on specific local regions within this space, which correspond to smaller intervals of the input numbers, and performs comparison on these patches. Finally, the model combines these local patches into a global comparison. In addition, we show how this algorithm extends to three number comparisons. Our findings challenge a central hypothesis in current mechanistic interpretability research, namely that detailed accounts of the representation geometry of individual concepts necessarily make it easier to characterize how models perform meaningful computations using these concepts. As we demonstrate, even though one can precisely characterize the geometry of number manifolds in model activations, simpler linear representations of numbers may be enough to causally affect the model's computation. Our findings complement and add nuance to our understanding of what representation geometry of neural networks truly teaches us: while the intricacies of their geometric structure may reflect the properties of the underlying data distribution, simpler structures within this geometry may be sufficient computationally and useful to the model.

Limitations. The limitations and assumptions of our work are stated below:

- Our analysis is restricted to a specific kind of input – numbers – and a specific operation – computing the maximum. Our claims about linear representations being useful despite the presence of manifold geometry hold for concepts that are ordered, i.e., those that can be mapped to the number line (without periodicity).
- We rely on DAS (Geiger et al., 2024) to obtain causal directions in model activations, whenever principal components have low causal effects. Therefore, our analysis may partially inherit the same challenges as DAS, such as finding shortcuts (Wu et al., 2024). However, we include additional evidence grounded in the model, such as neurons, attention heads, etc which may partially alleviate concerns of shortcuts.
- Much of our analysis shows sufficiency: we identify directions and subspaces which can modify the model's outputs when patched. We don't claim necessity: removing these components may not affect the model's outputs – the model can still use other paths of information processing within its layers to solve the same task (like the hydra effect (McGrath et al., 2023)).

AI USE STATEMENT

In this work, we used generative AI tools for writing and editing code, literature search, drafting portions of the appendix, and feedback on research ideas and experimental methodology. We have reviewed and verified all AI-assisted work. We take responsibility for the final content of this work, including text, claims, code, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix A provides the experimental setup, prompts, datasets and sampling procedures, while Appendix B provides details of the intervention, direction-learning, attribution, and evaluation methods. Sample sizes and hyperparameters are summarized in Table 1.

REFERENCES

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022. doi: 10.1162/coli_a_00422.

Usha Bhalla, Thomas Fel, Can Rager, Sheridan Feucht, Tal Haklay, Daniel Wurgaft, Siddharth Boppana, Matthew Kowal, Vasudev Shyam, Jack Merullo, Atticus Geiger, and Ekdeep Singh Lubana. Do sparse autoencoders capture concept manifolds?, 2026. URL https://arxiv.org/abs/2604.28119.

Lawrence D. Brown, T. Tony Cai, and Anirban DasGupta. Interval estimation for a binomial proportion. *Statistical Science*, 16(2):101–133, 2001. doi: 10.1214/ss/1009213286.

Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing: A method for rigorously testing interpretability hypotheses. AI Alignment Forum, https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing, 2022.

Ahmed Oumar El-Shangiti, Tatsuya Hiraoka, Hilal AlQuabeh, Benjamin Heinzerling, and Kentaro Inui. The geometry of numerical reasoning: Language models compare numeric properties in linear subspaces. 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 (Volume 2: Short Papers)*, pp. 550–561, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-190-2. doi: 10.18653/v1/2025.naacl-short.47. URL https://aclanthology.org/2025.naacl-short.47/.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Sheridan Feucht, Tal Haklay, Usha Bhalla, Daniel Wurgaft, Can Rager, Raphaël Sarfati, Jack Merullo, Thomas McGrath, Owen Lewis, Ekdeep Singh Lubana, et al. Arithmetic in the wild: Llama uses base-10 addition to reason about cyclic concepts. *arXiv preprint arXiv:2605.01148*, 2026.

Peter Gärdenfors. *Conceptual spaces*, volume 3. MIT press Cambridge, MA, 2000.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 9574–9586, 2021. arXiv:2106.02997.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah D. Goodman. Finding alignments between interpretable causal variables and distributed neural representations. In *Proceedings of the Third Conference on Causal Learning and Reasoning (CLeaR)*, volume 236 of *Proceedings of Machine Learning Research*, pp. 160–187. PMLR, 2024. arXiv:2303.02536.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023.

Wes Gurnee and Max Tegmark. Language models represent space and time. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.02207.

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, 2026. URL https://arxiv.org/abs/2601.04480.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36:76033–76060, 2023.

Amiri Hayes, Belinda Z Li, and Jacob Andreas. Explaining attention with program synthesis, 2026. URL https://arxiv.org/abs/2606.19317.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2733–2743, 2019. doi: 10.18653/v1/D19-1275.

Hyeonbin Hwang and Yeachan Park. Intrinsic task symmetry drives generalization in algorithmic tasks. *arXiv preprint arXiv:2603.01968*, 2026.

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. *arXiv preprint arXiv:2602.15029*, 2026.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. AtP*: An efficient and scalable method for localizing LLM behaviour to components. *arXiv preprint arXiv:2403.00745*, 2024.

Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K. Kummerfeld, and Rada Mihalcea. A mechanistic understanding of alignment algorithms: A case study on dpo and toxicity, 2024. URL https://arxiv.org/abs/2401.01967.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets, 2024. URL https://arxiv.org/abs/2310.06824.

David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010.

Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. The hydra effect: Emergent self-repair in language model computations, 2023. URL https://arxiv.org/abs/2307.15771.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 17359–17372, 2022. arXiv:2202.05262.

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.

Neel Nanda. Attribution patching: Activation patching at industrial scale. https://www.neelnanda.io/mechanistic-interpretability/attribution-patching, 2023.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *NeurIPS 2023 Workshop on Attributing Model Behavior at Scale (ATTRIB)*, 2023. arXiv:2310.10348.

Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. Linear representations of sentiment in large language models, 2023. URL https://arxiv.org/abs/2310.15154.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 12388–12401, 2020.

Andrey Voynov and Artem Babenko. Unsupervised discovery of interpretable directions in the gan latent space, 2020. URL https://arxiv.org/abs/2002.03754.

Kevin 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 *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2211.00593.

Zhengxuan Wu, Atticus Geiger, Thomas Icard, Christopher Potts, and Noah D. Goodman. Interpretability at scale: Identifying causal mechanisms in alpaca, 2024. URL https://arxiv.org/abs/2305.08809.

Daniel Wurgaft, Can Rager, Matthew Kowal, Vasudev Shyam, Sheridan Feucht, Usha Bhalla, Tal Haklay, Eric Bigelow, Raphael Sarfati, Thomas McGrath, Owen Lewis, Jack Merullo, Noah Goodman, Thomas Fel, Atticus Geiger, and Ekdeep Singh Lubana. Manifold steering reveals the shared geometry of neural network representation and behavior, 2026. URL https://arxiv.org/abs/2605.05115.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Fengting Yuchi, Li Du, and Jason Eisner. Llms know more about numbers than they can say, 2026. URL https://arxiv.org/abs/2602.07812.

Wentao Zhu, Zhining Zhang, and Yizhou Wang. Language models represent beliefs of self and others, 2024. URL https://arxiv.org/abs/2402.18496.

A EXPERIMENTAL SETUP

A.1 MODEL, TASK AND PROMPTS

We use Qwen2.5-7B-Instruct (Yang et al., 2024) (28 layers, $d_{\text{model}} = 3584$, 28 heads of dimension 128, SwiGLU MLPs of width $d_{\text{ff}} = 18944$) in `float16` with all parameters frozen, on a single NVIDIA A100 40GB GPU. The task $\max(y_1, \ldots, y_K)$ is posed as a raw completion (no chat template) with a one-shot exemplar that fixes the answer format:

> `Answer in the following format with a single answer. The maximum of 12 and 4 is 12. The maximum of` $\{y_1\}$ `and` $\{y_2\}$ `is`

for $K = 2$ in both digit regimes, with the exemplar "`The maximum of 12, 437 and 5 is 437.`" and a question "`The maximum of` $\{y_1\}$`,` $\{y_2\}$ `and` $\{y_3\}$ `is`" for $K \geq 3$; each prompt ends in a single space.

Qwen2.5 tokenizes numbers digit by digit, so an operand's first token is its leading digit. We read every residual-stream quantity at a number's last token, its *position*, where the whole number is first visible under causal attention; all operands are first visible at the position of y_2 ($K = 2$) or y_3 ($K = 3$). Positions are found from the operands' character spans after the last occurrence of "`The maximum of `" via the tokenizer's offset mapping. Within an experiment all operands have the same number of digits, so every prompt has the same token layout ($T = 36$ with y_1, y_2 at tokens $29, 33$ for two 2-digit operands; $T = 38$ with tokens $30, 35$ for two 3-digit operands; $T = 46$ with tokens $35, 39, 43$ for three 2-digit operands). All sampled operands have pairwise-distinct leading digits, so that each readout token identifies one operand.

