Understanding Gated Neurons in Transformers from Their Input-Output Functionality

Anonymous ACL submission

Abstract

Interpretability researchers have attempted to understand MLP neurons of language models based on both the contexts in which they activate and their output weight vectors. They have paid little attention to a complementary aspect: the interactions between input and output. For example, when neurons detect a direction in the input, they might add much the same direction to the residual stream ("enrichment neurons") or reduce its presence ("depletion neurons"). We address this aspect by examining the cosine similarity between input and output weights of a neuron. We apply our method to 12 models and find that enrichment neurons dominate in early-middle layers whereas later layers tend more towards depletion. To explain this finding, we argue that enrichment neurons are largely responsible for enriching concept representations, one of the first steps of factual recall. Our input-output perspective is a complement to activation-dependent analyses and to approaches that treat input and output separately.

1 Introduction

Despite recent progress in interpretability, there is still much that is unclear about how transformer-based (Vaswani et al., 2017) large language models (LLMs) achieve their impressive performance. Prior work has addressed the interpretation of MLP sublayers, and we follow this line of research. Some of this work analyzes neurons based only on the contexts in which they activate (Voita et al., 2024) or based only on their output weights[1] (Gurnee et al., 2024). In contrast, we put the input-output (IO) functionality of neurons in the center of our analysis, and classify neurons according to the interactions between input and output weights. We focus on gated activation functions (Shazeer, 2020), which are used in recent LLMs like OLMo, Llama and Gemma.

Theoretical framework. Following Elhage et al. (2021), our view of the Transformer architecture is centered on the residual (a.k.a. skip) connections between sublayers: they form the *residual stream*, and the individual units (such as MLP neurons) progressively update it, until it is multiplied by the unembedding matrix W_U to produce next-token logits. The information contained in the residual stream is represented as a high-dimensional vector (of dimension d_{model}). Individual model units *read* from the residual stream and then update it by *writing* (adding) other vectors to it. In the case of an MLP neuron, it detects certain directions in the residual stream (i.e., whether the current residual stream vector at least approximately points in one of these directions in model space), corresponding to its weight vectors on the input side; and then writes to a certain direction, corresponding to its output weight vector.

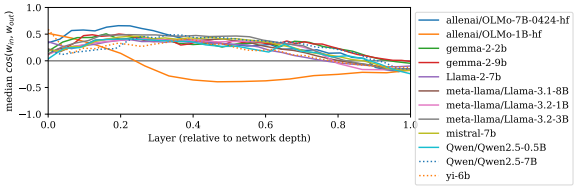


Figure 1: Median of $\cos(w_{\text{in}}, w_{\text{out}})$ by layer (x-axis) for 12 models. For all models, the value is positive in the beginning and negative in the end, indicating that early-middle layers "enrich" the residual stream whereas later layers tend more towards depletion.

A semantic intepretation is that a neuron detects a *concept* in the residual stream (for example, an intermediate guess about the next token), and in turn also writes a concept. This semantic interpretation is not a necessary assumption for our neuron classification, but is helpful for building intuition and interpreting results.

Theoretical contribution. These theoretical reflections naturally lead to our research question: **What is the relationship between what a neuron reads and what it writes?** We address this question by computing the **cosine similarity of input and output weights**, focusing on gated activation functions.

Specifically, with gated activation functions, each neuron has three weight vectors: the linear input, gate, and output weight vectors. When the output weight is similar enough to (one of) the detected directions, we speak of **input manipulation**, as opposed to **orthogonal output** neurons which write to directions not detected in the input. Intuitively, input manipulator neurons *manipulate* the concept that they detect. As special cases of input manipulation, we define **enrichment** and **depletion** neurons – neurons that detect a direction and then add it to / remove it from the residual stream. We

[1] We use "weight" to refer to a weight vector, not a scalar.

present a complete taxonomy of neuron IO functionalities in Section 4. See Figure 2 for a visualization.

Empirical study. We apply our method to 12 LLMs. We find that, for all of these models, a large proportion of neurons are input manipulators. In particular, we find that enrichment neurons dominate in early-middle layers of all models whereas later layers tend more towards depletion. See Figure 1.

We also present examples for the six major IO functionalities. We find that many neurons have the property of **double checking**: The two reading weight vectors (w_{gate} and w_{in}) are approximately orthogonal, but still intuitively represent the same concept.

Explaining the results. Our finding of different IO functionalities in different layers echoes the "stages of inference" framework (Lad et al., 2024). We hypothesize a correspondence: enrichment neurons may be responsible for "feature engineering" and depletion neurons for "residual sharpening".

We also provide a theoretical account of the double checking phenomenon. The usefulness of double checking explains the fact that many neurons have approximately orthogonal gate and input weights.

Contributions. (i) We develop a parameter-based (and therefore efficient) method to investigate neuron IO functionalities for gated activation functions (Section 4). (ii) Across 12 models, we find that enrichment neurons dominate in early-middle layers of all models whereas later layers tend more towards depletion (Figure 1). (iii) We define two novel concepts helpful in understanding neuron functionality: *input manipulation* and *double checking*. (iv) We find that many neurons are input manipulators (Section 5), which makes our classification scheme useful for understanding them. (v) We present examples for the six major IO functionalities, showing how the IO perspective complements other neuron analysis methods (Section 6). (vi) We propose theoretical explanations for some of these results (Section 7).

2 Related Work

There is a large body of work on interpretability of transformer-based LLMs. Elhage et al. (2021) introduce the notion of residual stream. nostalgebraist (2020), Belrose et al. (2023) propose to interpret residual stream states as intermediate guesses about the next token; Rushing and Nanda (2024) discuss this as the *iterative inference hypothesis*. On a similar note, many works hypothesize that directions in model space can correspond to concepts; Park et al. (2024) discuss this as the *linear representation hypothesis*. Lad et al. (2024) define *stages of inference*. Geva et al. (2023) explain how LLMs recall facts; a crucial early step is *representation enrichment*, which may be related to our *enrichment neurons* (see Section 7.4). Similar to our work, Elhelo and Geva (2024) investigate input-output functionality of heads (instead of neurons).

Much research has attempted to understand individual neurons. Geva et al. (2021) present them as a key-value memory. Other neuron analysis work includes (Miller and Neo, 2023; Niu et al., 2024). The focus on individual neurons has been criticized. Morcos et al. (2018) find that in good models, neurons are not monosemantic (but for image models, not LLMs). Millidge and Black (2022) compute a singular value decomposition (SVD) of layer weights and often find interpretable directions that do not correspond to individual neurons. Elhage et al. (2022) argue that interpretable features are non-orthogonal directions in model space and can be superposed. This corresponds to sparse linear combinations of neurons in MLP space. Taking the middle ground, Gurnee et al. (2023) argue that interpretable features correspond to sparse combinations of neurons, but this includes 1-sparse combinations, i.e., individual neurons.

Several works classify neurons based on the **contexts** in which they activate (Voita et al., 2024; Gurnee et al., 2024). For example, Voita et al. (2024) find *token detectors* that suppress repetitions. Gurnee et al. (2024) also define *functional roles* of neurons based on their **output** weight vector, such as *suppression neurons* that suppress a specific set of tokens. They note that suppression neurons seem to activate "when it is plausible but not certain that the next token is from the relevant set". Stolfo et al. (2024) also investigate some output-based neuron classes.

Researchers have paid less attention to the input-output perspective. Gurnee et al. (2024) compute cosine similarities between input and output weights for GPT-2 (Radford et al., 2019), but do not interpret their results. Elhage et al. (2022) mention the idea of input-output analysis (negative cosines between input and output weights "may also be mechanisms for conditionally deleting information", footnote 7), but do not follow up on this remark. Note also that input-output analysis for gated activation functions adds complexity because, in addition to input and output weight vectors, the gating mechanism is crucial for IO functionality.

3 Gated activation functions

In our neuron classification we assume *gated activation functions* like SwiGLU or GeGLU (Shazeer, 2020). In this section, we describe definition (Section 3.1) and properties (Section 3.2) of these functions. Gated activation functions are used widely, e.g., OLMo (Groeneveld et al., 2024) and Llama (Touvron et al., 2023) use SwiGLU, and Gemma (Gemma, 2024) uses GeGLU.

The following description focuses on SwiGLU. GeGLU replaces Swish with GeLU, but is otherwise identical. For a visualization of a SwiGLU neuron, see Figure 6 in Section E.

3.1 Definitions

To keep our description simple, we ignore bias terms and layer norm parameters. (Some models, like OLMo, lack these anyway.) We describe single neurons as opposed to whole MLP layers.

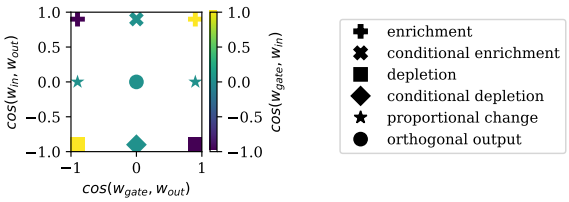


Figure 2: We define six input-output functionality classes or **IO classes** of gated activation neurons based on collinearity and orthogonality of their linear input, gate and output weight vectors. For example, depletion neurons remove the direction of the gate vector from the residual stream. Examples shown are prototypical.

We denote by x_{mid} the state of the residual stream before the MLP, and by $x_{\text{norm}} := \text{LN}(x_{\text{mid}})$ its layer normalization. We say that a **direction** $v \in \mathbb{R}^d$ is **present** (positively) in a vector $x \in \mathbb{R}^d$ if $x \cdot v \gg 0$.

Traditional activation functions like ReLU take a single scalar as argument: $\text{ReLU}(x_{\text{in}})$. In contrast, a *gated activation function* like SwiGLU takes two arguments:

$$\text{SwiGLU}(x_{\text{gate}}, x_{\text{in}}) = \text{Swish}(x_{\text{gate}}) \cdot x_{\text{in}}.$$

To compute the scalars x_{gate} and x_{in}, each neuron has a **linear input** weight vector w_{in} and a **gate** weight vector w_{gate} of dimension d_{model}. We refer to these two weight vectors as the **reading weights**. Then x_{gate} is defined as $w_{\text{gate}} \cdot x_{\text{norm}}$, and x_{in} as $w_{\text{in}} \cdot x_{\text{norm}}$.

Finally, the product of $\text{SwiGLU}(x_{\text{gate}}, x_{\text{in}})$ and the **output** weight vector, w_{out} , is added to the residual stream.

3.2 Properties

There are three properties of gated activation functions that are key for understanding IO functionality.

Positive vs negative activation. Strong activations can be either positive or negative. If $w_{\text{gate}} \cdot x_{\text{norm}} \gg 0$ and $w_{\text{in}} \cdot x_{\text{norm}} \gg 0$, the activation is strongly positive. If $w_{\text{gate}} \cdot x_{\text{norm}} \gg 0$ and $w_{\text{in}} \cdot x_{\text{norm}} \ll 0$, the activation is strongly negative. So, depending on the context, a given gated activation neuron can either add the output weight vector to the residual stream or subtract it.

Negative values of Swish. Swish and GeLU are often seen as essentially ReLU. However, we found clearly different cases (see Section 6). $w_{\text{gate}} \cdot x_{\text{norm}}$ can be **weakly negative**, i.e., negative but close to zero. In this case its image under Swish is also weakly negative. This leads to a negative activation if w_{in} is present positively and positive otherwise.

Symmetry. Switching the signs of both w_{in} and w_{out} preserves IO behavior.

4 Method

We now describe how we investigate input-output functionalities of gated neurons, based on their weights only.

4.1 Intuition

As a running example, we consider what a neuron would do to a residual stream state representing the next-token prediction *review*.

Before we introduce our method, let us consider a simpler case to develop our intuition: non-gated activation functions like ReLU (see also Gurnee et al. (2024)). Here, a neuron detects just one direction, determined by its input weight w_{in} (say *review*). (Given x_{norm}, the activation depends only on $x_{\text{norm}} \cdot w_{\text{in}}$, and is positive whenever this is positive.) Roughly, we can distinguish three cases: the neuron output (determined by w_{out}) can be similar to the input direction (in our case, *review*: we call this *enrichment*), different (we call this *orthogonal output*), or roughly opposite (in our case, "minus *review*": we call this *depletion*). In terms of weights, these cases correspond to $cos(w_{\text{in}}, w_{\text{out}})$ being close to 1, close to 0, or close to -1.

Note that a neuron could also detect "minus *review*" (i.e., "*review* is not the next token"), and enrich or deplete that direction.

4.2 Extension to gated activation functions

In this paper, we consider gated activation functions. Here, a neuron detects two directions (w_{gate} and w_{in}), not one; so there are more cases to consider. Luckily, the symmetry property (see Section 3.2) simplifies the analysis: a neuron's behavior does not change if we switch the signs of both w_{in} and w_{out}. This implies that *the sign of* $\cos(w_{gate}, w_{out})$ *does not matter*.

Accordingly, we define six IO classes, depending on $cos(w_{\text{in}}, w_{\text{out}})$ (three rows: positive, negative, or zero) and $|cos(w_{\text{gate}}, w_{\text{out}})|$ (two columns: positive or zero). Although there is a third cosine similarity – $cos(w_{\text{gate}}, w_{\text{in}})$ – this similarity is determined by the two others in prototypical cases. We will consider these prototypical cases first.

4.3 Prototypical cases

See Table 1 for an overview of all cases and Figure 2 for a visualization. We also encourage the use of the interactive visualization in supplementary.

For the prototypical cases we assume the cosines are ≈ 1, ≈ -1 or ≈ 0. In these cases, knowing two of the cosine similarities implies knowing the third one: If w_{gate} and w_{in} are collinear, then w_{out} has the same cosine similarity with both (up to sign). Conversely, if w_{gate} and w_{in} are orthogonal, w_{out} cannot be collinear to both, and in fact, $\cos(w_{\text{gate}}, w_{\text{out}})^2 + \cos(w_{\text{in}}, w_{\text{out}})^2 \leq 1$, with equality when w_{out} is in the space spanned by w_{gate} and w_{in}.

We first focus on on textbfenrichment and depletion: $\cos(w_{\text{in}}, w_{\text{out}}) \approx \pm 1$. The gate vector can be collinear as well, i.e., $\cos(w_{\text{gate}}, w_{\text{out}}) \approx \pm 1$ (leftmost "typical" column). In this case, all three vectors are approximately in a one-dimensional subspace, so the neuron detects one direction and writes to the same direction, up to sign. The sign is relevant: Assume x_{norm} represents the token

$\cos(w_{\text{in}}, w_{\text{out}})$	$\lvert\cos(w_{\text{gate}}, w_{\text{out}})\rvert \gg 0$		$\cos(w_{\text{gate}}, w_{\text{out}}) \approx 0$	
$\gg 0$	**enrichment**		**conditional enrichment**	
	$\lvert\cos(w_{\text{gate}}, w_{\text{in}})\rvert \gg 0$	$\lvert\cos(w_{\text{gate}}, w_{\text{in}})\rvert \approx 0$	$\lvert\cos(w_{\text{gate}}, w_{\text{in}})\rvert \approx 0$	$\lvert\cos(w_{\text{gate}}, w_{\text{in}})\rvert \gg 0$
	typical	atypical	typical	atypical
$\ll 0$	**depletion**		**conditional depletion**	
	$\lvert\cos(w_{\text{gate}}, w_{\text{in}})\rvert \gg 0$	$\lvert\cos(w_{\text{gate}}, w_{\text{in}})\rvert \approx 0$	$\lvert\cos(w_{\text{gate}}, w_{\text{in}})\rvert \approx 0$	$\lvert\cos(w_{\text{gate}}, w_{\text{in}})\rvert \gg 0$
	typical	atypical	typical	atypical
≈ 0	**proportional change**		**orthogonal output**	
	$\lvert\cos(w_{\text{gate}}, w_{\text{in}})\rvert \approx 0$	$\lvert\cos(w_{\text{gate}}, w_{\text{in}})\rvert \gg 0$		
	typical	atypical		

Table 1: Our six IO classes, in **boldface**. Five of them have "atypical" variants. We use a threshold of 0.5 (resp. -0.5) to distinguish $\cos() \approx 0$ from $\lvert\cos()\rvert \gg 0$.

review and w_{gate} detects that direction, so that the neuron activates. If $cos(w_{\text{in}}, w_{\text{out}}) \approx 1$ ($w_{\text{in}}, w_{\text{out}}$ also lie in the *review* subspace, and both have the same orientation), the neuron will again write *review*. We call this (typical) **enrichment**. On the other hand, if $cos(w_{\text{in}}, w_{\text{out}}) \approx -1$ (they again lie in the *review* subspace but have different orientations), the neuron will write "minus *review*". We call this (typical) **depletion**.[2] The same neurons can also get a *weak negative* activation if $-w_{\text{gate}}$ ("minus *review*") is weakly present in the residual stream. In this case, Swish has a negative value (Section 3.2) and the enrichment neuron writes "plus *review*" to the residual stream and the depletion neuron "minus *review*".