Two-digit operands lie in $[10, 100)$ with a minimum gap $g = 10$ between the values of a sampled tuple, and three-digit operands in $[100, 1000)$ with $g = 100$; the three-digit regime is used only to replicate the number-representation analysis.

A.2 COUNTERFACTUAL PAIRS

For $K = 2$ we sample triples (a, b, r) which satisfy $a > b + g > r + 2g$ (g is the gap). The clean prompt (e.g., (a, b)) has a at the winning operand and b at the other (answer is a), and the corrupted prompt replaces a in place by r (so corrupted prompt has (r, b)) (answer b), so that only which operand wins changes. The same triples are used in both arrangements (only the order of values is changed), $y_1 > y_2$ and $y_2 > y_1$. Patching a clean state into the corrupted run makes the model name r (smaller value in its input pair) rather than a (Section C.1): the patch changes which operand the model treats as the maximum, not the value it outputs. We therefore score interventions on $t(r)$ against $t(b)$, where $t(\cdot)$ is a number's first token; unlike $t(a)$, $t(r)$ cannot be raised by a patch that merely re-inserts the clean value.

For $K = 3$ we sample quadruples $a > b > c > r$ with consecutive gaps larger than g, place a, b, c in a chosen value order in the clean prompt, and replace a in place by r in the corrupted one, so that the corrupted winner is always the clean runner-up. The *head groups* $y_1 > y_2 > y_3$ (patched at the position of y_2), $y_1 > y_3 > y_2$ and $y_2 > y_3 > y_1$ (both patched at the position of y_3) vary the runner-up's position and the maximum's position one at a time and are used to identify heads. All other analyses use three *perturbation cases*, patched at the position of y_3 and named by the number the corruption changes: case y_1 ($y_1 > y_3 > y_2$), case y_2 ($y_2 > y_3 > y_1$) and case y_3 ($y_3 > y_2 > y_1$).

A.3 POPULATIONS AND SPLITS

Intervention scores are computed on 400 evaluation examples, and directions and neuron rankings are fitted on 128 disjoint fitting examples, both taken from one deduplicated pool drawn by a deterministic sampler, so that every analysis uses the same examples. For $K = 3$ we draw $1{,}400$ quadruples, keep the $1{,}341$ whose clean and corrupted prompts are solved (next-token argmax $t(a)$ and $t(b)$) in all four value orders, following Meng et al. (2022), and use the first 528. For $K = 2$ no filter is applied: in the two-digit regime the model answers every ordered pair of distinct two-digit numbers correctly, while the three-digit populations include one example per arrangement with a non-positive clean-minus-corrupted gap. Geometry and probes use separate clouds of clean prompts, with the two-number clouds containing each unordered pair in both orders.

Behavioural accuracy is measured by greedy decoding of one token more than the operands' digit count, comparing the first integer of the output with the true maximum, on all $8{,}010$ ordered pairs of distinct two-digit numbers and on $4{,}000$ three-digit pairs drawn with replacement. The operand-count sweep uses 200 prompts of distinct two-digit operands per length in the three-operand prompt format, with Wilson score intervals (Brown et al., 2001). Table 1 lists all sample sizes and hyperparameters.

Table 1: Sample sizes and hyperparameters.

	quantity	value
populations	evaluation / fitting examples	400 / 128
	clouds (L13 two-number / L14 / three-number)	2000 / 3000 / 1500
	operand counts k in the accuracy sweeps, prompts per k	$2, 3, 4, 5, 10, 20$; 200
	base random seed	52
learned directions	Adam steps, learning rate	100, 0.05
	minibatch (gradients accumulated)	16–32
	initializations in the stability check ($\boldsymbol{u}$)	2
attribution	minibatch, half-precision loss scale	16, 100
	candidate neurons (MLP14 $\cup$ MLP15)	37,888
	freeze-sweep depths k	$1, 3, 10, 30, 10^2, 3{\cdot}10^2, 10^3, 3{\cdot}10^3$
sweeps and fields	dose-response grid values, window	$15, 25, \ldots, 95$; ± 2
	prompts per dose-response cell (K=2)	64
	receptive-field grid	stride 4 over $y \in [11, 99]$
probes	ridge penalty grid (leave-one-out)	13 log-spaced values in $[10^{-2}, 10^{4}]$
	held-out fraction	0.2

B Methods

B.1 Interventions and sites

Causal measurements are interchange interventions (Geiger et al., 2021; 2024): we run the corrupted prompt and replace a component of the activation at one site by its value on the clean run. With $\boldsymbol{x}$ the corrupted activation, $\boldsymbol{x}^{\text{clean}}$ the clean one and $P \in \mathbb{R}^{d\times k}$ an orthonormal basis,

$$\boldsymbol{x}' \;=\; \boldsymbol{x} \;+\; PP^{\top}\big(\boldsymbol{x}^{\text{clean}} - \boldsymbol{x}\big)\,, \tag{4}$$

computed in single precision. $P = I$ is the full-rank restoration of Meng et al. (2022). Each rank-restricted patch is shown next to a full-rank patch at the same site as a reference rather than an upper bound: rank-restricted patches can exceed it.

The sites are: (i) the residual stream leaving a decoder block, at one position; (ii) one attention head's contribution $\boldsymbol{z}_h(W_O^h)^{\top}$, obtained from the head's 128-dimensional slice of the output-projection input, where a direction P is patched by reading the coefficient $(\boldsymbol{z}_h^{\text{clean}} - \boldsymbol{z}_h)(W_O^h)^{\top}P$ from the head's contribution and adding it along P to the residual stream (P is not constrained to the head's output space; for $\boldsymbol{v}_1$ in L14.H14, 81% of its norm lies inside it); (iii) the pre-MLP residual $\boldsymbol{x}_{\ell-1} + \mathrm{Attn}_{\ell}(\boldsymbol{x}_{\ell-1})$, patched through the attention output, which gives the same result as an end-of-layer patch at full rank; and (iv) MLP neurons, the coordinates of the post-SwiGLU vector that feeds the down-projection.

B.2 Metrics

With $\mathrm{PLD} = \mathrm{logit}[t(r)] - \mathrm{logit}[t(b)]$ computed at the last token, using the same token pair on the clean, corrupted and patched runs, *position recovery* is

$$\mathrm{PR} \;=\; \frac{\mathrm{PLD}_{\text{patched}} - \mathrm{PLD}_{\text{corr}}}{\mathrm{PLD}_{\text{clean}} - \mathrm{PLD}_{\text{corr}}}, \tag{5}$$

the normalized restoration of Meng et al. (2022) applied to the logit difference of Wang et al. (2023), computed per example and averaged. The numerator is the indirect effect of the patched component

on the PLD (Vig et al., 2020), and the denominator is a per-example logit scale of 10–13 logits on average, which makes values comparable across sites and cases but is not meant to make a full effect equal 1: since $t(r)$ is not an answer on the clean prompt, $\mathrm{PLD}_{\text{clean}}$ is can be negative or above 1. Where IIA is near zero, as in three-number cases y_1 and y_2, PR measures how far the PLD moves rather than a change of answer.

Interchange intervention accuracy (Geiger et al., 2021) is $\mathrm{IIA} = \mathbb{I}\left[\arg\max_{v \in \mathcal{V}} \mathrm{logit}[v] = t(r)\right]$ over the full vocabulary at the last token, which scores the first generated token, the leading digit of r, so it also counts the mixed answers produced by a full-rank patch at the last token of y_2 (Section C.1). For the ablations we use *full-number accuracy*: the answer is decoded greedily for one token more than the operands' digit count and counted correct when its leading characters match the true maximum.

B.3 Directions

Learned directions are rank-k subspaces found by distributed alignment search for a single causal variable (Geiger et al., 2024): the basis $Q = \mathrm{qr}(R)$ (using QR factorization) of a free matrix R is inserted into Equation 4 at the chosen site and trained on the fitting examples with a cross-entropy toward $t(r)$ restricted to $\{t(b), t(r)\}$, with gradients accumulated so that each step uses the full-batch mean. With 16 fitting examples a fit reached a training IIA of 1.000 against a held-out 0.688, hence the 128 used here. Retraining $\boldsymbol{u}$ from a second initialization gives $|\cos| = 0.99$ and the same held-out IIA (0.94) for two-digit operands, and $|\cos| = 0.82$ with IIAs 0.86 and 0.80 for three-digit operands; other directions use a single initialization. Where a site may carry one variable shared across cases, a subspace fitted on each case is evaluated on every case. Directions are signed so that their component rises with the number they carry. Table 2 lists the directions.

Table 2: Directions, where they live and how they are obtained. DAS directions are rank one and fitted in the listed case; PC denotes the top principal component(s) of the corresponding clean cloud.