Next we consider **conditional enrichment** and **conditional depletion**: w_{in} and w_{out} are roughly collinear and w_{gate} is orthogonal to them. Consider the example that $w_{\text{in}}, w_{\text{out}}$ correspond to the *review* direction and w_{gate} to "verb expected as next token". The neuron will only activate *conditional on w_{gate} being present in the residual stream* (here: verb expected). If $\pm w_{\text{in}}$ ("plus" or "minus" *review*) is *also* present in the residual stream, then $\pm w_{\text{out}}$ ("plus" or "minus" *review*) will be added to the residual stream. For this scenario, we define a (typical) **conditional enrichment** neuron as one with $\cos(w_{\text{in}}, w_{\text{out}}) \approx 1$; this neuron will enrich the residual stream with w_{in} if w_{in} is present and with $-w_{\text{in}}$ if $-w_{\text{in}}$ is present ("plus" *review* leads to more of "plus" *review*, and "minus" *review* leads to more of "minus" *review*). Conversely, we define a (typical) **conditional depletion** neuron as one that depletes $\pm w_{\text{in}}$ (whichever was present) from the residual stream: "plus" *review* leads to "minus" *review* and vice versa. As before, if $-w_{\text{gate}}$ is weakly present in the residual stream (there is a weak expectation that the next token is not a verb), Swish yields a negative value; so in this situation conditional enrichment and depletion neurons switch their behaviors; e.g., for a conditional enrichment neuron "plus" *review* will lead to "minus" *review*.

Turning to the bottom part of Table 1, we define a (typical) **proportional change neuron** as one whose w_{out} is in the same one-dimensional subspace as w_{gate}, but is orthogonal to w_{in}. (This implies that w_{gate} and w_{in} are orthogonal.) Take the case where $w_{\text{gate}}, w_{\text{out}}$) are represent *review* and w_{in} "verb expected". If w_{gate} (*review*) is present in the residual stream, then the neuron writes a *positive or negative* multiple of *review* to the residual stream. This multiple is proportional to the presence of w_{in} ("verb expected") in the residual stream: If a verb is expected, the neuron writes *review*, if not, it writes "minus *review*".

All of the above neuron types are **input manipulators**: they write to one of the directions they detect. Our final category is the negation of this: We define an **orthogonal output neuron** as one whose output weight vector is orthogonal to both reading weight vectors. If w_{gate} and w_{in} are also orthogonal to each other, then such a neuron defines an interaction of three completely different meaning components.

4.4 General case: Typical vs atypical functions

Many cosines will not be close to 0 or ± 1. For example, such a neuron may write a concept different from but semantically related to the one it detects (say, *Ireland* -> *Dublin*) and thus be be similar to an enrichment neuron in terms of weight vector geometry.

For this general case, this paper explores three options to understand neuron IO functionalities at different levels of granularity: (1) Classify neurons according to the closest prototypical case. (2) Plot the marginal distributions of the three cosine similarities. (3) Place neurons in a plot analogous to Figure 2, based on their three weight cosines.

For (1), we need two refinements. (i) We need a threshold τ for counting a cosine similarity as clearly different from zero. In this paper, we set $\tau = 0.5$, a relatively permissive cutoff that we believe gives rise to a more informative classification of neurons.

(ii) $\cos(w_{\text{in}}, w_{\text{gate}})$ may not always "match" the other two cosine similarities; e.g., the two reading weights may be orthogonal, but $w_{\text{out}} = \frac{1}{\sqrt{2}} w_{\text{gate}} + \frac{1}{\sqrt{2}} w_{\text{in}}$; then both cosine similarities are $\frac{1}{\sqrt{2}} > 0.5$. We are mainly interested in IO behavior rather than comparing the two reading weights, so we classify such cases based

[2] We prefer these terms to alternatives like *increase / reduction* because in practice output directions will not be exactly the same as the reading directions. See Section 7.4.

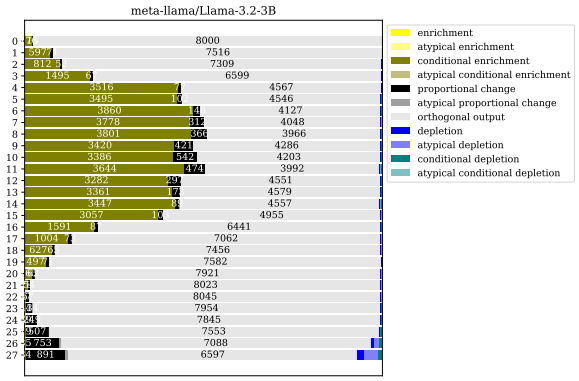


Figure 3: Distribution of neurons by layer and category.

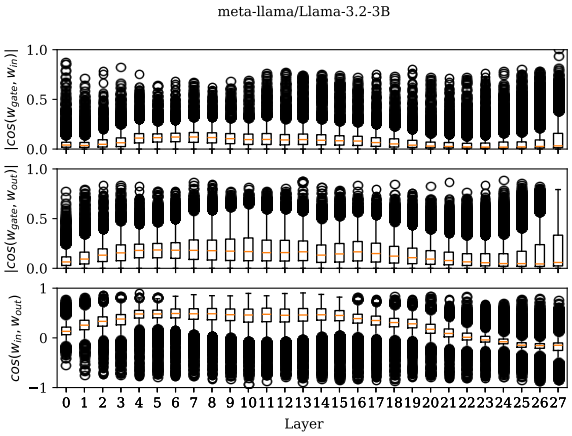


Figure 4: Boxplots for the distribution of weight cosine similarities in each layer. For $\cos(w_{\text{gate}}, w_{\text{in}})$ and $\cos(w_{\text{gate}}, w_{\text{out}})$ we show the absolute value since their sign does not carry any information on its own.

on $\cos(w_{\text{in}}, w_{\text{out}})$ and $\cos(w_{\text{gate}}, w_{\text{out}})$. To signal the "mismatch" of $\cos(w_{\text{in}}, w_{\text{gate}})$, we prepend **atypical** to the category's name. In the above example, we will speak of an atypical enrichment neuron. In Figure 2, the atypical classes share their position with typical classes, but differ in color.

Table 1 shows all atypical (and typical) classes.

5 IO functionalities by layers

We conduct our study on 12 models: Gemma-2-2B, Gemma-2-9B (Gemma, 2024), Llama-2-7B, Llama-3.1-8B, Llama-3.2-1B, Llama-3.2-3B (Touvron et al., 2023), OLMo-1B, OLMo-7B-0424 (Groeneveld et al., 2024), Mistral-7B (Jiang et al., 2023), Qwen2.5-0.5B, Qwen2.5-7B (Yang et al., 2024), Yi-6B (01.AI et al., 2025). These models use SwiGLU, except for Gemma, which uses GeGLU. For each model, we classify the MLP neurons based on the cosine similarities of the three weight vectors, as described in Section 4.

Here we describe the results for **Llama-3.2-3B**. They are representative of the general trends we observe. Section I in the appendix contains the plots for all models.

We progress from (i) the coarse-grained version of our method, with discrete classes, to (ii) the marginal distributions of each cosine similarity, to (iii) fine-grained scatter plots showing all individual neurons.

5.1 Discrete classes

Figure 3 shows IO class distribution across layers.

We see that a large proportion of neurons are input manipulators (i.e., they are not orthogonal output neurons): in the Llama model, these are 25% of all neurons, and as much as 50% in early-middle layers (layers 7–11). This highlights an advantage of our parameter-based IO classes: It is an exhaustive analysis of all neurons, and we can make non-trivial statements about a large subset of them. Other methods only assign a subset of neurons to classes; e.g., Gurnee et al. (2024)'s classification only covers 1-5% of neurons.

The majority of these input manipulators (more than 80% in Llama) belong to just one class: conditional enrichment. Across all models, conditional enrichment dominates early-middle layers. In contrast, the (relatively few) input manipulators in late layers are often proportional change neurons or depletion neurons.

The dominance of conditional enrichment neurons in early-middle layers echoes Geva et al. (2023)'s and Lad et al. (2024)'s findings that these layers perform enrichment (or feature engineering). We discuss this in Section 7.4.

These patterns hold for all models. Some other models display additional patterns, for example a large number of conditional depletion neurons in middle-late layers. See Section I.

5.2 Marginal distributions

Figure 4 shows the distribution of weight cosine similarities in each layer. In Figure 1 we also show the median of $\cos(w_{\text{in}}, w_{\text{out}})$, across all investigated models.

We already know that conditional enrichment neurons are plentiful in the early-middle layers. Correspondingly, the median value of $\cos(w_{\text{in}}, w_{\text{out}})$ peaks in these layers. Later on, it moves below zero, indicating that now the majority of neurons have negative $\cos(w_{\text{in}}, w_{\text{out}})$. Figure 1 shows that this generalizes across models.

Regarding $|\cos(w_{\text{gate}}, w_{\text{out}})|$, the median values are relatively close to zero (corresponding to conditional classes and orthogonal output). But there is a large spread in early-middle layers and in the last few layers. This seems to correspond to the proportional change neurons appearing in all of these layers, as well as depletion neurons in the last few layers.

$|\cos(w_{\text{gate}}, w_{\text{in}})|$ is mostly concentrated around zero. Thus most neurons operate on two input directions in the residual stream (not a single one), resulting in higher expressivity and more complex semantics. This is likely related to double checking; see Section 7.2.

We also notice that there are many outliers for all three cosine similarities, in almost all layers. This sug-

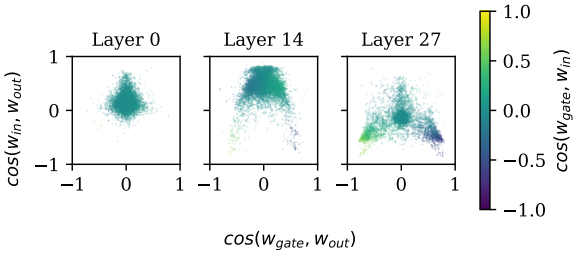


Figure 5: Fine-grained analysis of neuron IO behavior in three layers based on the configuration of their three weight vectors in parameter space. Each subplot represents a layer, each dot a neuron.

gests that a non-negligible number of neurons perform special tasks different from the "average" neuron.

5.3 Fine-grained analysis of IO behavior

We now investigate weight vector configurations in detail, as shown in Figure 5 for a few selected layers. The distribution of neurons in each layer is plotted by displaying each neuron as a point with $\cos(w_{\text{gate}}, w_{\text{out}})$ indicated on the x-axis, $\cos(w_{\text{in}}, w_{\text{out}})$ on the y-axis and $\cos(w_{\text{gate}}, w_{\text{in}})$ as its color.

This visualization reinforces three findings from Sections 5.1 and 5.2. (i) We already know that many neurons are input manipulators. Now we see that, even though there are many neurons we classified as orthogonal output, there is no cluster around the origin as we might expect. Instead, the orthogonal output neurons often belong to clusters that are centered above/below the horizontal line. This suggests that even the orthogonal output neurons perform input manipulation to some extent. (ii) We also have already observed a smooth transition from enrichment-like functionalities in early-middle layers to more depletion-like functionalities in the last few layers. We indeed see a large cluster of neurons, centered clearly above the x-axis in most layers, but moving below it in the last few layers. (iii) We also observe that the vast majority of neurons is turquoise, i.e., $\cos(w_{\text{gate}}, w_{\text{in}}) \approx 0$, confirming the finding in Section 5.2.

We also gain four new insights. (i) The *first layer* exhibits quite different patterns from model to model. (ii) In *middle layers*, all models have a big cluster related to conditional enrichment neurons, as described above. Additionally, many models have outlier "arms" from this cluster, towards the plot areas corresponding to proportional change and depletion. Other models, such as OLMo, additionally have a cluster of neurons below the x-axis, corresponding to conditional depletion neurons. (iii) Neurons with orthogonal w_{gate} and w_{in} must be within the unit disk. It is striking to see that they do not fill out this disk evenly. Instead, as already mentioned, there is a big cluster above the x-axis (close to conditional enrichment). But this cluster is not right at the border of the disk, but more inside (in particular $\cos(w_{\text{in}}, w_{\text{out}})$ is still clearly below 1). This echoes and extends Gurnee et al. (2024)'s findings that in GPT2 the IO cosine similarity is approximately bounded by ± 0.8. In other words, we almost never get the *prototypical* cases of conditional enrichment / depletion etc., as defined in Section 4. This helps us refine our notion of "input manipulators": these neurons do more than just outputting a w_{out} that is already present in the residual stream; instead, they add novel but related information. (iv) In the *last few layers* (Llama: layers 25-27), some new phenomena occur: apart from the big cluster, there is a new cluster in the bottom corners of the plot (close to depletion). Additionally, in the last layer of some models, there is a cluster of turquoise points around the upper y-axis (close to conditional enrichment).

6 Case studies

We now demonstrate how the IO perspective can complement other methods to help understand individual neurons. To this effect, we present 6 case studies for OLMo-7B, one for each discrete IO class. We restrict the search space to prediction/suppression neurons (two of the output-based *functional roles* of Gurnee et al., 2024), i.e., each of the six neurons is a prediction/suppression neuron as well as exemplifying one of our six classes. For ease of interpretability, we choose that prediction/suppression neuron of a particular IO class with the highest $\cos(w_{\text{out}}, W_U)$ kurtosis, where $W_U \in \mathbb{R}^{d_{\text{model}} \times d_{\text{vocab}}}$ denotes the unembedding matrix. (For orthogonal output we chose the clearest of all suppression neurons.) The 6 neurons are in the last layers of the model because that's where prediction/suppression neurons tend to appear.

See Section F for details on prediction/suppression, Section G for more details on these case studies, and Section H for more case studies.

6.1 Methods

We combine the IO perspective with two well-established neuron analysis methods. For each neuron, we project its weight vectors to vocabulary space with the unembedding matrix W_U and inspect high-scoring tokens. (This is analogous to (nostalgebraist, 2020) and has been done e.g. in (Geva et al., 2022; Gurnee et al., 2024; Voita et al., 2024).) Additionally, we examine examples for which the neuron is strongly activated (positive or negative) among a subset of 20M tokens from Dolma (Soldaini et al., 2024), OLMo's training set. (Activation-based analyses have been done e.g. in Geva et al., 2021; Voita et al., 2024; Gurnee et al., 2024. The size of 20M tokens follows Voita et al., 2024.)

6.2 Analysis

For many of these neurons, the largest positive activation is much larger than the largest negative one (or vice versa). Often the larger of the two is also more interpretable. In these cases we just describe the larger activation and refer to Table 4 in Section H for more details.

Enrichment neuron 28.4737 predicts *review* (and related tokens) if activated positively, which happens if *review* is already present in the residual stream. The maximally positive activations are in standard contexts that continue with *review* or similar, such as the newline after the description of an e-book (the next paragraph often is the beginning of a review).

Conditional enrichment neuron 28.9766's IO functionality concerns *well* and similar tokens. 28.9766 promotes them if activated positively, which happens when both w_{gate} and w_{in} indicate that *well* is represented in the residual stream. This is a case of double checking. The maximally positive activation in our sample occurs on ***Oh***, in a context in which *Oh, well* makes sense (and is the actual continuation).

Depletion neuron 31.9634. $-w_{\text{out}}$ of 31.9634 is closest to forms of *again*. Judging by the weights, the neuron activates positively when the residual stream contains information both for and against predicting *again*, and then depletes the *again* direction. It activates negatively when the residual stream contains the "minus *again*" direction, and then depletes that direction. Surprisingly, despite its strong negative cosine similarity ($\cos(w_{\text{gate}}, w_{\text{in}}) = -0.7164$), the neuron often activates positively. On the positive side, strong activations are often on punctuation, and the actual next token is often *meanwhile* or *instead*. The neuron may ensure only these tokens are predicted, and not the relatively similar *again*. On the negative side, the activations do not have any obvious semantic relationship to *again*. We hypothesize that sometimes the residual stream ends up near "minus *again*" for semantically unrelated reasons (there are many more possible concepts than dimensions, so the corresponding directions cannot be fully orthogonal; see Elhage et al., 2022); in these cases the neuron would reduce the unjustified presence of this "minus *again*" direction. There are also weaker negative activations when *again* is a plausible continuation, e.g., on the token ***once***. In these cases, *again* is already weakly present in the residual stream before the last MLP. Accordingly, $Swish(w_{\text{gate}} \cdot x_{\text{norm}})$ is weakly negative (but distinct from 0), and $w_{\text{in}} \cdot x_{\text{norm}} > 0$, which leads to a negative activation and thus reinforces *again*.

Conditional depletion neuron 29.10900. Gate and linear input weight vectors act as two independent ways of checking that *these* is not present in the residual stream (i.e., a case of double checking). At the same time, they check for predictions like *today, nowadays*. When such predictions are present, the neuron promotes *these*. This is a plausible choice in these cases because of the expression *these days*. An example is *social media tools change and come and go at the drop of a* ***hat***. (This sentence talks about a characteristic of current times, so *these days* would indeed be a plausible continuation.)

Proportional change neuron 30.10972 predicts the token *when* if activated negatively. This happens if *when* is absent from the residual stream (gate condition) and is proportional to the presence of time-related tokens ($-w_{\text{in}}$). An example for a large negative activation is *puts you on multiple webpages* ***at***.[3] Conversely, if *when* is absent, and time-related tokens are absent too, the neuron activates positively and suppresses *when* further.

Orthogonal output neuron 29.4180 predicts *there* (positive activation) if the residual stream contains a component that we interpret as "complement of place expected" (e.g., *here*, *therein*). Both w_{gate} and w_{in} check for (different aspects of) this component being present, another case of double checking. The largest positive activation is on *here* ***or***.

Overall, these neurons all promote a specific set of tokens (we chose them that way), but under very different circumstances. The (conditional) enrichment neurons are the most straightforward to interpret, because their input and output clearly correspond to the same concept. In contrast, depletion neurons inherently involve (an apparent) conflict between the intermediate model prediction and what the neuron promotes.

7 Discussion

7.1 Variation across models

Our work on gated activation functions questions the generality of previous findings (Voita et al., 2024; Gurnee et al., 2024) on non-gated activation functions. Specifically, we saw in Section 5 that (conditional) depletion neurons appear mostly in later layers. On the other hand, Gurnee et al. (2024) find (for GPT-2 (Radford et al., 2019), with activation GeLU) that what we call depletion neurons mostly appear in earlier layers. Similarly, Voita et al. (2024) find (for OPT (Zhang et al., 2022), with activation ReLU) that some neurons in early layers detect specific tokens and then suppress them. (Their analysis is not weight-based, so these may or may not be depletion neurons in our weight-based sense.)

This confirms the importance of our work for models with gated activation functions: their internal structure is quite different from older models with GeLU or ReLU.

Despite minor differences (especially in the first layer), our results across gated activation models are remarkably consistent. Most importantly, all of them are dominated by conditional enrichment neurons in early-middle layers and all of them tend towards depletion in the very last layers.

7.2 Double checking

Our case studies suggest that conditional enrichment or conditional depletion neurons often behave in a way

[3] The actual sentence ends with *as soon as* and comes from a now-dead webpage. We also found one occurrence of *at when* in what seems to be a paraphrase of the same text, on https://www.docdroid.net/RgxdG5s/fantastic-tips-for-bloggers-of-all-amountsoystcpdf-pdf . We suspect that both texts are machine-generated paraphrases of an original text containing *at once* (*when* and *as soon as* can be synonyms of *once* in other contexts), and that the model has (also) seen a paraphrased version with *at when*. In fact many of the largest negative activations are on *at* in contexts calling for *at once*.

analogous to their unconditional counterparts. One reason is that our threshold for distinguishing conditional and unconditional classes is somewhat arbitrary.

These and other neurons (for example, proportional change neurons like 25.8607, Section H) display a phenomenon we called double checking: They use two quite different reading weight vectors to check for a single concept.

Double checking is rooted in the following geometric fact: Two vectors w_1, w_2 (w_{gate} and w_{in} in our case) can be orthogonal to each other but still have a high similarity to a third vector u (e.g., a token unembedding). Example: $w_1 = (1, 0), w_2 = (0, 1), u = (1, 1)$. Here, w_1, w_2 are orthogonal, but u has a cosine of $\frac{\sqrt{2}}{2} \approx 0.7$ to both.

Double checking is useful because it shrinks the region in model space that activates the neuron positively. If (say) $w_{\text{in}} = w_{\text{gate}} = (1, 0)$, the neuron activates whenever the (normalized) residual input x satisfies $x \cdot (1, 0) > 0$; this happens on the whole half-space $x_1 > 0$. If however $w_{\text{gate}} = (1, 0)$ and $w_{\text{in}} = (0, 1)$, the neuron activates positively only in the first quadrant $(x_1, x_2 > 0)$.

This behavior thus enables more precise concept detection. This may explain why conditional neurons are more frequent than their unconditional counterparts.

7.3 Stages of inference

We saw in Section 5 that different layers are dominated by different IO functionalities. This leads to a follow-up question: Why does the model use these specific IO functionalities in these specific layers? In particular: Why are there so many conditional enrichment neurons in early-middle layers? And what is the role of (conditional) depletion neurons in later layers? We hypothesize that different IO classes might be responsible for different *stages of inference* (Lad et al., 2024), as described in the following subsections. In future work, we plan to test this hypothesis using ablation experiments.

7.4 Enrichment

We saw in Section 5 that there often is positive similarity between reading and writing weights of neurons, especially with conditional enrichment neurons in early-middle layers.

These neurons seem a good fit for the *feature engineering* stage (Lad et al., 2024), corresponding to enrichment as defined by Geva et al. (2023). Indeed, they output a direction similar to the one they detect, which could correspond to related concepts. Geva et al.'s (2023) description of enrichment precisely involves writing related concepts to the residual stream.

In later layers, the (conditional) enrichment neurons we investigated in our case studies (Section 6) have an output that is semantically identical to the input. Thus they seem to reinforce existing predictions.

In general, we use the term *enrichment* because the output weight is never mathematically identical to one of the reading weights. But depending on the analysis of a particular neuron (e.g., by way of a case study), magnification (no change) or enrichment (e.g., change *Ireland* in the input to *Dublin* in the output) may be the more intuitive human interpretation.

7.5 Depletion

We saw in Section 5 that depletion neurons appear mostly in the last few layers, and conditional depletion neurons appear in later-middle layers (if at all).

These neurons reduce the presence of the directions they detect. Therefore they seem a good fit for the *residual sharpening* stage – getting rid of attributes that are not directly needed for next token prediction.

We found depletion neurons more difficult to interpret than enrichment neurons. Most notably, neuron 31.9634 was a complex case in that we found contexts in which a weak positive presence of *again* led to an enrichment-like functionality (see Section 6.2). This mechanism involves a negative value of Swish. Previous authors (Gurnee et al., 2023) often assumed that GELU (or equivalently, Swish) is "essentially the same activation as a ReLU", and said they "would be particularly excited to see future work exhibiting [...] case studies" of mechanisms involving negative values of such an activation function. To our knowledge, we show for the first time that negative values of Swish can play a crucial role in how transformers function.

Still, all neurons we investigated do deplete input directions from the output even if they do not do so in all contexts. We plan to further elucidate the intuitive role depletion plays in follow-up work.

8 Conclusion

We explored the IO perspective for investigating gated neurons in LLMs. Our method complements prior interpretability approaches and provides new insights into the inner workings of LLMs.

We observed that a large share of neurons exhibit non-trivial IO interactions. The concrete IO functionalities differ from layer to layer, which is probably related to different stages of inference. In particular, early-middle layers are dominated by conditional enrichment neurons, which may be responsible for representation enrichment.

We plan to further develop this new perspective in future work. In particular, we will do ablation experiments to conclusively show if, as we hypothesized, the conditional enrichment neurons in early-middle layers are responsible for representation enrichment and the depletion neurons in the last few layers contribute to residual sharpening. We also plan to investigate the evolution of IO functionalities during model training. Finally, we would like to go beyond the analysis of single neurons and address the question of how neurons work together within and across IO classes.

Limitations

This paper focuses on a *parameter-based* interpretation of *single neurons*. This has the advantage of being simple and efficient, but is also inherently limited in scope. Accordingly, our method is not designed to replace other neuron analysis methods, but to complement them.

The mathematical similarities of weights are insightful, but they should not be taken as one-to-one representations of semantic similarity: We find cases in which close-to-orthogonal vectors represent very similar concepts (Section 7.2), and cases in which mathematically similar vectors represent related but non-identical concepts (Section 7.3).

Our case studies of individual neurons can be accused of cherry-picking: we picked neurons that we expected to be interpretable, all of which occur on the last few layers. Therefore our interpretations may not carry over to less interpretable (e.g. polysemantic) neurons, or to neurons in earlier layers.

Finally, we provide only possible interpretations of the phenomena we observe, and do not claim them to be definitive explanations.

References

01.AI, :, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yanpeng Li, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. 2025. Yi: Open foundation models by 01.ai. *Preprint*, arXiv:2403.04652.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits.

Amit Elhelo and Mor Geva. 2024. Inferring functionality of attention heads from their parameters. *Preprint*, arXiv:2412.11965.

Team Gemma. 2024. Gemma.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, Singapore. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, William Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah Smith, and Hannaneh Hajishirzi. 2024. OLMo: Accelerating the science of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809, Bangkok, Thailand. Association for Computational Linguistics.

Wes Gurnee, Theo Horsley, Zifan Carl Guo, Tara Rezaei Kheirkhah, Qinyi Sun, Will Hathaway, Neel Nanda, and Dimitris Bertsimas. 2024. Universal neurons in gpt2 language models.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Vedang Lad, Wes Gurnee, and Max Tegmark. 2024. The remarkable robustness of llms: Stages of inference? *Preprint*, arXiv:2406.19384.

Joseph Miller and Clement Neo. 2023. We found an neuron in gpt-2.

Beren Millidge and Sid Black. 2022. The singular value decompositions of transformer weight matrices are highly interpretable.

Ari S. Morcos, David G.T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. 2018. On the importance of single directions for generalization.

Neel Nanda and Joseph Bloom. 2022. Transformerlens. `https://github.com/TransformerLensOrg/TransformerLens`.

Jingcheng Niu, Andrew Liu, Zining Zu, and Gerald Penn. 2024. What does the knowledge neuron thesis have to do with knowledge?

nostalgebraist. 2020. Interpreting gpt: The logit lens.

Kiho Park, Yo Joong Choe, and Victor Veitch. 2024. The linear representation hypothesis and the geometry of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 39643–39666. PMLR.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Cody Rushing and Neel Nanda. 2024. Explorations of self-repair in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 42836–42855. PMLR.

Noam Shazeer. 2020. Glu variants improve transformer.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Evan Walsh, Luke Zettlemoyer, Noah Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. 2024. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788, Bangkok, Thailand. Association for Computational Linguistics.

Alessandro Stolfo, Ben Wu, Wes Gurnee, Yonatan Belinkov, Xingyi Song, Mrinmaya Sachan, and Neel Nanda. 2024. Confidence regulation neurons in language models.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*.

Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. 2024. Neurons in large language models: Dead, n-gram, positional. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1288–1301, Bangkok, Thailand. Association for Computational Linguistics.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

A Overview of the appendix

Section B: Software and data.

Section C: Impact statement.

Section D: "Responsible NLP" statements.

Section E: Visualization of a SwiGLU neuron (Section 3).

Section F: IO classes vs. Gurnee et al.'s (2024) *functional roles*. Used in Section 6.

Section G: Details on case studies (Section 6).

Section H: More case studies (complementing Section 6).

Section I: Results across models (complementing Section 5).

We chose to put the last section at the end because it is very long and would otherwise disrupt reading of the other sections.

B Software and data

This review version is accompanied by zip archives containing software and data, respectively. See the readme file for detailed documentation.

We plan to release the software under a permissive license such as Apache 2.0.

The data archive currently contains only the visualizations of max/min activations for the neuron case studies in Section 6. Everything else can be quickly reproduced, and the plots are included in this paper. We plan to release these visualizations under the Apache 2.0 license (they contain text from Dolma, which is under the same license).

C Impact statement

This paper presents work whose goal is to advance the field of machine learning interpretability. The underlying assumption of the field is that models have underlying structure (are not just an inscrutable mess) and that discovering this structure will have several benefits. First, ideally, any scientific field should have a deep understanding of the models it uses; results that are obtained using blackbox models are hard to understand, replicate and generalize. Second, once we understand our models better, we will be better able to address failure modes. For example, once we understand how unaligned behavior like bias and hallucinations comes about, it will be easier to address them, e.g., by changing the model architecture. Third, interpretability can support explainability. If we understand how a recommendation or answer came about, we can better assess its validity.

D "Responsible NLP" statements

D.1 Models and data

Gemma. To download the model one needs to explicitly accept the terms of use. NLP research is explicitly listed as an intended usage. Primarily English and code (Gemma, 2024).

Llama. Inference code and weights under an ad hoc license. There is also an "Acceptable Use Policy". Our work is well within those terms. Languages mostly include English and programming languages, but also Wikipedia dumps from "bg, ca, cs, da, de, en, es, fr, hr, hu, it, nl, pl, pt, ro, ru, sl, sr, sv, uk" (Touvron et al., 2023).

OLMo and Dolma. Training and inference code, weights (OLMo), and data (Dolma) under Apache 2.0 license. "The Science of Language Models" is explicitly mentioned as an intended use case. Dolma is quality-filtered and designed to contain only English and programming languages (though we came across some French sentences as well, see Table 4) (Groeneveld et al., 2024; Soldaini et al., 2024).