K	direction	position	site	obtained by
2	$\boldsymbol{u}$	y_1	L13 residual	DAS, y_1 perturbed
2	$\boldsymbol{v}_1$	y_2	output of L14.H14	DAS, y_1 perturbed
2	$\boldsymbol{v}_2$	y_2	L13 residual	PC
2	comparison flag	y_2	L15 residual	DAS, each case
3	$\boldsymbol{u}_1$, $\boldsymbol{u}_2$	y_1, y_2	L13 residual	DAS, cases y_1, y_2
3	$\boldsymbol{v}_1$, $\boldsymbol{v}_2$	y_3	outputs of L14.H14, L14.H18	DAS, cases y_1, y_2
3	$\boldsymbol{v}_3$	y_3	L13 residual	PC
3	$\boldsymbol{w}_1$, $\boldsymbol{w}_2$	y_2	L13 residual	DAS, cases y_1, y_2 ($\boldsymbol{w}_2 = \boldsymbol{u}_2$)
3	comparison flags	y_2, y_3	L15 residual	DAS, each case
3	answer subspace	last token	L20 residual	top two PCs

$\boldsymbol{v}_2$ is a principal component because a direction learned at that site partly encodes the comparison outcome instead of y_2 (Section C.3), and heads L14.H14 and L14.H18 come from our earlier localization of the three-number task. Multi-dimensional patches use the span of the $\boldsymbol{v}_j$, orthogonalized by Gram–Schmidt with $\boldsymbol{v}_1$ first; the directions are nearly orthogonal to begin with ($\cos(\boldsymbol{v}_1, \boldsymbol{v}_2) = -0.08$ for two numbers), and because they are obtained independently, the span's score is a lower bound for a rank-k patch at that site.

Principal components are fitted on mean-centred, unscaled activations (Gurnee & Tegmark, 2024). Linear decodability is measured with ridge regression (Alain & Bengio, 2016; Belinkov, 2022) on the logarithm of each number, and the comparison $y_1 > y_2$ with a ridge classifier, reporting held-out R^2 or accuracy; probes for operands that a position cannot depend on serve as floors, in the spirit of the control tasks of Hewitt & Liang (2019). Log-linear trends are summarized by the least-squares fit $p \log y + q$.

B.4 Localization

The causal trace (Meng et al., 2022) restores the clean residual stream leaving one layer at one token, for every layer and every token from the first token of y_1 to the last token of the prompt, and reports position recovery averaged within each arrangement. Heads are then interchanged at site (ii) of layer 14 (Wang et al., 2023; Goldowsky-Dill et al., 2023), each together with a restoration of the layer-13

residual at the same position (in full for three numbers, along $\boldsymbol{v}_2$ for two numbers), since a single head written into a corrupted residual stream has almost no effect; the co-patch alone is shown as the baseline, and for two numbers the sweep is also shown without it. At the last token, full-rank patches of the residual leaving layers 18 to 22 over the head-group orders locate the read-out at layer 20, where mean position recovery increases most (0.09 to 0.55). At that layer we patch each head's contribution without a co-patch, with the whole attention sublayer as a reference, and compare a patch of the top two principal components of the residual with a full-rank patch.

B.5 Neurons

We rank the neurons of MLP14 and MLP15 by attribution patching (Nanda, 2023; Syed et al., 2023; Kramár et al., 2024), the first-order estimate of the effect on the PLD of freezing neuron i while the shared representation is patched, $s_i \approx (a_i^{\text{corr}} - a_i^{\text{patched}})\, \partial \text{PLD}/\partial a_i$, with the gradient taken at the patched state through a zero-valued differentiable input at the down-projection. Rankings are computed on the fitting examples and verified by freezing the top k neurons on the evaluation examples, against two random draws of k neurons. The shared set is the intersection of the top-20 rankings of the two two-number cases, 12 neurons (six in each MLP), which the three-number analysis reuses unchanged.

To freeze a sub-block or a set of neurons, we pin its output (for neurons, their post-SwiGLU values) at the target position to its value on the unpatched corrupted run, the severed-path variant of causal tracing (Meng et al., 2022; Vig et al., 2020), for the attention and MLP sub-blocks of layers 14 to 16 individually and for MLP14 and MLP15 jointly. For injection we write only the clean outputs of the selected MLPs into the corrupted run at the target position. For ablation we set the shared neurons' post-SwiGLU activations to zero, at every position or only at the last token of the final operand, throughout greedy decoding, and compare with two random sets of 12 neurons zeroed at every position, scoring full-number accuracy.

B.6 Dose-response sweeps, receptive fields and connectivity

For the dose-response sweep (Geiger et al., 2021; Chan et al., 2022) we set the two coordinates of the $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ frame in the pre-MLP layer-14 residual at the position of y_2 to the mean coordinate of a chosen number (over a ± 2 window in the cloud), on a 9×9 grid with one value per leading digit, and score the fraction of argmax answers that name the first operand among those naming either operand, masking the diagonal. The pre-MLP site is used because a per-number mean explains $R^2 = 0.87$ of the $\boldsymbol{v}_1$ coordinate there, against 0.35 at the end of the layer.

A neuron's receptive field is its post-SwiGLU activation at the target position over a grid of prompts, one forward pass per cell and without the distinct-leading-digit constraint. Three-number fields are shown as pairwise marginals, each averaged over the third number and divided by its own peak, and an operand that a position cannot depend on is held fixed ($y_3 = 55$ at the position of y_2).

The causal edge from an MLP14 neuron to an MLP15 neuron is the shift in the MLP15 neuron's activation when only the MLP14 neuron is set to its clean value, in units of the MLP15 neuron's standard deviation over a $1{,}500$-prompt two-number cloud. The virtual weight (Elhage et al., 2021) through the gate projection is the cosine between the upstream write column and the downstream gate read row with the RMSNorm scale folded in, and the alignment of a write direction $\boldsymbol{w}_j$ with a learned subspace Q is $\rho_j = \|Q^\top \boldsymbol{w}_j\| / \|\boldsymbol{w}_j\|$, read against the distribution over all $18{,}944$ neurons of the layer, since write directions are not isotropic (the analytic level for a random unit vector and a k-dimensional subspace is about $\sqrt{k/d_{\text{model}}}$). As a specificity control, each three-number direction is re-evaluated, without refitting, in the cases where the number it carries does not change, where it should have no effect.

C Further Experimental Results

The figures below follow the order of the main-text algorithm. Unless a caption says otherwise, scores are means over the 400 held-out counterfactuals of Section A.3, error bars are one standard error, and PR (position recovery) and IIA (interchange intervention accuracy) are as defined in Section B.2.

C.1 Behaviour and what an intervention changes

We first report task accuracy and the answers the model generates under two interventions.

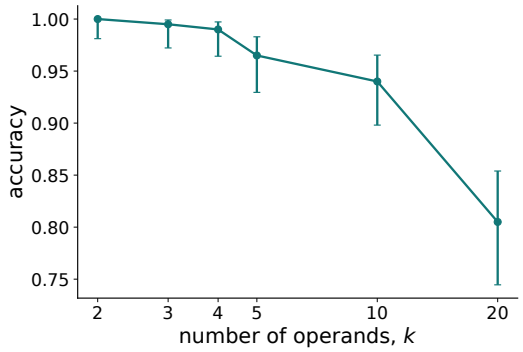

Figure 6: Accuracy on $\max(y_1, \ldots, y_k)$ against the number of operands k, under greedy decoding, for 200 prompts of distinct two-digit operands per length. Error bars are Wilson score intervals.

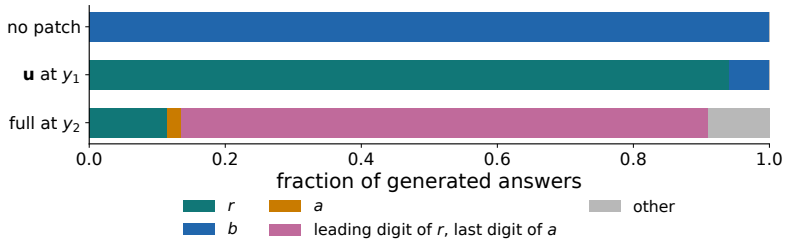

Figure 7: Answers generated by greedy decoding on the 400 held-out counterfactuals, as fractions per category: r, b, a, the leading digit of r followed by the last digit of a, and other. Rows: the corrupted prompt with no patch and with the full layer-13 residual patched at the last token of y_2 (both with y_2 perturbed), and with $\mathbf{u}$ patched at the position of y_1 (y_1 perturbed). An answer equal to r is counted as r.

Table 3: Examples of generated answers on held-out counterfactuals: the clean and corrupted operands, their maxima, and the answer generated under each intervention.

intervention	clean (y_1, y_2)	max	corrupted (y_1, y_2)	max	patched answer
$\mathbf{u}$ at y_1	(99, 61)	99	(12, 61)	61	12
$\mathbf{u}$ at y_1	(70, 55)	70	(33, 55)	55	33
full at y_2	(61, 99)	99	(61, 12)	61	12
full at y_2	(56, 97)	97	(56, 27)	56	27
full at y_2	(55, 70)	70	(55, 33)	55	30
full at y_2	(71, 89)	89	(71, 52)	71	59

C.2 WHERE THE COMPUTATION HAPPENS

Causal traces and linear probes over every layer and question token (Section B.4) locate the sites analysed in the rest of the appendix.

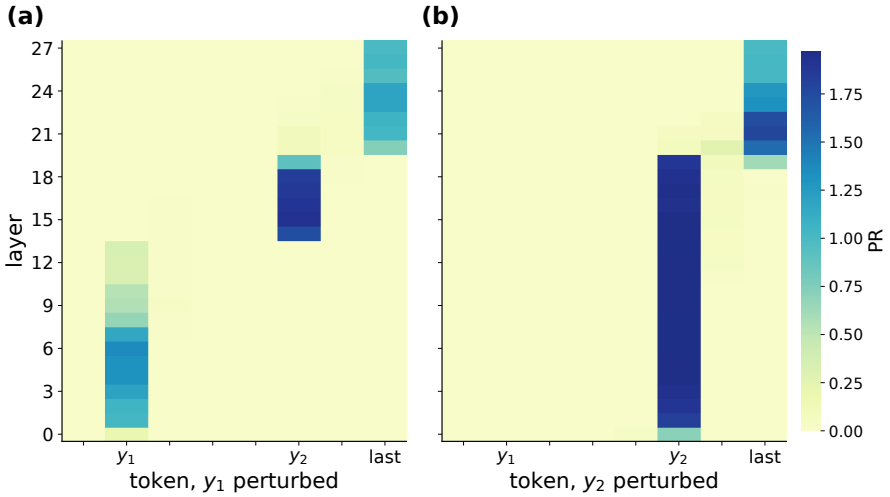

Figure 8: Position recovery when the clean residual stream leaving one layer (rows) is restored at one token (columns), from the first token of y_1 to the last token of the prompt, with (a) y_1 and (b) y_2 perturbed. Each operand spans two tokens and the tick marks its last one. The panels share a colour scale.

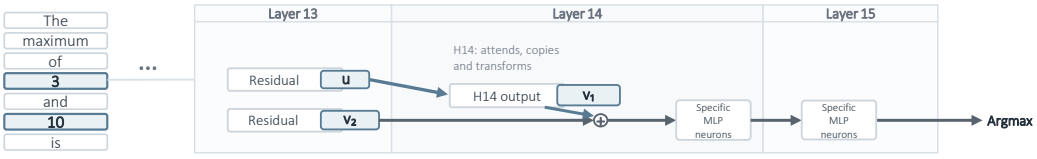

Figure 9: Illustration of the key components implementing algorithm underlying the pairwise number comparisons algorithm Alg. 1

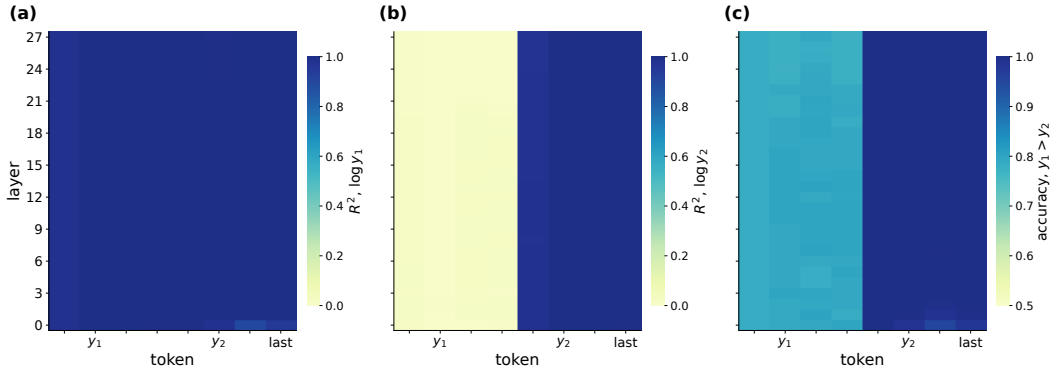

Figure 10: Ridge probes fitted at every layer (rows) and token (columns) on 2,000 clean two-number prompts and scored on a 20% held-out split. (a,b) Held-out R^2 for $\log y_1$ and $\log y_2$. (c) Held-out accuracy of a ridge classifier for $y_1 > y_2$. The axes are those of Figure 8.

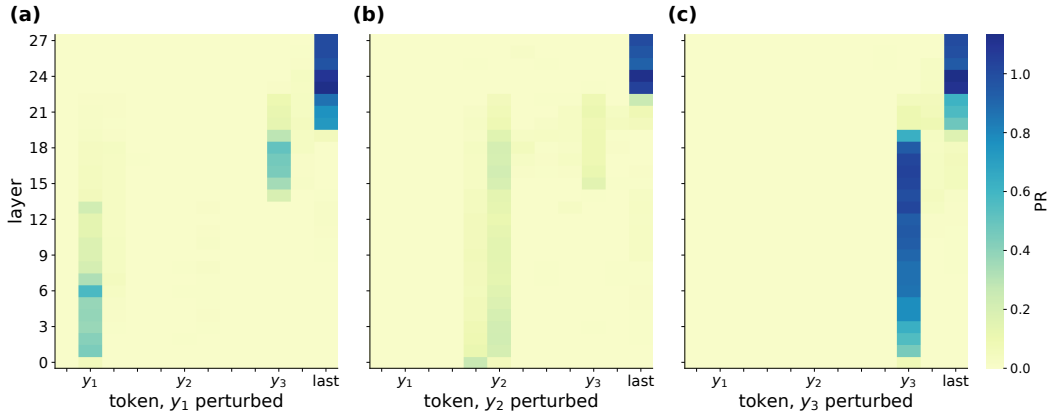

Figure 11: The causal trace of Figure 8 for three numbers, with (a) y_1, (b) y_2 and (c) y_3 perturbed, on the 400 held-out quadruples. The panels share a colour scale. These plots provide us with hypotheses about which layers are involved causally in processing the perturbed number at which token positions. y_2 is involved at both its own position as well as at y_3's position in the intermediate layers, while y_1, y_3 are processed at the y_3 position in these intermediate layers.

C.3 INDIVIDUAL NUMBER REPRESENTATIONS

For the direction $\mathbf{u}$ at the position of y_1, we show its place in the layer-13 geometry, a comparison with the top principal component, the same analysis at earlier layers and with three-digit operands. We then show the learned alternative to $\mathbf{v}_2$ and the output of the transport head L14.H14.

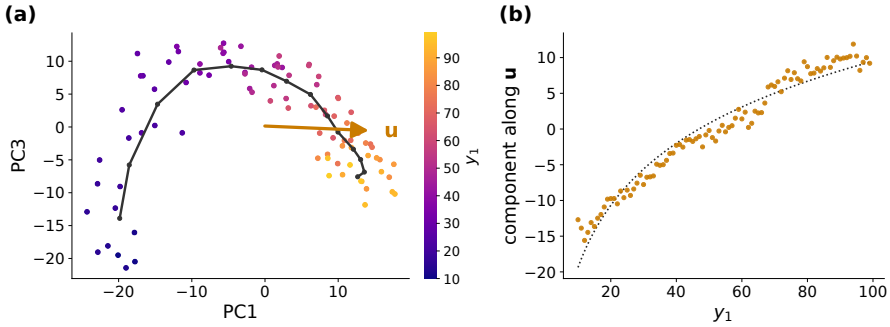

Figure 12: The layer-13 residual at the position of y_1, one point per value of y_1. (a) Projection on the first and third principal components, coloured by y_1, with the smoothed mean position along y_1 (line) and the direction of $\mathbf{u}$'s projection onto the plane (arrow). (b) Component along $\mathbf{u}$ against y_1, with the least-squares fit $p \log y_1 + q$ (dotted).