Mistral. Inference code and weights are released under the Apache 2.0 license, but accessing them requires accepting the terms. Languages are not explicitly mentioned in the paper, but clearly include English and code (Jiang et al., 2023).

Qwen. Inference code and weights under Apache 2.0 license. Supports "over 29 languages, including Chinese, English, French, Spanish, Portuguese, German, Italian, Russian, Japanese, Korean, Vietnamese, Thai, Arabic, and more" (Yang et al., 2024).

Yi. Inference code and weights under Apache 2.0 license. Trained on English and Chinese (01.AI et al., 2025).

D.2 Computational experiments

All our experiments can be run on a single NVIDIA RTX A6000 (48GB). The main analysis, computing the weight cosines, needs less than a minute per model. The most expensive part was the activation-based analysis in Section 6: We needed a single run of ≈ 25 h to store the max/min activating examples for all neurons, and then ≈ 45 s per neuron (≈ 5 min) to recompute its activations on the relevant texts and visualize them.

We use TransformerLens (Nanda and Bloom, 2022). A colleague kindly provided us with a version that also supports OLMo.

E More on SwiGLU

Figure 6 visualizes a SwiGLU neuron (described in Section 3).

F IO classes vs. functional roles

We compare our results with those of another classification scheme we mentioned in Section 2: the *functional roles* defined by Gurnee et al. (2024). See Section F.3 for the results.

F.1 Definition of functional roles

The definition of functional roles is based exclusively on the neuron's output weight w_{out}. Most of the roles are defined by their output token distribution, i.e., properties of the distribution $\cos(w_{\text{out}}, W_U) = \left(\frac{w_{\text{out}} \cdot W_U[:,1]}{\|w_{\text{out}}\| \|W_U[:,1]\|}, \ldots, \frac{w_{\text{out}} \cdot W_U[:,d_{\text{vocab}}]}{\|w_{\text{out}}\| \|W_U[:,d_{\text{vocab}}]\|}\right) \in [-1,1]^{d_{\text{vocab}}}$, the cosine of the product of output weight vector and unembedding matrix.

Functional roles are defined as follows. **Prediction** and **suppression** neurons have a $\cos(w_{\text{out}}, W_U)$ with high *kurtosis* (meaning there are many outliers) and a high *skew* in absolute value (meaning the outliers tend to be only on one side). Positive skew corresponds to predicting a subset of tokens, negative skew to suppressing it. **Partition** neurons have a distribution $\cos(w_{\text{out}}, W_U)$ with high *variance*. This often corresponds to two sets of output tokens, one that is promoted and one that is suppressed. In **entropy** neurons (examined in more detail by Stolfo et al. (2024), w_{out} lies in a direction that does not correspond to any output tokens. Mathematically, a high proportion of the norm of w_{out} is in W_U's effective null space, i.e., it corresponds to singular vectors of W_U whose corresponding singular values are

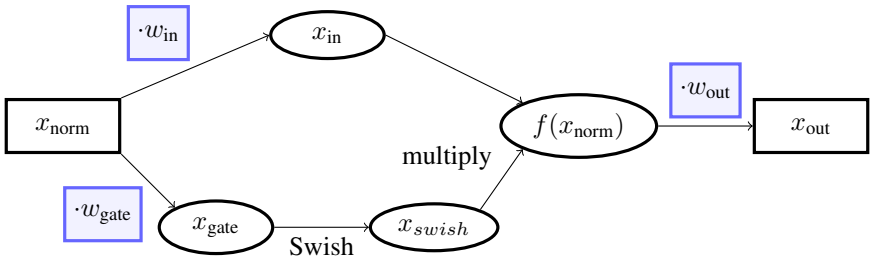


Figure 6: Visualization of the SwiGLU activation function for a single neuron. Boxes represent vectors, ellipses represent scalars.

close to zero. Entropy neurons increase or decrease the presence of such directions. This changes the norm of the residual stream, but leaves the token ranking more or less untouched. Because a final LayerNorm is applied before W_U, this indirectly affects the logits of all tokens: the output token probabilities become more evenly distributed (higher entropy), or less so (lower entropy). **Attention (de)activation** neurons (de)activate an attention head by having it put less (or more) of its attention on the BOS token. (The effect of a head attending only to BOS is negligible.) Consider an attention head with query matrix $W_Q \in \mathbb{R}^{d_{\text{model}} \times d_{head}} = \mathbb{R}^{4096 \times 128}$ and BOS key vector $k_{BOS} \in \mathbb{R}^{d_{head}}$. Attention (de)activation neurons for this head are those with a high positive or negative score $w_{\text{out}} W_Q k_{BOS}$.

All of these definitions require a threshold and/or some adaptation to gated activation functions. We describe our approach in Section F.2.

F.2 Adapting the definitions

The *functional role* definitions require a threshold and/or some adaptation to gated activation functions. We proceed as follows:

- We set the number of *partition* neurons to be 1000, which gives a variance of 0.0007 as a threshold.
- Preliminary experiments show that (absolute) skew and kurtosis are highly correlated in practice, so we decide to focus on kurtosis to find prediction / suppression neurons. We then choose a kurtosis threshold for *prediction/suppression*, such that the *prediction/suppression* class is disjoint from *partition*. This gives a (very high) excess kurtosis of 230.9736.
- *Entropy*: Following Stolfo et al. (2024), we focus on the last layer, and we define the null space of W_U as the subspace of model space spanned by its last 40 singular vectors. We find that two neurons have a particularly high proportion of their norm in this null space, and define these as entropy neurons.
- *Attention (de)activation*: To ensure comparability across heads, we normalize w_{out} and $W_Q k_{BOS}$. Thus the scores can be intuitively understood as cosine similarities between these two vectors. We choose $\pm\frac{\sqrt{2}}{2}$ as a cutoff. We keep only those neurons that we did not already classify as partition or prediction/suppression.
- In our case the neuron can be activated positively or negatively, so we cannot distinguish prediction from suppression *a priori*. Instead, we automatically distinguish *prediction* and *suppression* from each other by the sign of $\cos(w_{\text{in}}, w_{\text{gate}}) \cdot \text{skew}(\cos(w_{\text{out}}, W_U))$ (as opposed to just the sign of the skew). The quantity $\cos(w_{\text{in}}, w_{\text{gate}})$ indicates the typical sign of the activation *a priori*. Even though this is not very trustworthy it gives some interesting results.
- The same problem occurs for the distinction of attention activation and deactivation. As before, we multiply the original quantity $w_{\text{out}} W_Q k_{BOS}$ by $\cos(w_{\text{in}}, w_{\text{gate}})$ and only then look at the sign. Note that here a positive sign means high attention on BOS, hence attention *de*activation. It turns out that all relevant neurons are attention *de*activation according to this metric.

F.3 Results

The contingency matrix in Table 2 is a systematic comparison of our IO classes with Gurnee et al. (2024)'s functional roles.

We first see again that Gurnee et al. (2024) assign a functional role to only a small proportion of all neurons. 349,521 of 352,256 neurons remain unclassified. In contrast, our IO classes are exhaustive and robustly identify functionalities like conditional depletion and enrichment that are explanatory for how transformers process language.

We find that prediction neurons, suppression neurons and (less consistently) partition neurons mostly occur in the final layers, replicating Gurnee et al. (2024)'s findings.

Most of these neurons are orthogonal output or proportional change. This is not unexpected, as these are some of the largest classes. Conversely, however, a majority of the (relatively few) depletion neurons have prediction or partition as functional role.

The only two entropy neurons in OLMo-7B-0424 occur in the last layer and are conditional depletion neurons.

					attention			
	prediction	suppression	partition	entropy	deactivation		other	total
depletion	73	0	51	0	2	14	117	243
at. depletion	114	0	61	0	0	3	429	604
c. depletion	68	1	24	2	0		12,344	12,439
at. c. depletion	19	0	13	0	0		12	44
orthogonal output	826	203	516	0	0		190,832	192,377
proportional change	111	206	139	0	0	1	23,358	23,814
at. proportional change	25	0	16	0	2		85	128
c. enrichment	48	0	179	0	0	1	121,446	121,673
at. c. enrichment	14	0	0	0	0		660	674
enrichment	6	0	0	0	0		18	24
at. enrichment	15	0	1	0	0		220	236
total	1,319	410	1,000	2	4	21	349,521	352,256

Table 2: Contingency table of IO classes (rows) vs Gurnee et al. (2024)'s functional roles (columns) for OLMo-7B-0424. c = conditional. at = atypical. Cutoffs for prediction/suppression and partition were chosen as described in Section F.2. Many neurons with high attention deactivation score are also partition neurons; the left column unter "attention deactivation" counts only those that are not. OLMo-7B-0424 has no attention activation neurons with high enough score.

Neuron	IO category	$\cos(w_{\text{gate}}, w_{\text{in}})$	$\cos(w_{\text{gate}}, w_{\text{out}})$	$\cos(w_{\text{in}}, w_{\text{out}})$
28.4737	enrichment	0.5290	0.5048	0.7060
28.9766	conditional enrichment	0.4764	0.4119	0.5982
31.9634	depletion	-0.7164	0.7218	-0.8542
29.10900	conditional depletion	0.4988	-0.4992	-0.5775
30.10972	proportional change	-0.4543	0.5814	-0.4182
29.4180	orthogonal output	-0.0272	-0.4057	0.0669

Table 3: Overview of prediction/suppression neurons chosen for case studies in Section 6

G Details on case studies

See Tables 3 and 4 for more details on the case studies of Section 6.

H More case studies

These are various neurons that popped out to us as possibly interesting, for not very systematic reasons, for example because they strongly activated on a specific named entity. All of them are in OLMo-7B. We present them by IO class. For most of these case studies we did only a quick and dirty weight-based analysis. In some cases we also tried W_E (input embeddings) instead of W_U (unembeddings) for the logit-lens style analysis.

H.1 Conditional enrichment neurons

0.1480: $w_{\text{gate}}, -w_{\text{in}}, -w_{\text{out}}$ all have tokens similar to *box* (when using W_E). Activates on *Xbox*.

4.1940: *country* appears in w_{in} among many other things. When using W_E, *Philippines* and *Manila* appear in w_{out}. Activates on *Philippines*.

4.3720: gate seems country/government related. When using W_E, we find $w_{\text{out}}, w_{\text{gate}}$ contain some country names. Activates on *Denmark*.

4.4801: *Muhammad* appears in the gate vector. Activates on *Muhammad*.

4.5772: predicts *ian* as in *Egyptian*. When using W_E, all three weight vectors contain *Egypt*. Activates on *Egypt*.

4.6517 has a very Ireland (or Celtic nations) related gate vector. The interpretations of the other two weights are less obvious, but *Irish* and *Dublin* appear in w_{in} among many other things, and *UK* and *London* appear in $-w_{\text{out}}$ (Ireland is emphatically *not* in the UK!) When using W_E, *Ireland* appears among the top tokens of all three weight vectors. Activates on *Ireland*.

4.6799: When using W_E, *Vietnam* is among the tokens corresponding to $-w_{\text{out}}$. Activates on *Vietnam*

4.7667: all three weights related to consoles in different ways. Activates on *Xbox*

4.9983: w_{out} is related to electronic devices, w_{in} either electronic devices or sports (surfing may belong to both), w_{gate} is also mostly related to electronic devices. When using W_E, we find w_{out} contains *iPhone* as a top token. Activates on *iPhone*.

4.10859: When using W_E, we find $w_{\text{gate}}, w_{\text{out}}$ include *Thailand* as a top token, w_{out} additionally *Buddha, Buddhist*. Activates on *Thailand*.

4.10882: When using W_E, we find $-w_{\text{out}}$ contains *Italy*, $-w_{\text{in}}, w_{\text{gate}}$ additionally contain *Rome*. Activates on *Italy*.

4.10995: *Boston* appears in gate and *Massachusetts* in $-w_{\text{in}}$. When using W_E, we find $-w_{\text{out}}, w_{\text{gate}}$ contain *Massachusetts* and *Boston*, $-w_{\text{in}}$ contains *Boston*. Activates on *Massachusetts*.

22.2589: w_{gate} and $-w_{\text{in}}$ recognize tokens like *Islam, Muhammad* and others related to the Arabo-Islamic world. The same goes for $-w_{\text{out}}$ (as it is similar to w_{in}).

Neuron, IO class	w_{gate}	w_{in}	w_{out}	Top activations
28.4737 enrichment	$\approx w_{\text{out}}$	$\approx w_{\text{out}}$	pos: *review* *Review*	pos (13.75): *Download EBOOK [...] Description of the book [...]* ***\n*** *-> Reviews* neg (-2.25): *The answer's at the bottom of* ***this*** *-> post*
28.9766 conditional enrichment	pos: *well* *well* neg: *far* *high*	$\approx w_{\text{out}}$	pos: *well* *well*	pos (18.63): *Could have saved myself some time.* ***Oh*** *-> , well* neg (-3.66): *Seek to understand them* ***more*** *-> fully*
31.9634 depletion	$\approx w_{\text{out}}$	$\approx -w_{\text{out}}$	neg: *again* *Again*	pos (5.12): *jumping off the roof of his Los Angeles apartment building.* ***->*** *Meanwhile* neg (-3.48): *the areas of the doorjamb where the* ***door*** *-> often*
29.10900 conditional depletion	pos: *today* *nowadays* neg: *these* *these*	$\approx -w_{\text{out}}$	pos: *these* *These*	pos (12.79): *social media tools change and come and go at the drop of a* ***hat*** *-> .* neg (-2.18): *la couleur de sa robe* ***et*** *-> le*
30.10972 proportional change	$\approx w_{\text{out}}$	pos: *when* *when* neg: *timing* *dates*	neg: *when* *when*	pos (2.67): *Take pleasure in the rest of the new year.* ***->*** *You* neg (-6.14): *puts you on multiple webpages* ***at*** *-> as soon as*
29.4180 orthogonal output	pos: *here* *therein* neg: *there* *we*	pos: *here* *in* neg: ?	pos: *there* *there*	pos (14.41): *here* ***or*** *-> there* neg (-2.31): *without any consideration being issued or paid* ***there*** *-> for*

Table 4: Description of the weight vectors of the selected neurons, by top tokens or similarity to w_{out}. The question mark, ?, signals unknown unicode characters. The last column presents the (shortened) text samples on which the respective neuron activates most strongly (positively or negatively).

Activates on *Muhammad*.

24.4880: For all three weight vectors the first four tokens (but not more) are Philippine-related (even though the gate vector is actually not very similar to the others). The gate vector also reacts to other geographical names, which *may* have in common that they are associated with non-"white" (Black, Asian or Latin) people in the US sense (*Singapore, Malaysian, Nigerian, Seoul, Pacific, Kerala, Bangkok*, but also *(Los) Angeles* and *Bronx*). Activates on *Philippines*.

24.6771: $w_{\text{gate}}, -w_{\text{in}}, -w_{\text{out}}$ all correspond to capitalized first names. Activates on *Muhammad*.

25.2723: Some tokens associated with w_{in} and w_{out} are possible completions for *th* (*th-ousand, th-ought, th-orn*. When using W_E, in all three weights there are a few *th* tokens, but also with *ph* and similar. Activates on *Thailand*.

25.10496: $-w_{\text{in}}, -w_{\text{out}}$ correspond to tokens starting with *v* (upper or lower case, with or without preceding space). w_{gate} on the other hand seems to react to appropriate endings for tokens starting in *v*: *vol-atility, v-antage, v-intage, vel-ocity, V-ancouver*. When using W_E, we also find all three weight-vectors are very *v*-heavy. Activates on *Vietnam*.