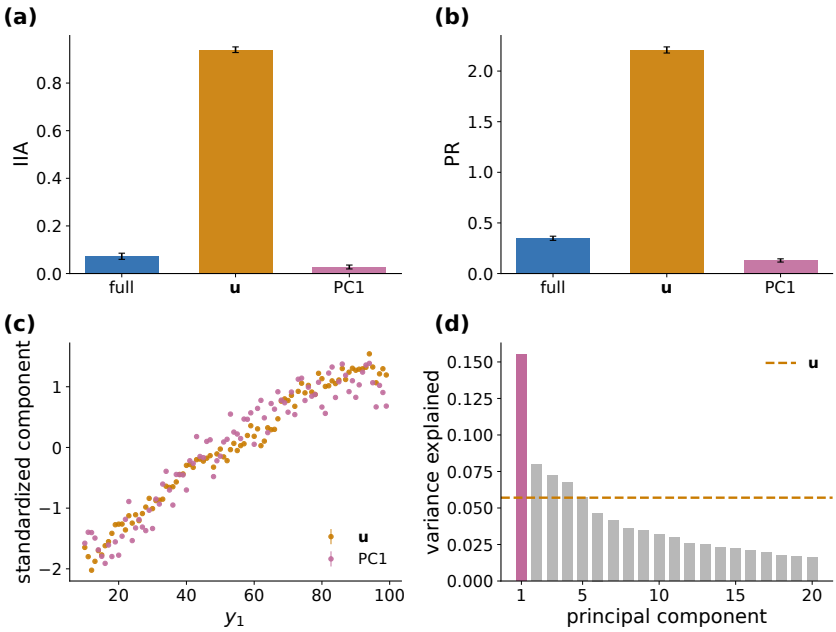

Figure 13: The layer-13 residual at the position of y_1, with y_1 perturbed. (a) IIA and (b) position recovery of a full-rank patch, a rank-one patch of $\mathbf{u}$ and a rank-one patch of the top principal component (PC1). (c) Standardized components along $\mathbf{u}$ and PC1 against y_1, as per-value means. (d) Variance explained by each of the first 20 principal components (bars) and by $\mathbf{u}$ (dashed line).

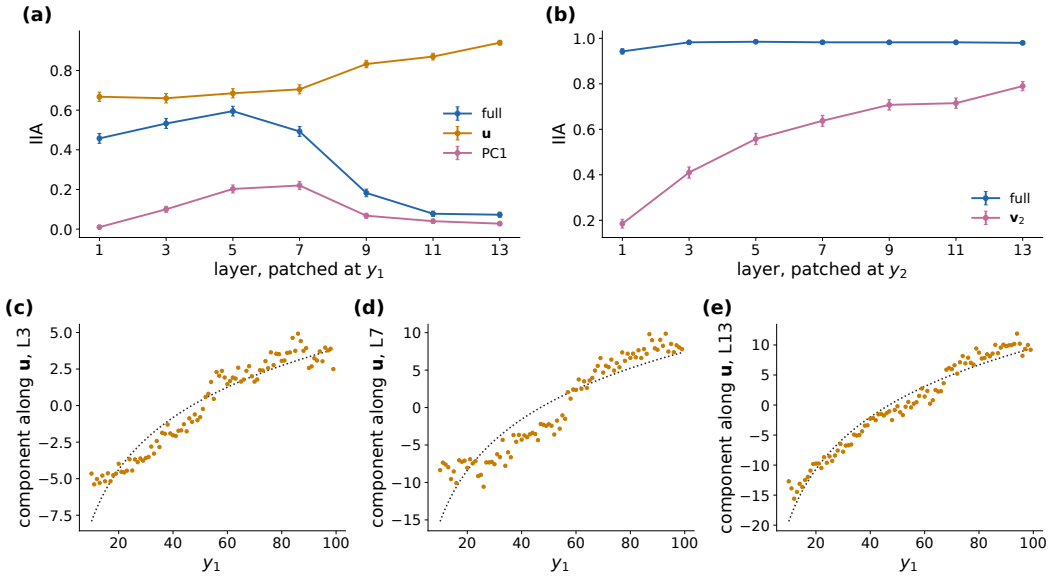

Figure 14: The number-representation analysis at layers 1 to 13, with $\mathbf{u}$ refitted at each layer. (a) IIA at the position of y_1, with y_1 perturbed, for a full-rank patch, $\mathbf{u}$ and that layer's top principal component. (b) IIA at the position of y_2, with y_2 perturbed, for a full-rank patch and $\mathbf{v}_2$, taken as that layer's top principal component. (c–e) Component along $\mathbf{u}$ against y_1 at layers 3, 7 and 13, as per-value means, with the fit $p \log y_1 + q$ (dotted).

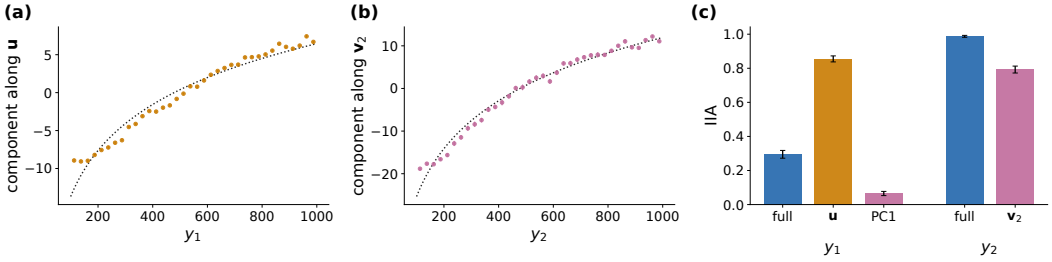

Figure 15: The number representations with three-digit operands. (a) Component along $\mathbf{u}$ at the position of y_1 against y_1, and (b) component along $\mathbf{v}_2$ at the position of y_2 against y_2, both averaged in bins of 25, with the fit $p \log y + q$ (dotted). (c) IIA of a full-rank patch, $\mathbf{u}$ and PC1 at the position of y_1 with y_1 perturbed, and of a full-rank patch and $\mathbf{v}_2$ at the position of y_2 with y_2 perturbed.

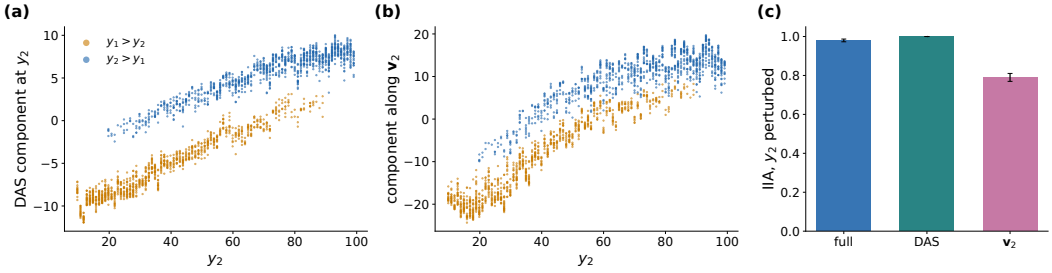

Figure 16: The layer-13 residual at the position of y_2 on the 2,000-prompt cloud, coloured by which number is larger. (a) Component along a rank-one direction fitted with y_2 perturbed, against y_2. (b) Component along $\mathbf{v}_2$, the top principal component, against y_2. (c) IIA with y_2 perturbed for a full-rank patch, the fitted direction (DAS) and $\mathbf{v}_2$.

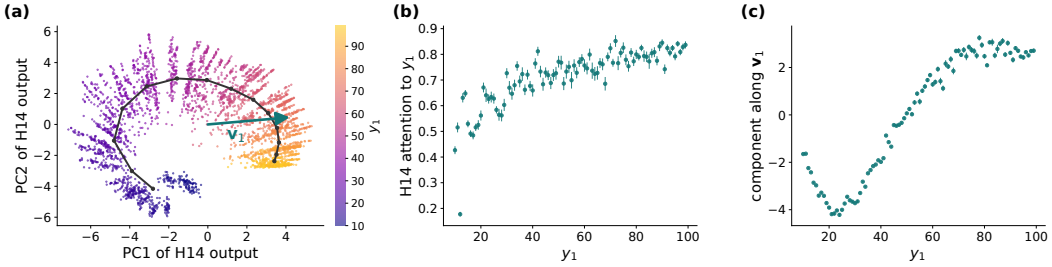

Figure 17: The output of head L14.H14 at the position of y_2. (a) Its top two principal components, coloured by y_1, with the smoothed mean position along y_1 (line) and the direction of $\mathbf{v}_1$'s projection onto the plane (arrow). (b) The head's attention from the position of y_2 to y_1, against y_1, as per-value means. (c) Component along $\mathbf{v}_1$ against y_1, as per-value means.

C.4 THE SHARED REPRESENTATION

Probes and position recovery complement the main-text IIA results for the shared representation at the position of y_2.