H.2 Depletion neurons

30.9996: Downgrades weird tokens if present / promotes frequent English stopwords if absent. Also an attention deactivation neuron for 15 heads in layer 31.

H.3 Proportional change neurons

25.7032: Some tokens associated with w_{gate} and w_{out} are possible completions for *x* or *ex* (*X-avier, x-yz, ex-cel, ex-ercise*. When using W_E, both *x* and *box* (with variants) appear in all three weight vectors. Activates on *Xbox*.

25.8607: All three vectors correspond to tokens related to cities. Moreover, $-w_{\text{out}}$ seems to correspond to non-city places, such as national governments or villages. w_{in} is actually not that similar to $w_{\text{gate}}, w_{\text{out}}$ (in terms of cosine similarities), but all three correspond to city-related tokens. When using W_E, in all three weights there are a few city-related tokens. Activates on *Paris*. We may think of the two input directions as two largely independent ways of checking that "it's about a city" (this is a recurring phenomenon that we describe in Section 7.2). When the gate activates but the linear input does not confirm it's about a city, the output promotes closely related but non-city interpretations (for example *Paris* actually refers to the French government in some contexts).

29.8118: Partition neuron, highest variance of all proportional change neurons. Also an attention deactivation neuron for 4 heads (0,2,11,15) in layer 30.

31.5490: Activates on *Muhammad*. w_{gate} reacts to various Asian names and Asian-sounding subwords, w_{in} to surnames as opposed to other English words starting with space and uppercase letter. w_{out} corresponds to more Asian stuff (mostly subwords) as opposed to English surnames.

31.6275: Mostly promotes two-letter tokens (no preceding space, typically uppercase). $-w_{\text{in}}$ typically lowercase single letters. $-w_{\text{gate}}$ mostly lowercase two-letter tokens. "If no lowercase two-letter tokens, promote uppercase two-letter tokens proportionally to absence of lowercase single letters" ?

31.8342: This is an *-ot-* neuron: w_{gate} and w_{out} correspond to *-o(t)-* suffixes, $-w_{\text{in}}$ to various *-ot-* stuff. Judging by the weight similarities, we expect that w_{out} is typically activated negatively: downgrade *-o(t)-* suffixes if present in the residual stream. Activates on *Egypt*.

H.4 Orthogonal output neurons

0.1758: When using W_E, all three weight vectors' top tokens are famous web sites, including *YouTube*. Activates on *YouTube*.

0.3338: When using W_E, we find especially w_{gate} and $-w_{\text{in}}$, but also $-w_{\text{out}}$ are similar to smartphone-related tokens. Activates on *iPhone*.

0.3872: When using W_E, we find especially w_{gate}, but also $-w_{\text{in}}$ and $-w_{\text{out}}$ correspond to city names. Activates on *Paris*.

0.7829: When using W_E, we find $w_{\text{in}}, w_{\text{out}}$ and to a lesser extent w_{gate} correspond in large part to software names. Activates on *iTunes*.

0.7966: When using W_E, the weight vectors mostly correspond to tokens starting with *th*. Activates on *Thor*.

29.2568: w_{out} Asian (Thai?) sounding syllables vs. (Asian) geographic names in English and other stuff; w_{in} reacts to Thailand and Asian (geography) stuff as opposed to (mostly) US stuff; w_{gate} pretty much the same. Activates on *Thailand*.

29.3327: w_{gate} mostly reacts to city names (*Paris* being the most important one), $-w_{\text{in}}$ countries and cities, especially in continental Europe (*France* and *Paris* on top) as opposed to stuff related to the former British Empire. Relevant is $-w_{\text{out}}$ which corresponds to pieces of geographical names and especially rivers in France (*Se-ine, Rh-one / Rh-ine, Mar-ne, Mos-elle... Normandie, Nancy, commun...*). w_{gate} and $-w_{\text{in}}$ also react to *river(s)*. Activates on *Paris*.

29.4101: w_{gate} and w_{in} react to *YouTube* (top token!), w_{out} downgrades it (almost bottom token) and promotes *subscrib*, views, channels* etc. Activates on *YouTube*.

29.6417: Downgrades *recording* and similar. w_{gate} and w_{in} are also similar and involve *iTunes*. Activates on *iTunes*.

29.9734: w_{gate} reacts to the East in a broad sense as opposed to the West (*Iran, Kaz-akhstan, Kash-mir, Ukraine...*), w_{in} mostly to male first names without preceding space. w_{out} seems to produce word pieces that could begin a foreign name. Activates on *Muhammad*.

30.2667: w_{gate} reacts to suffixes (for adjectives derived from place names) like *en, ian, ians*, basically the same for w_{in} and w_{out}. Activates on *Muhammad*.

30.3143: w_{gate} reacts to words related to entities that are authoritative for various reasons (*officials, au-*

thorities, according, researchers, spokesman, investigators...). $-w_{\text{in}}$ reacts to uncertainty (*reportedly, according... allegedly... accused*). $-w_{\text{out}}$ is again *police, authorities, officials, court* but with no preceding space. Activates on *Philippines*. What authorities and uncertainty have to do with the Philippines is unclear.

30.3883: w_{gate} and $-w_{\text{in}}$ react to *Virginia* and *Afghanistan*, among others (in the case of w_{gate}: as opposed to other geographical names with no preceding space associated with the South and the sea); $-w_{\text{out}}$ is activated and promotes all variants of *af* (and *ghan*) but downgrades Virginia etc. Activates on *Afghanistan*.

30.4577: Seems to be related to rugby: w_{gate} and slightly less obviously w_{in} react to rugby-related tokens (*midfielder, quarterback...*); w_{out} promotes different tokens that upon reflection could be related to rugby as well. Activates on *Ireland*.

30.5372: Promotes *natural* and related, downgrades *inst* tokens. w_{in} reacts to *wildlife* etc. as opposed to *institute* etc, w_{gate} reacts to *institute* as opposed to *natural*. Activates on *Massachusetts* (in which situation it promotes *Institute*, which makes sense because of MIT).

30.8535: $-w_{\text{out}}$ is *one* in all variants, w_{gate} too, w_{in} splits *one, ones* and the equivalent Chinese characters, on the positive side, from *One, 1, ONE* on the negative side (and many other things on both sides). Activates on *Xbox*. Presumably this happens because *One* is a possible prediction (*Xbox One*), and presumably the output reinforces that.

31.2135: orthogonal output, on the conditional enrichment side (weak conditional enrichment, one of the neurons on the vertical axis). w_{gate} reacts to single letters or symbols as opposed to some English content words without preceding space; w_{in} and w_{out} mostly Chinese or Japanese characters as opposed to some Latin diacritics and other weird stuff. Language choice? "If it's not English and single letters are floating around, make sure to choose the right language / character set."

31.10424: $w_{\text{gate}}, -w_{\text{in}}, w_{\text{out}}$ correspond to *score* in the top tokens, which is downgraded if present. Activates on *Paris*. No idea what's happening here.

I Results across models

These final figures show our analyses of IO functionalities by layer (Section 5) for all the models we investigated.

We note a few additional patterns that appear only in some of these models:

- In Yi and the OLMo models, the prevalence of conditional enrichment neurons starts even earlier, at the very first layer. A particularly interesting example is Yi: In layer 0 an enormous 68% of all neurons are conditional enrichment, then almost none, then there is a second wave around layers 11-17 (out of 32) which have around 25% of conditional enrichment neurons each.

- In some models, especially the OLMo ones, there is a non-negligible number of conditional depletion neurons. They tend to appear in middle-to-late layers, shortly after the conditional enrichment wave. The clearest example is OLMo-1B, with a peak of 1418 conditional depletion neurons out of 8192 (17%) in layer 9 out of 16.

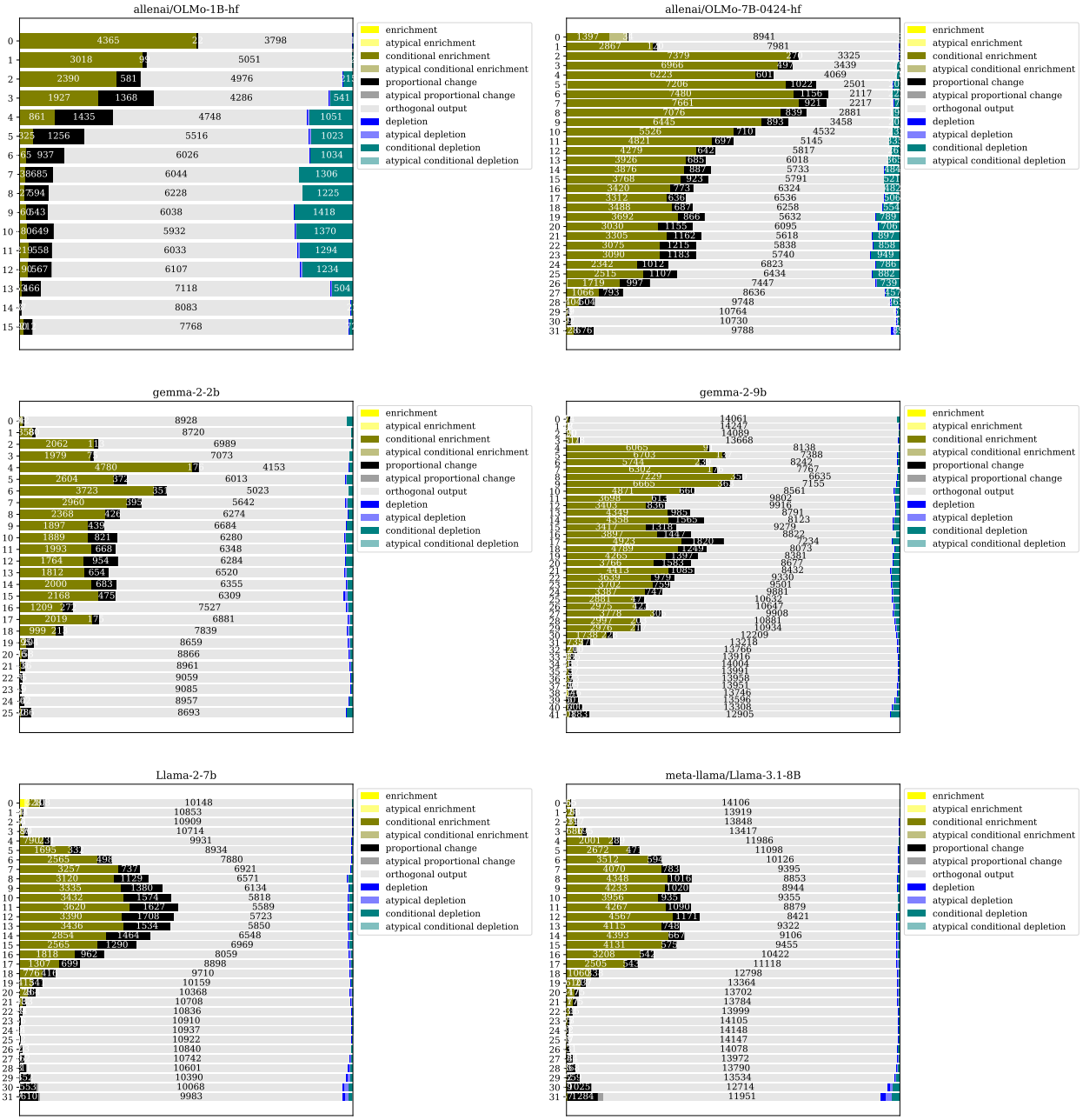


Figure 7: Distribution of neurons by layer and category for a range of models

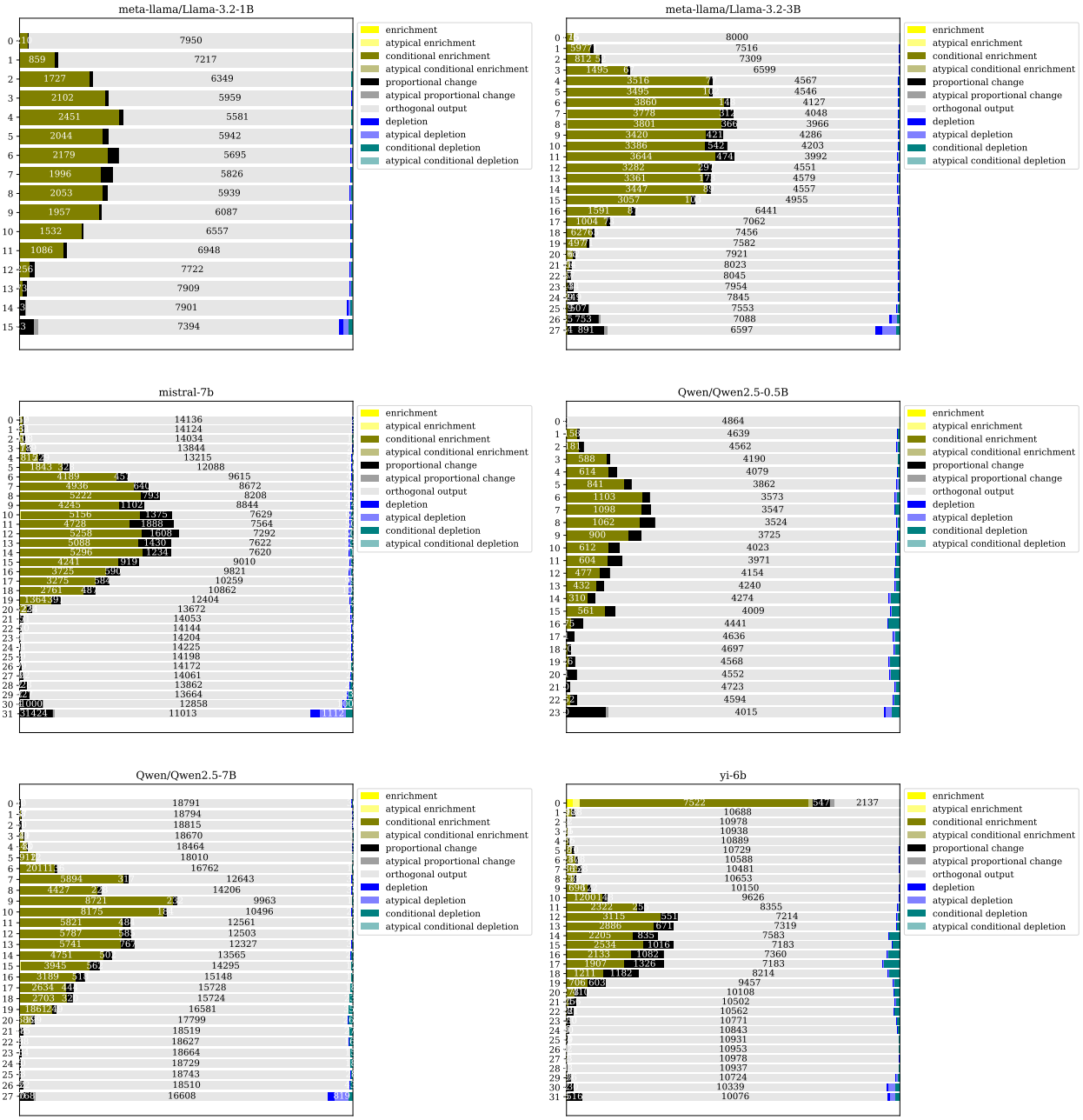


Figure 8: Continuation of Figure 7. Including a copy of Figure 3 (Llama-3.2-3B) for convenience.