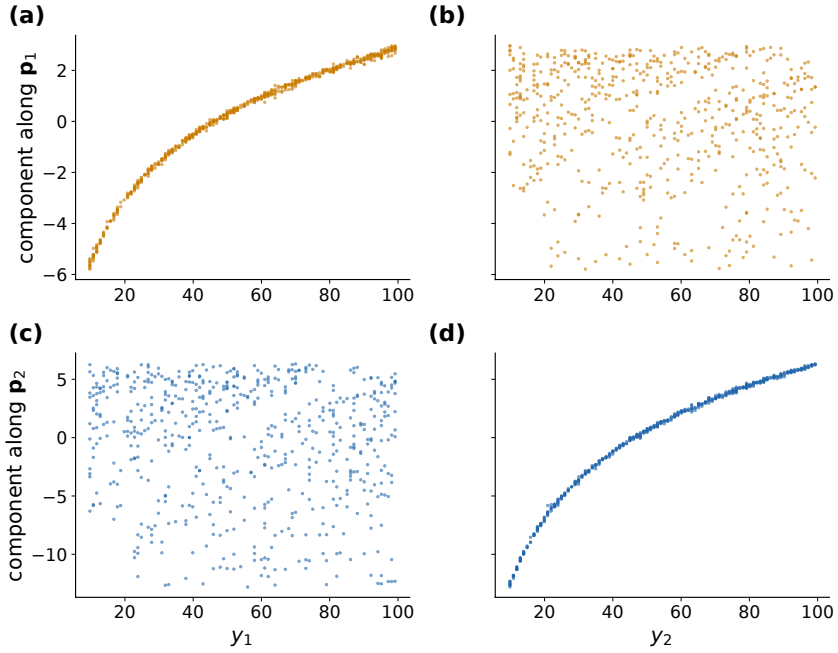

Figure 18: Components of the pre-MLP layer-14 residual at the position of y_2 along ridge probes for $\log y_1$ ($\mathbf{p}_1$, top row) and $\log y_2$ ($\mathbf{p}_2$, bottom row), against y_1 (left) and y_2 (right), on 600 held-out prompts.

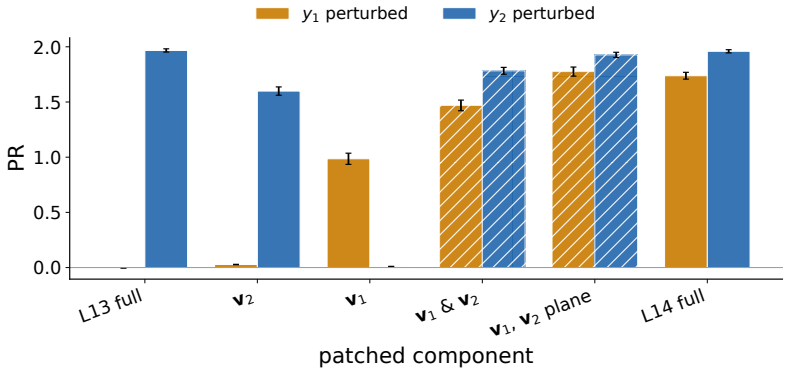

Figure 19: Position recovery at the position of y_2 for the patches of the main-text shared-representation figure, with y_1 or y_2 perturbed: the full layer-13 residual, $\mathbf{v}_2$, $\mathbf{v}_1$, $\mathbf{v}_1$ and $\mathbf{v}_2$ as two separate rank-one patches, the rank-two $(\mathbf{v}_1, \mathbf{v}_2)$ plane in the pre-MLP layer-14 residual, and the full pre-MLP layer-14 residual. Hatched bars are the two conditions that patch both directions.

C.5 THE COMPARATOR NEURONS

Freezing, attribution, receptive fields and connectivity for MLP14 and MLP15 appear in the order of the selection procedure of Sections B.5–B.6.

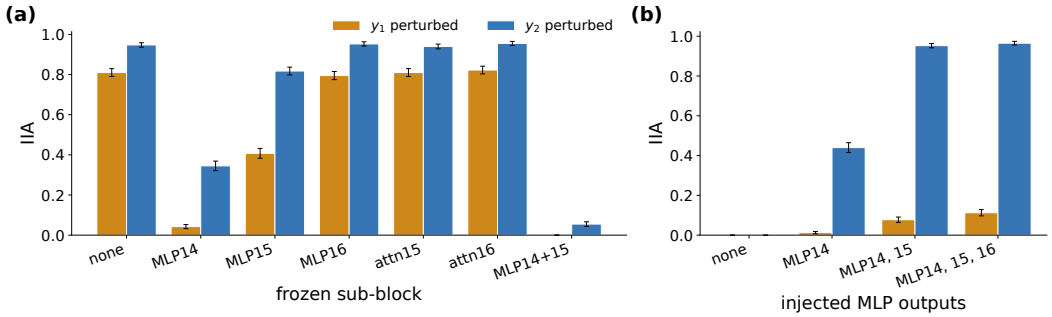

Figure 20: (a) IIA of the rank-two $(\mathbf{v}_1, \mathbf{v}_2)$ plane patch when one sub-block's output at the position of y_2 is frozen to its corrupted value, with y_1 or y_2 perturbed. The layer-14 attention freeze is omitted, because the pre-MLP patch is written through that output (Section B.1). MLPs in layer 14, 15 are important for upstream $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ patching to work. (b) IIA when only the clean outputs of the listed MLPs are injected at the position of y_2.

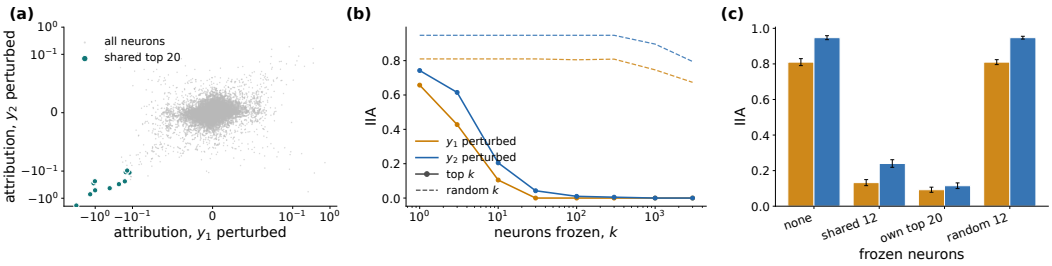

Figure 21: (a) Attribution scores of the 37,888 neurons of MLP14 and MLP15, with y_1 perturbed (x-axis) and y_2 perturbed (y-axis), on symmetric-log axes. The 12 neurons in the top 20 of both cases are highlighted. (b) IIA of the plane patch when the top k neurons by attribution (solid) or k random neurons (dashed) are frozen. (c) IIA with no neurons frozen, with the 12 shared neurons frozen, with each case's own 12 top-ranked neurons frozen, and with 12 random neurons frozen.

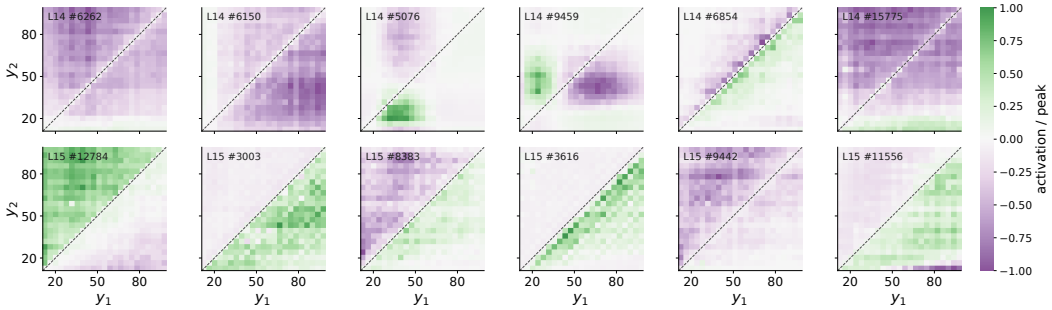

Figure 22: Receptive fields and tuning curves of the 12 shared neurons, six in MLP14 (top) and six in MLP15 (bottom), ordered by worst-case attribution rank. Each field is the post-SwiGLU activation at the position of y_2 over a grid of (y_1, y_2) prompts with stride 4, divided by its peak. The dashed line is $y_1 = y_2$.

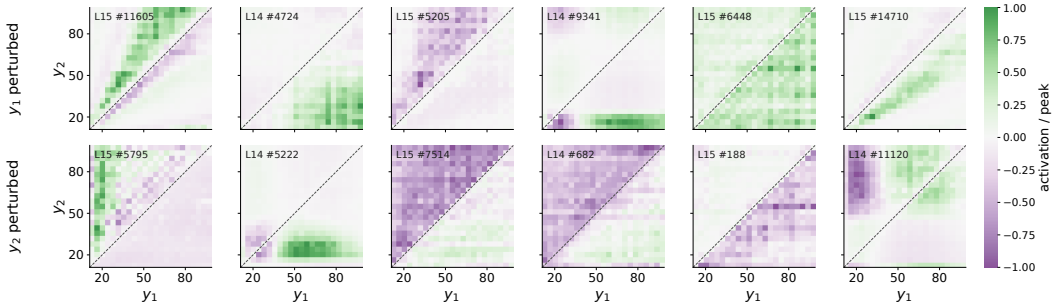

Figure 23: Receptive fields and tuning curves, drawn as in Figure 22, of the six highest-ranked neurons that are in the top 20 of one perturbation case only: y_1 perturbed (top) and y_2 perturbed (bottom).