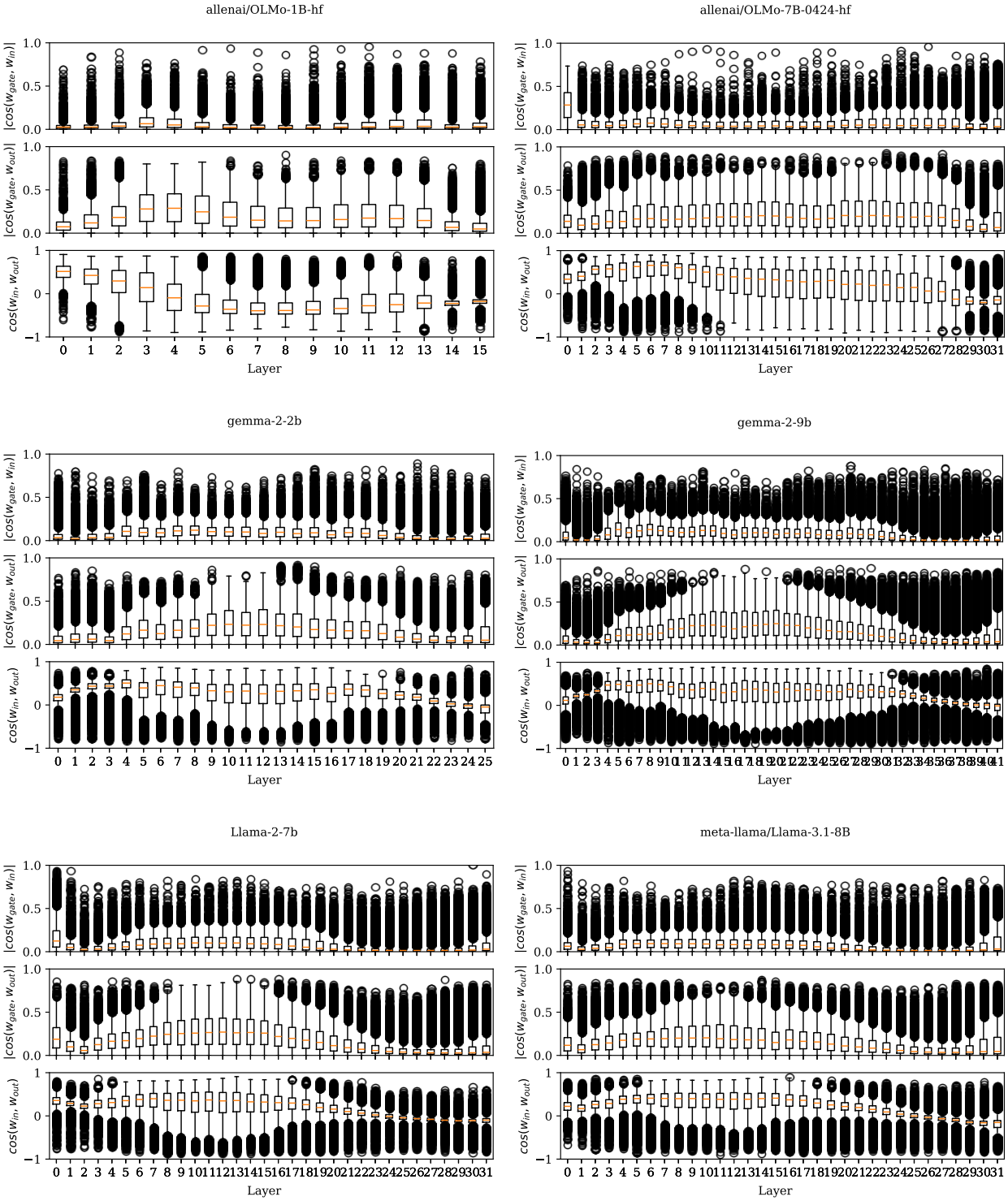


Figure 9: Boxplots for the distribution of weight cosine similarities in each layer. For $\cos(w_{\text{gate}}, w_{\text{in}})$ and $\cos(w_{\text{gate}}, w_{\text{out}})$ we show the absolute value since their sign does not carry any information on its own.

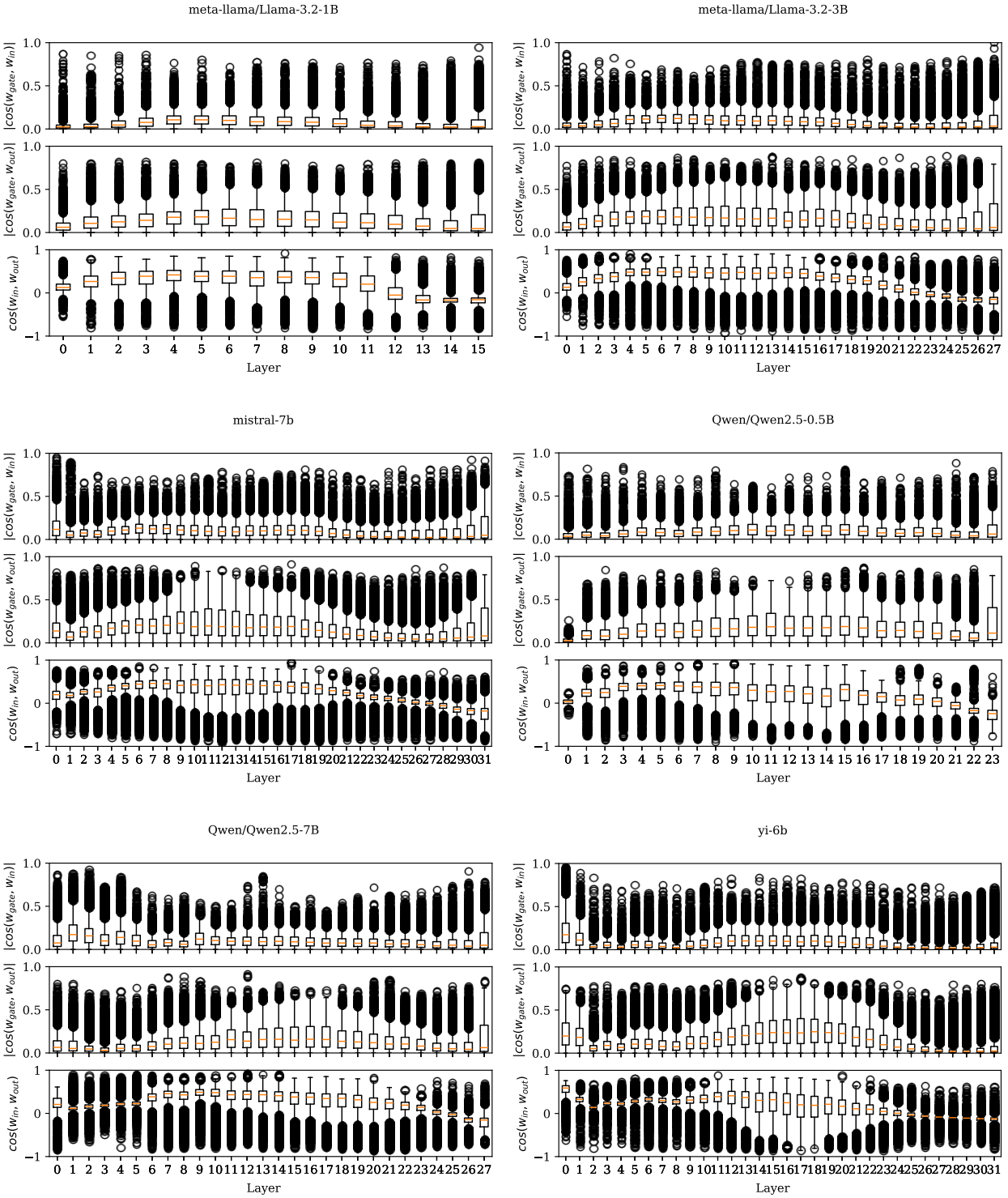
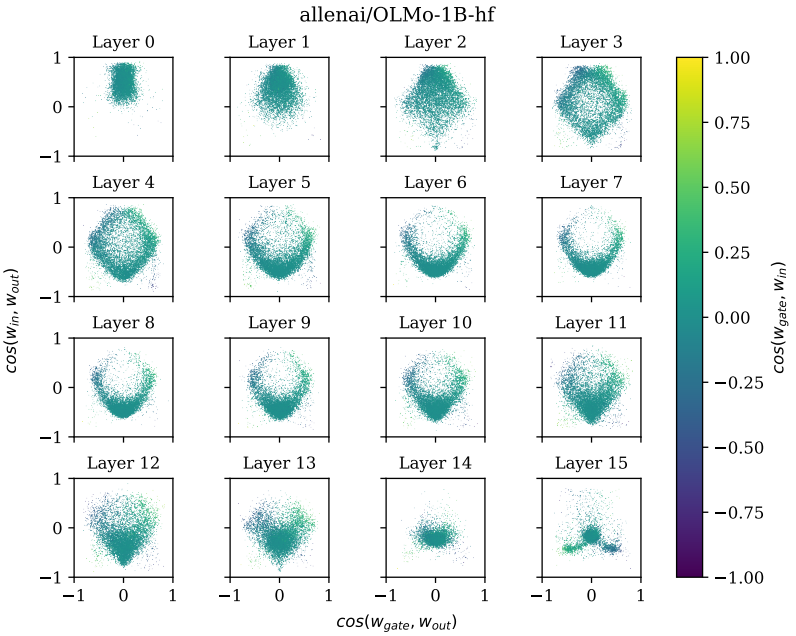


Figure 10: Continuation of Figure 9. Including a copy of Figure 4 (Llama-3.2-3B) for convenience.


allenai/OLMo-1B-hf
Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Layer 11
Layer 12
Layer 13
Layer 14
Layer 15
cos(w_in, w_out)
cos(w_gate, w_out)
cos(w_gate, w_in)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Figure 11

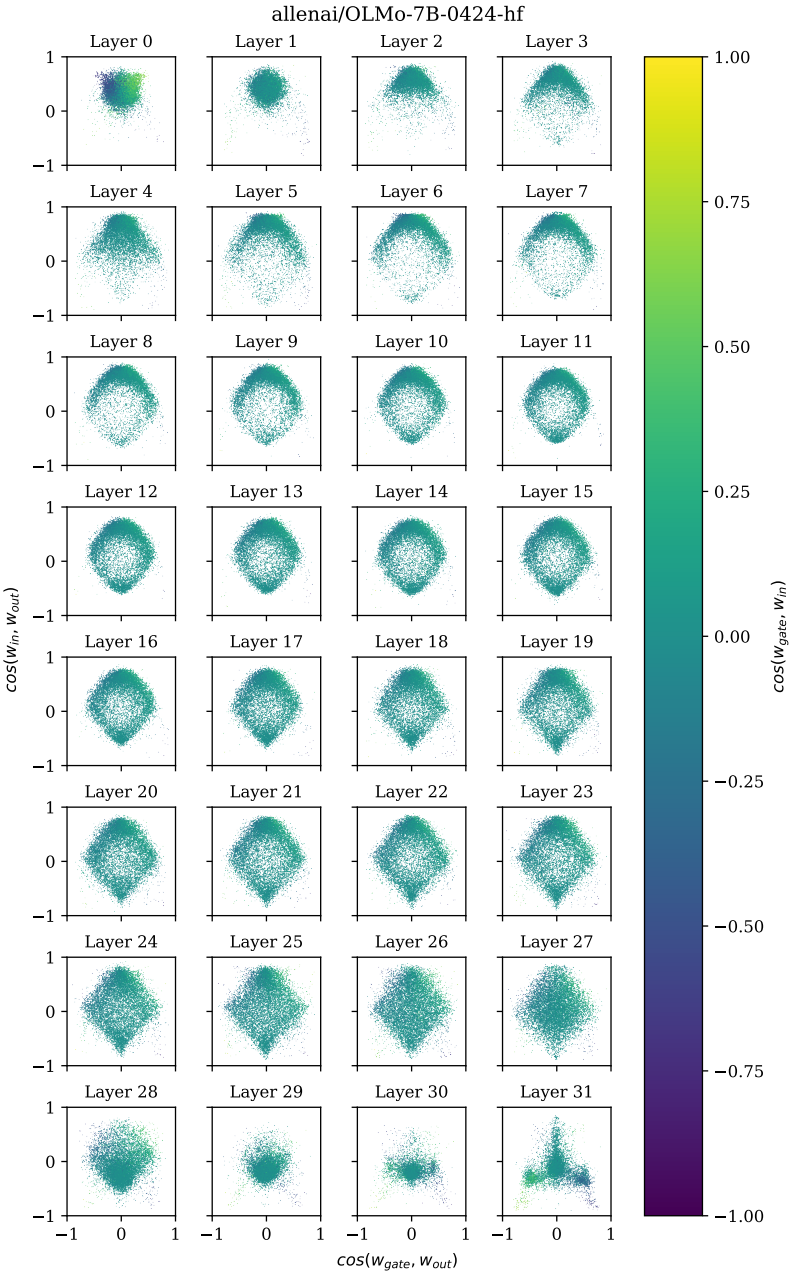
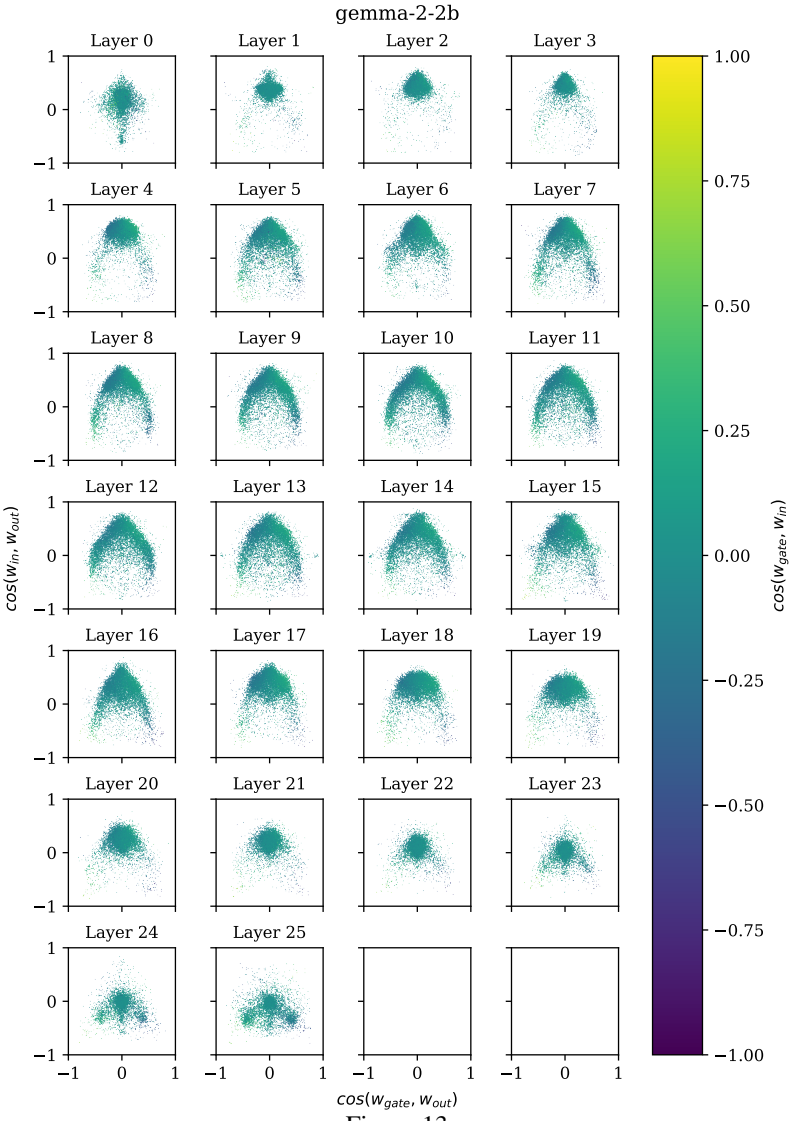
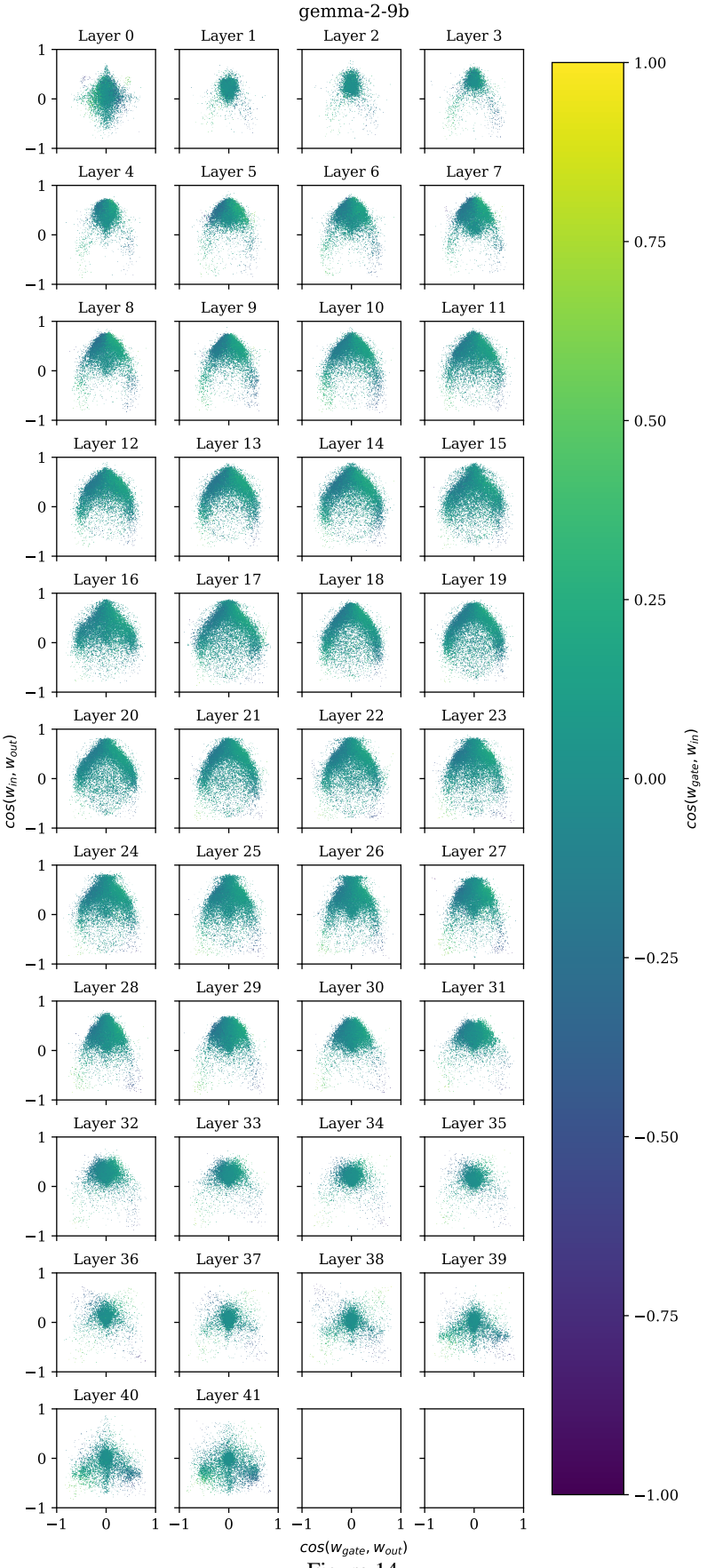

allenai/OLMo-7B-0424-hf
Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Layer 11
Layer 12
Layer 13
Layer 14
Layer 15
Layer 16
Layer 17
Layer 18
Layer 19
Layer 20
Layer 21
Layer 22
Layer 23
Layer 24
Layer 25
Layer 26
Layer 27
Layer 28
Layer 29
Layer 30
Layer 31
cos(w_in, w_out)
cos(w_gate, w_out)
cos(w_gate, w_in)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Figure 12


gemma-2-2b
Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Layer 11
Layer 12
Layer 13
Layer 14
Layer 15
Layer 16
Layer 17
Layer 18
Layer 19
Layer 20
Layer 21
Layer 22
Layer 23
Layer 24
Layer 25
$cos(w_{in}, w_{out})$
$cos(w_{gate}, w_{out})$
$cos(w_{gate}, w_{in})$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Figure 13


gemma-2-9b
Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Layer 11
Layer 12
Layer 13
Layer 14
Layer 15
Layer 16
Layer 17
Layer 18
Layer 19
Layer 20
Layer 21
Layer 22
Layer 23
Layer 24
Layer 25
Layer 26
Layer 27
Layer 28
Layer 29
Layer 30
Layer 31
Layer 32
Layer 33
Layer 34
Layer 35
Layer 36
Layer 37
Layer 38
Layer 39
Layer 40
Layer 41
$cos(w_{in}, w_{out})$
$cos(w_{gate}, w_{out})$
$cos(w_{gate}, w_{in})$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Figure 14

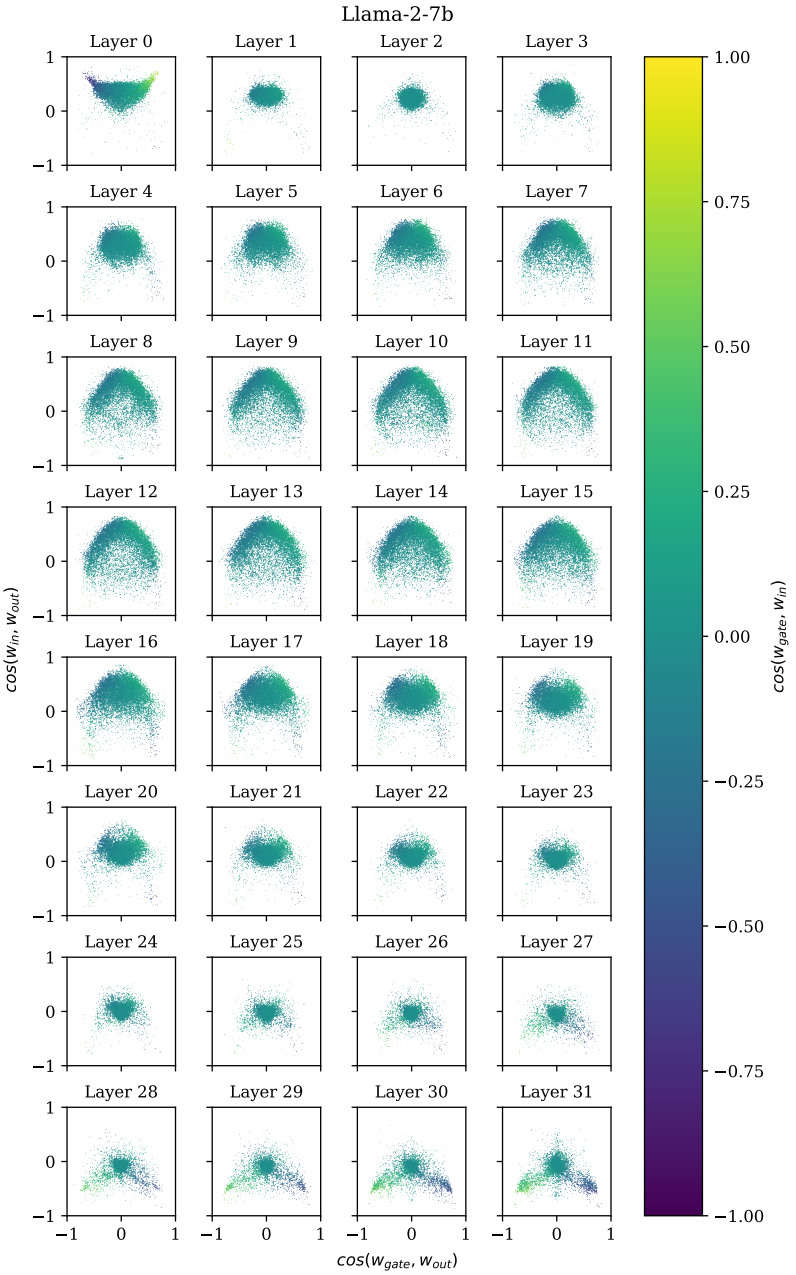
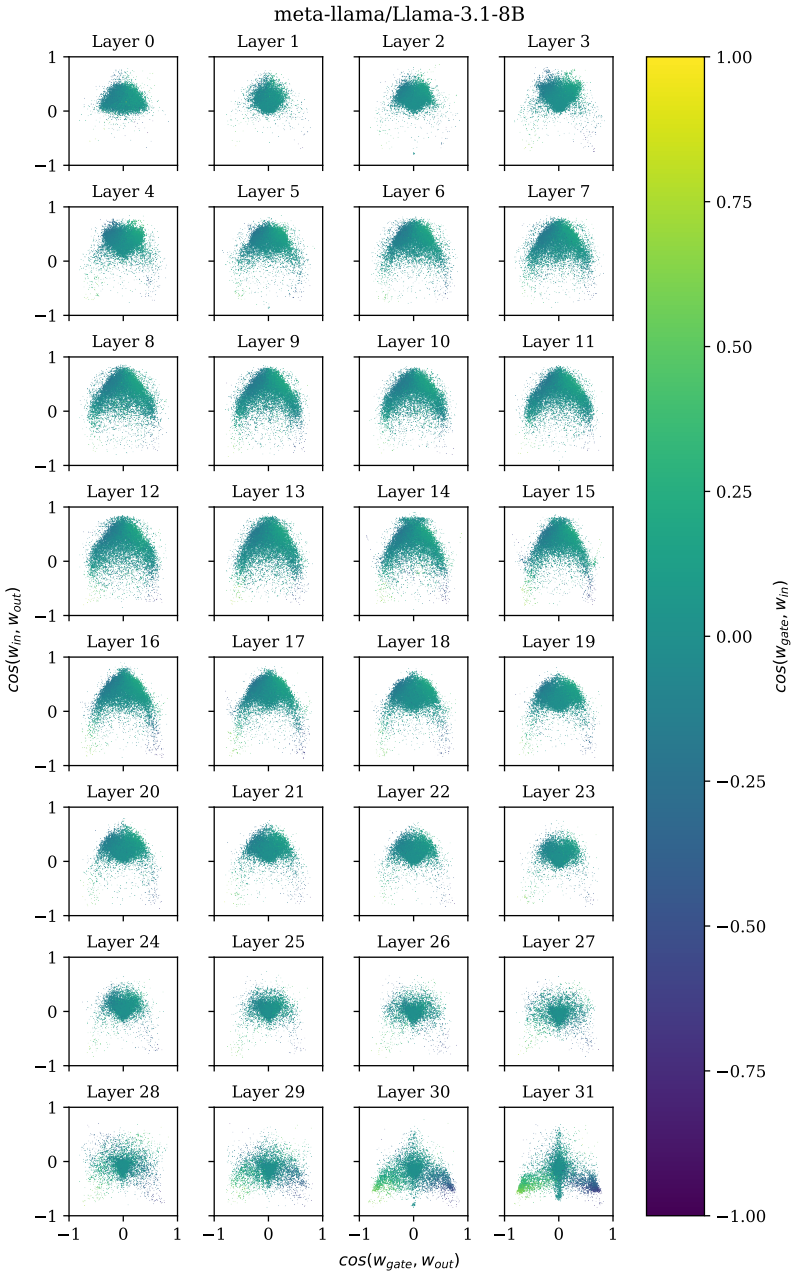

Llama-2-7b
Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Layer 11
Layer 12
Layer 13
Layer 14
Layer 15
Layer 16
Layer 17
Layer 18
Layer 19
Layer 20
Layer 21
Layer 22
Layer 23
Layer 24
Layer 25
Layer 26
Layer 27
Layer 28
Layer 29
Layer 30
Layer 31
$cos(w_{in}, w_{out})$
$cos(w_{gate}, w_{out})$
$cos(w_{gate}, w_{in})$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Figure 15


meta-llama/Llama-3.1-8B
Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Layer 11
Layer 12
Layer 13
Layer 14
Layer 15
Layer 16
Layer 17
Layer 18
Layer 19
Layer 20
Layer 21
Layer 22
Layer 23
Layer 24
Layer 25
Layer 26
Layer 27
Layer 28
Layer 29
Layer 30
Layer 31
$cos(w_{in}, w_{out})$
$cos(w_{gate}, w_{out})$
$cos(w_{gate}, w_{in})$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Figure 16