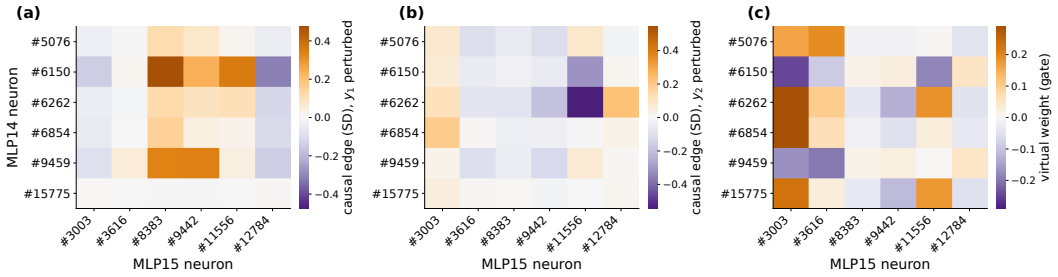

Figure 24: (a,b) Causal edges from each shared MLP14 neuron (rows) to each shared MLP15 neuron (columns): the shift in the MLP15 neuron's activation, in units of its standard deviation, when only the MLP14 neuron is set to its clean value, with (a) y_1 or (b) y_2 perturbed. (c) Virtual weights through the gate projection, computed as the cosine between each MLP14 neuron's write direction and each MLP15 neuron's normalization-folded gate read direction (Section B.6).

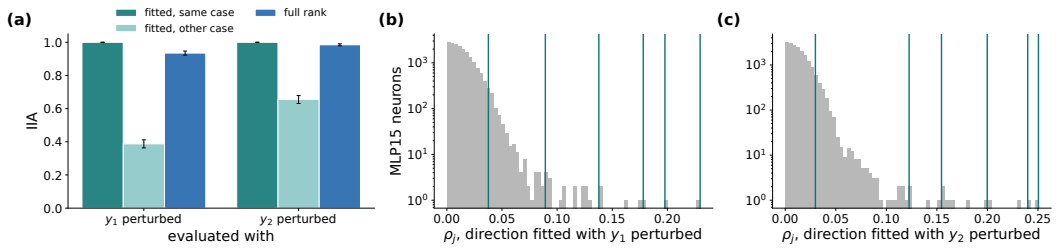

Figure 25: (a) IIA of a rank-one direction in the layer-15 residual at the position of y_2, fitted in the case it is evaluated on (fitted, same case) or in the other case (fitted, other case), and of the full-rank layer-15 patch, for each perturbation case. (b,c) Histograms of the alignment ρ_j of the write direction of every MLP15 neuron with the direction fitted with (b) y_1 or (c) y_2 perturbed, on a log count axis. Vertical lines mark the six shared MLP15 neurons.

C.6 THREE NUMBERS

A summary figure covers the positions of y_3 and y_2 and the last token, and the figures after it expand each part.

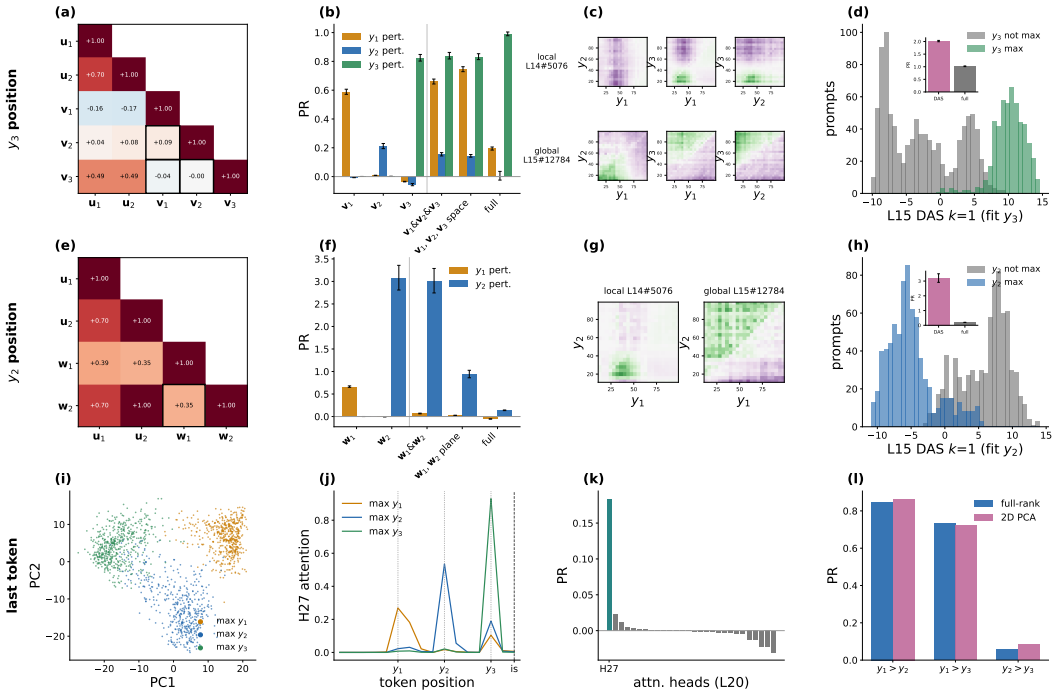

Figure 26: The three-number analysis at the position of y_3 (top row), the position of y_2 (middle row) and the last token (bottom row). (a) Cosine similarities between $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$; the pairs among $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are boxed. (b) Position recovery at the position of y_3, by perturbed number, for each direction alone, the three as separate rank-one patches, their span in the pre-MLP layer-14 residual, and the full layer-14 residual. (c) Receptive fields of L14#5076 (top) and L15#12784 (bottom) at the position of y_3 over a (y_1, y_2, y_3) grid with stride 4, shown as the three pairwise marginals, each averaged over the third number and divided by its own peak. (d) The 1,500-prompt cloud projected on a rank-one direction in the layer-15 residual fitted with y_3 perturbed, split by whether y_3 is the maximum; the inset shows the position recovery of this direction and of the full layer-15 residual. (e–h) The same at the position of y_2, where $\mathbf{w}_1$ and $\mathbf{w}_2$ are rank-one directions in the layer-13 residual fitted with y_1 and y_2 perturbed ($\mathbf{w}_2$ and $\mathbf{u}_2$ are the same fit). The fields in (g) are over (y_1, y_2) with $y_3 = 55$. (i) Top two principal components of the residual leaving layer 20 at the last token, coloured by which operand holds the maximum. (j) Attention of head L20.H27 from the last token to the question tokens, averaged by the position of the maximum. (k) Position recovery of each layer-20 head's own output at the last token, for the order $y_1 > y_2 > y_3$. (l) Position recovery of a full-rank patch and of a patch of the top two principal components of the layer-20 residual at the last token, for three value orders; the principal components are fitted on a separate 1,500-prompt cloud.

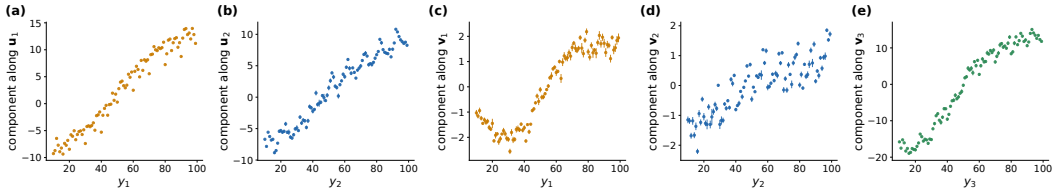

Figure 27: Component along each direction of the three-number analysis against the number it carries, as per-value means on the 1,500-prompt cloud. (a,b) $\mathbf{u}_1$ and $\mathbf{u}_2$ in the layer-13 residual at the positions of y_1 and y_2. (c,d) $\mathbf{v}_1$ and $\mathbf{v}_2$ in the outputs of heads L14.H14 and L14.H18 at the position of y_3. (e) $\mathbf{v}_3$ in the layer-13 residual at the position of y_3.

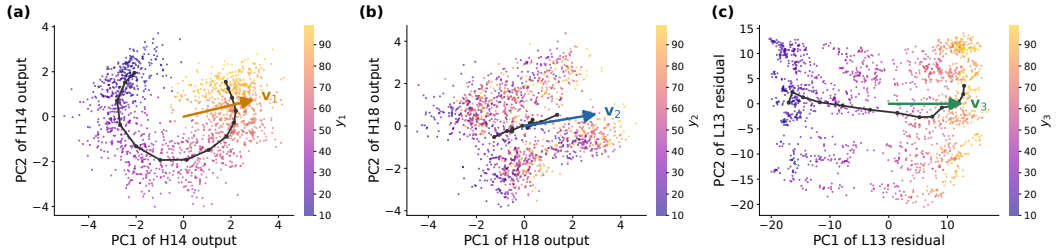

Figure 28: The spaces at the position of y_3 that contain the three-number directions, each in its top two principal components, coloured by the number its direction carries, with the smoothed mean position along that number (line) and the direction of the corresponding projection onto the plane (arrow). (a) The output of head L14.H14 with $\mathbf{v}_1$. (b) The output of head L14.H18 with $\mathbf{v}_2$. (c) The layer-13 residual with $\mathbf{v}_3$.