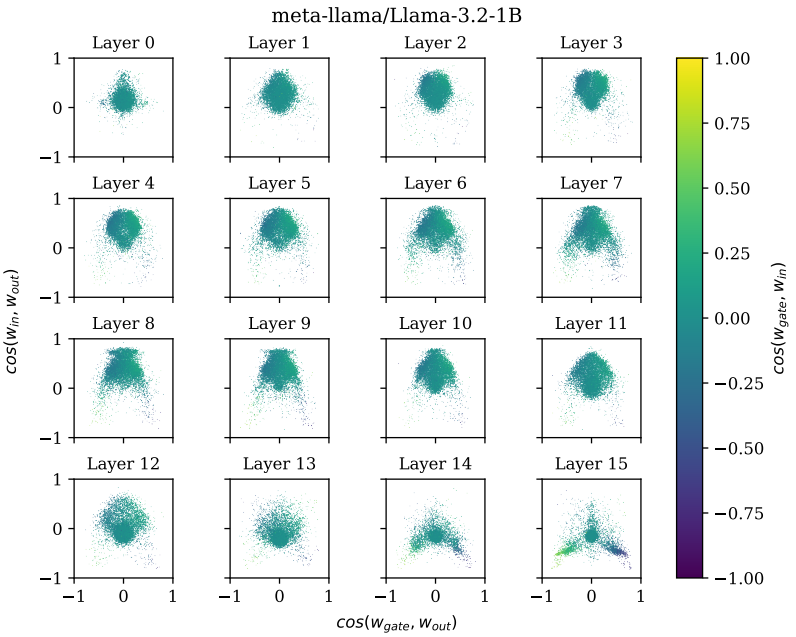


Figure 17

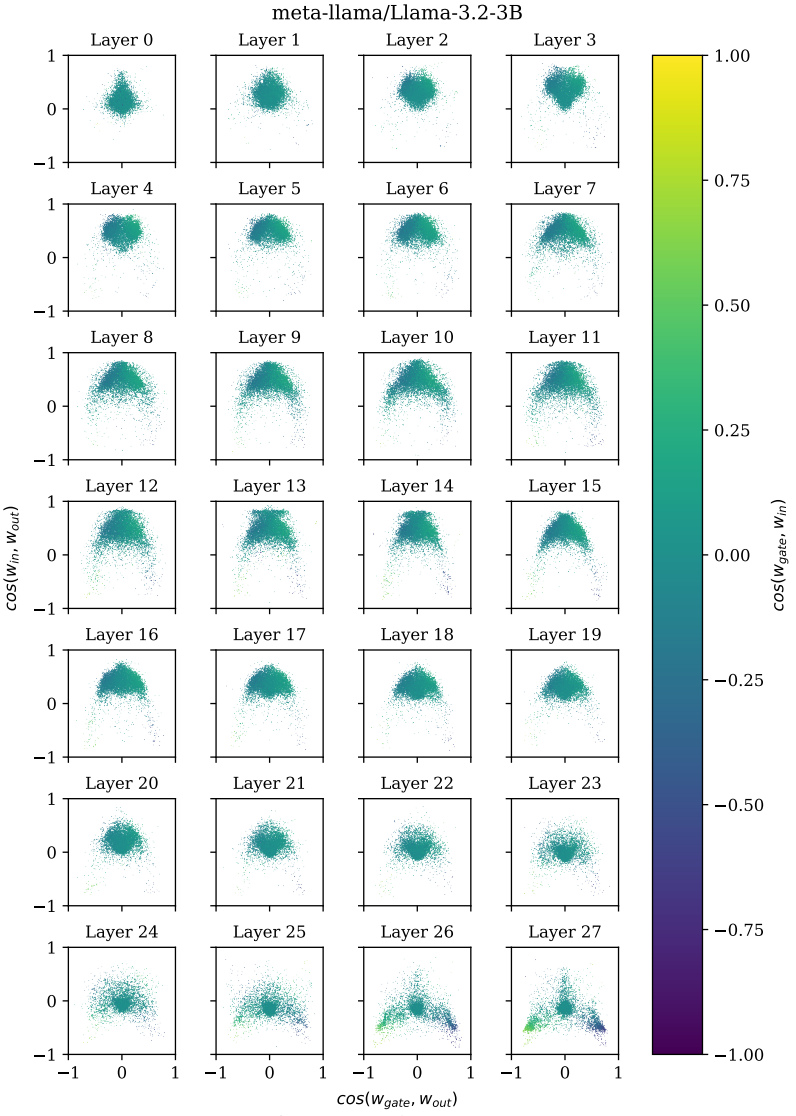


Figure 18: Llama-3.2-3B

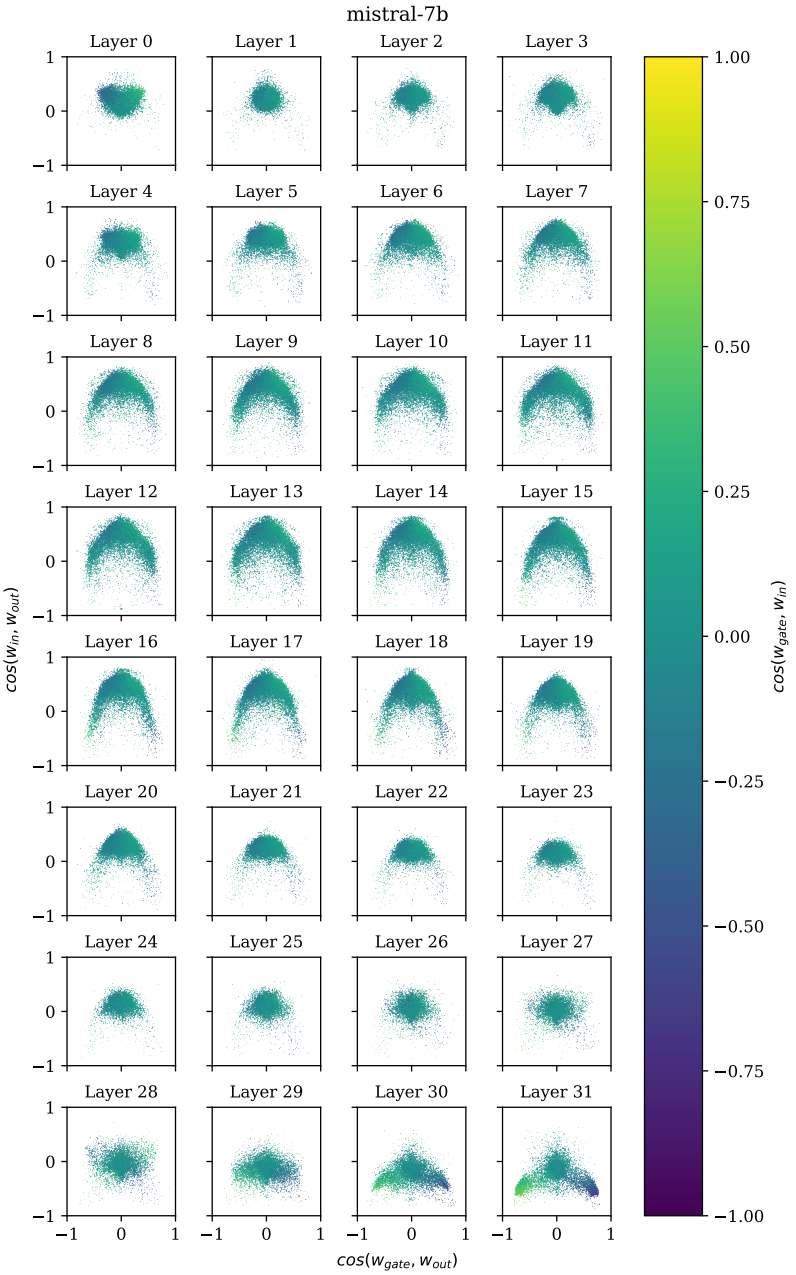
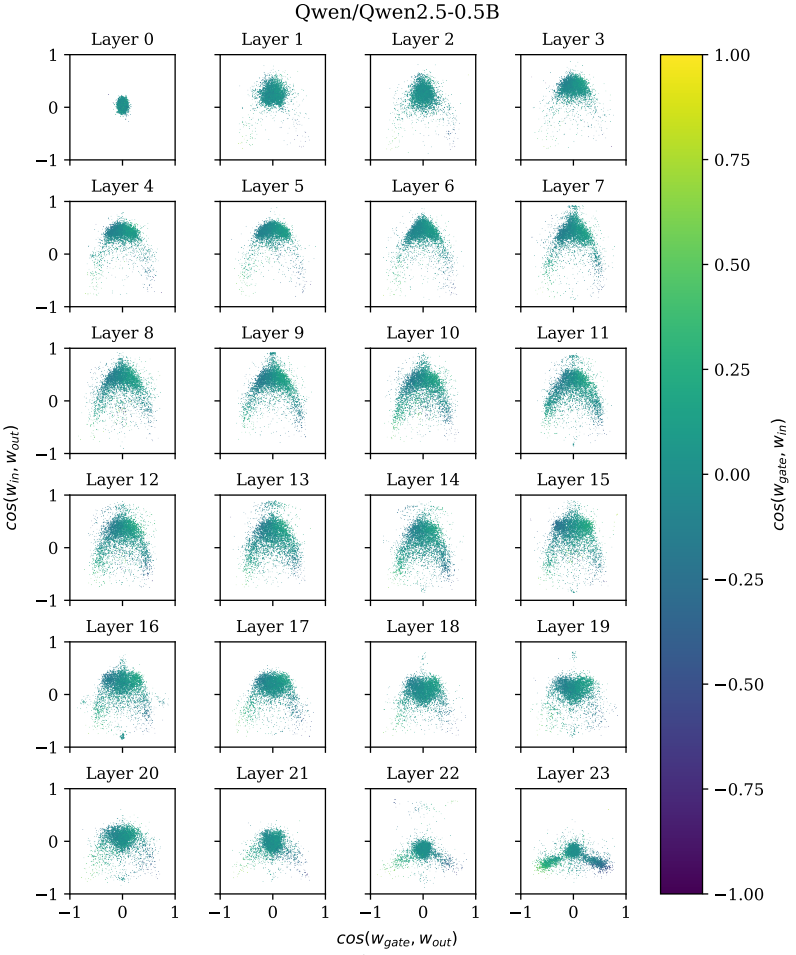
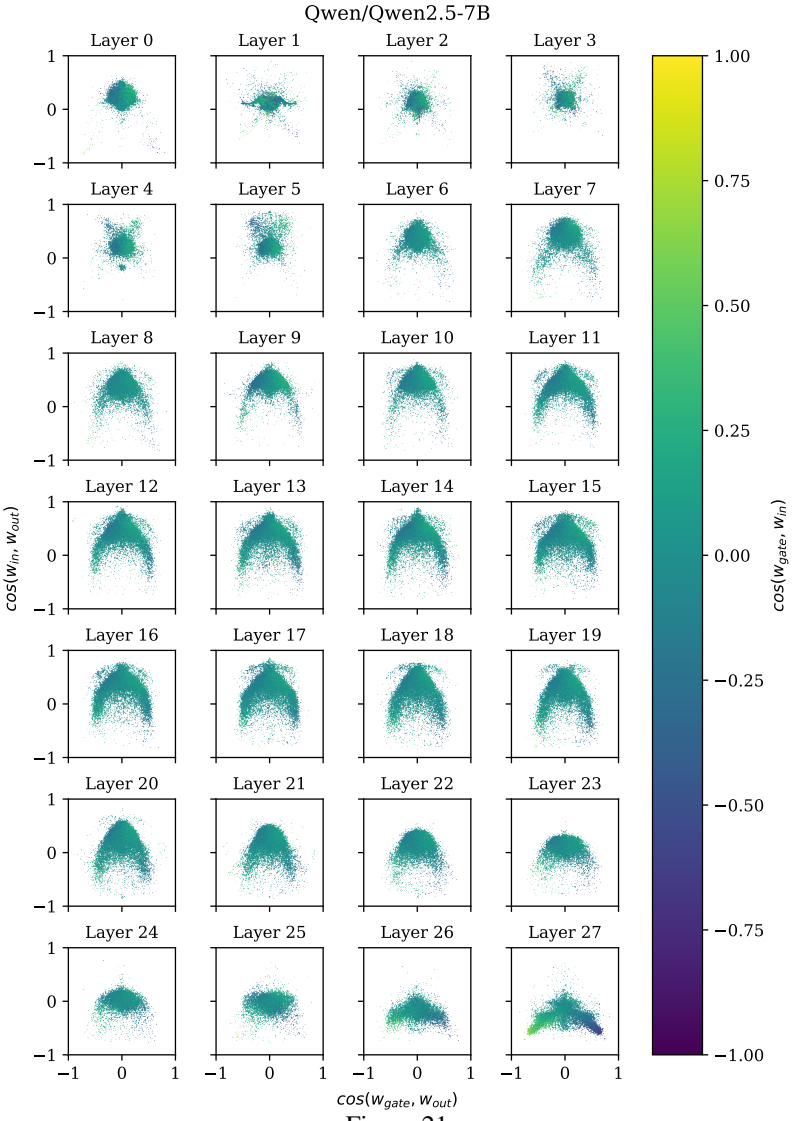

mistral-7b
Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Layer 11
Layer 12
Layer 13
Layer 14
Layer 15
Layer 16
Layer 17
Layer 18
Layer 19
Layer 20
Layer 21
Layer 22
Layer 23
Layer 24
Layer 25
Layer 26
Layer 27
Layer 28
Layer 29
Layer 30
Layer 31
$cos(w_{in}, w_{out})$
$cos(w_{gate}, w_{out})$
$cos(w_{gate}, w_{in})$

Figure 19


Qwen/Qwen2.5-0.5B
Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Layer 11
Layer 12
Layer 13
Layer 14
Layer 15
Layer 16
Layer 17
Layer 18
Layer 19
Layer 20
Layer 21
Layer 22
Layer 23
$cos(w_{in}, w_{out})$
$cos(w_{gate}, w_{out})$
$cos(w_{gate}, w_{in})$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Figure 20


Qwen/Qwen2.5-7B
Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Layer 11
Layer 12
Layer 13
Layer 14
Layer 15
Layer 16
Layer 17
Layer 18
Layer 19
Layer 20
Layer 21
Layer 22
Layer 23
Layer 24
Layer 25
Layer 26
Layer 27
cos(w_in, w_out)
cos(w_gate, w_out)
cos(w_gate, w_in)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Figure 21

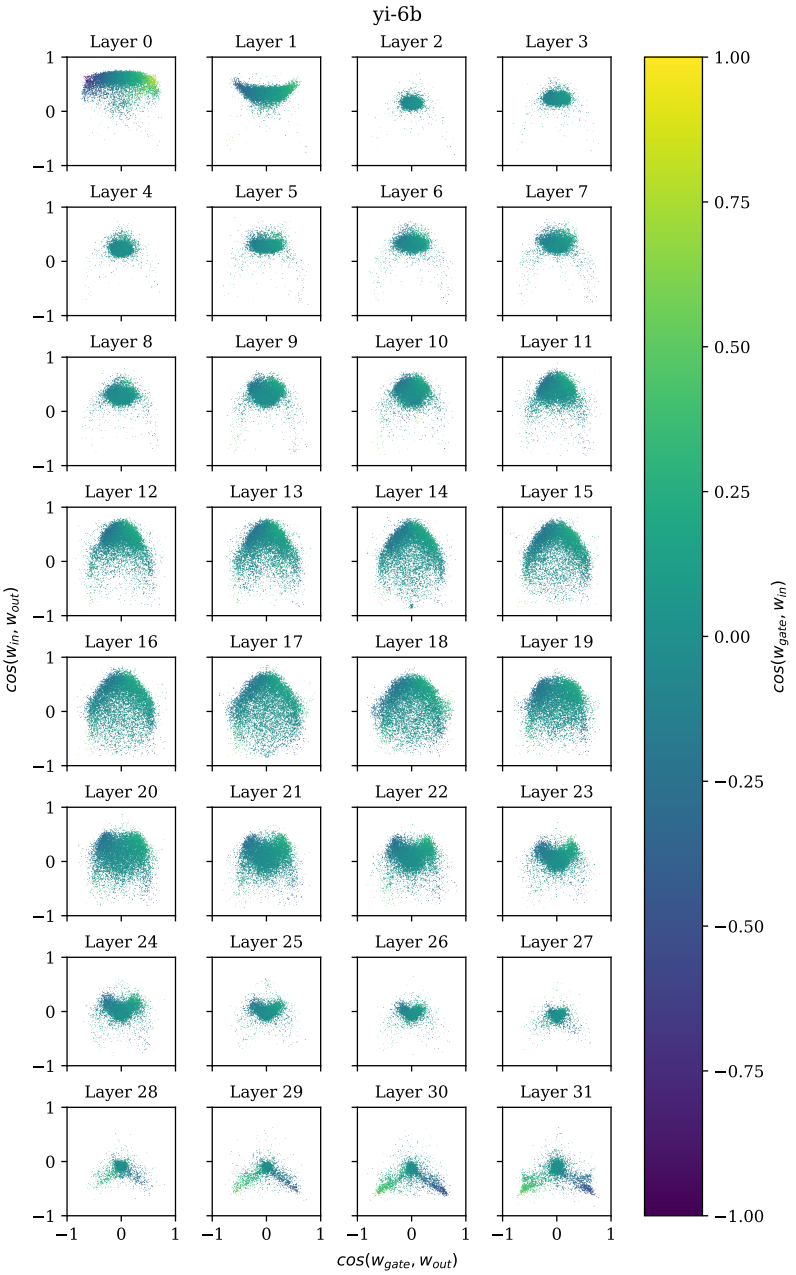

yi-6b
Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Layer 11
Layer 12
Layer 13
Layer 14
Layer 15
Layer 16
Layer 17
Layer 18
Layer 19
Layer 20
Layer 21
Layer 22
Layer 23
Layer 24
Layer 25
Layer 26
Layer 27
Layer 28
Layer 29
Layer 30
Layer 31
$cos(w_{in}, w_{out})$
$cos(w_{gate}, w_{out})$
$cos(w_{gate}, w_{in})$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Figure 22