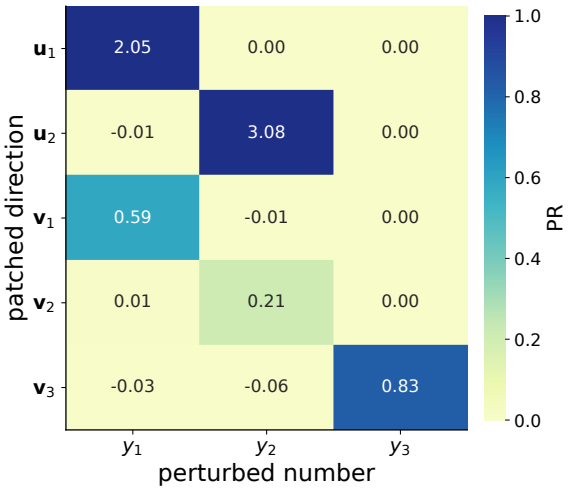

Figure 29: Position recovery of a rank-one patch of each direction (rows) in each perturbation case (columns). Colours are clipped at 1; printed values are not.

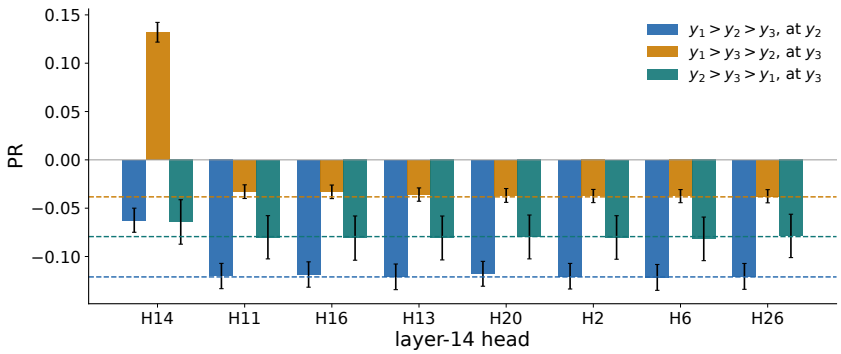

Figure 30: Position recovery of each layer-14 head's own output, interchanged at the runner-up's position together with the full layer-13 residual there, for three value orders and the eight heads with the largest mean absolute effect. Dashed lines show the layer-13 patch alone for each order.

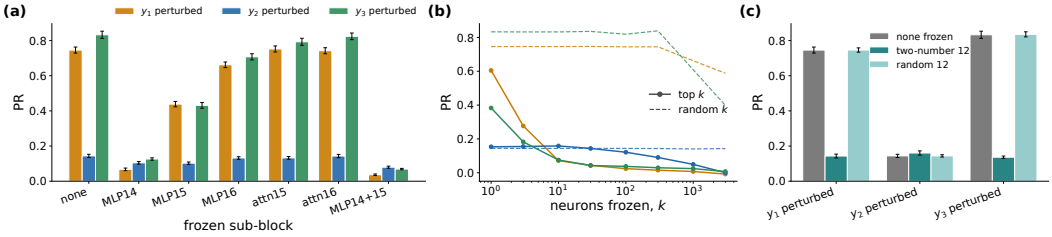

Figure 31: Position recovery of the patch of the span of $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ in the pre-MLP layer-14 residual at the position of y_3, by perturbed number, while parts of the network are frozen. (a) One sub-block's output at the position of y_3 frozen to its corrupted value. (b) The top k neurons by this task's attribution ranking (solid) or k random neurons (dashed) frozen. (c) No neurons frozen, the 12 shared neurons of the two-number task frozen, and 12 random neurons frozen.

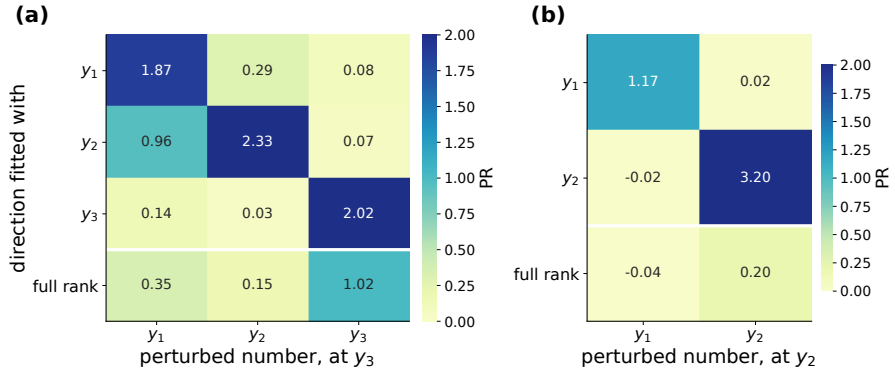

Figure 32: Position recovery of a rank-one direction in the layer-15 residual fitted in one case (rows) and evaluated in each case (columns), with the full-rank layer-15 patch in the last row, (a) at the position of y_3 and (b) at the position of y_2. Colours are clipped at 2; printed values are not.

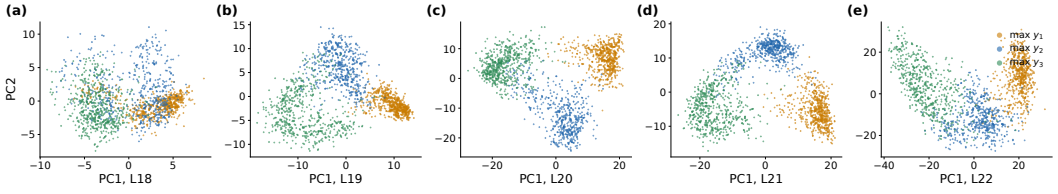

Figure 33: Top two principal components of the residual stream leaving layers 18 to 22 at the last token, on 1,500 clean three-number prompts, coloured by which operand holds the maximum.

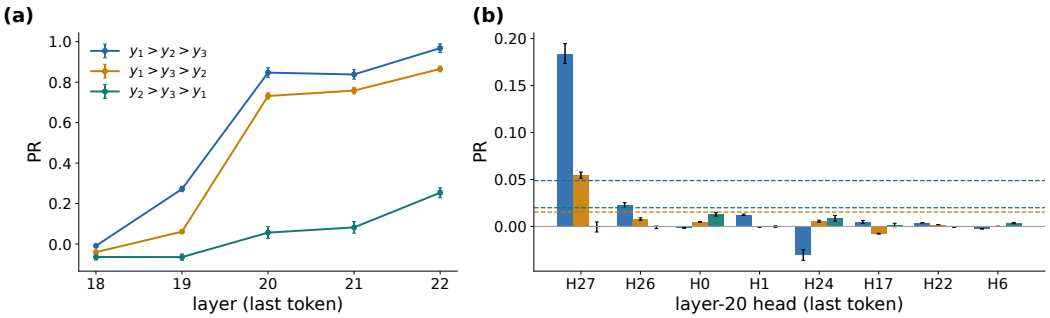

Figure 34: (a) Position recovery of a full-rank patch of the residual leaving each of layers 18 to 22 at the last token, for three value orders. (b) Position recovery of each layer-20 head's own output at the last token, for the eight heads with the largest effect and the same three orders. Dashed lines show a patch of the whole layer-20 attention sublayer for each order (Section B.4).

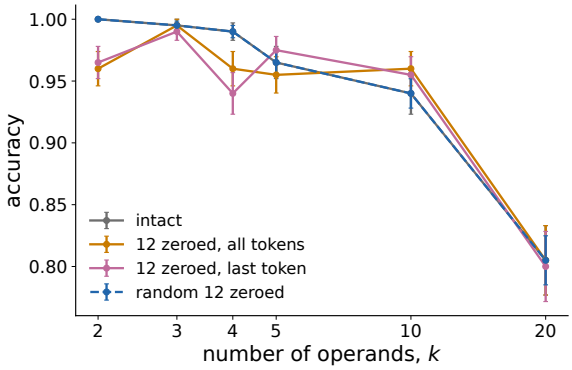

Figure 35: Accuracy on $\max(y_1, \ldots, y_k)$ for 200 two-digit prompts per length, scored on the full decoded number (Section B.2): the intact model, the 12 shared neurons zeroed at every token, the same neurons zeroed only at the last token of the final operand ("last token" in the legend), and 12 random neurons zeroed at every token (two random sets, pooled).