Equivalent Linear Mappings of Large Language Models

Anonymous authors
Paper under double-blind review

Abstract

Despite significant progress in transformer interpretability, an understanding of the computational mechanisms of large language models (LLMs) remains a fundamental challenge. We demonstrate that LLM inference for a given input sequence can be mapped to an equivalent and interpretable linear system that reconstructs the predicted output embedding, where the token probabilities computed from the linear reconstruction achieve relative error below 10^{-14} at double floating-point precision. Our approach exploits a fundamental structural property of transformer architectures wherein every operation (gated activations, attention, and normalization) can be expressed as $A(x) \cdot x$, where $A(x)$ represents an input-dependent coefficient matrix and x preserves the linear pathway. To expose this linear structure, we strategically detach components of the gradient computation with respect to an input sequence, freezing the $A(x)$ terms at their values computed during inference. This "detached'' Jacobian of the model reconstructs the output with one linear operation per input token. We demonstrate this approach across models, including Qwen 3, Gemma 3 and Llama 3, up to Qwen 3 14B. These linear representations of models for particular input token sequences reveal that LLMs operate in extremely low-dimensional subspaces where the singular vectors can be decoded to interpretable semantic concepts. We examine how the linear representations of individual layers as well as their attention and multilayer perceptron modules build predictions, and use these as steering operators to insert semantic concepts into unrelated text. Despite their expressive power and global nonlinearity, modern LLMs can be interpreted through equivalent linear representations that reveal interpretable semantic structures in the next-token prediction process.

1 Introduction

The transformer decoder has become the architecture of choice for large language models (Vaswani et al., 2017) and efforts toward a conceptual understanding of its mechanisms are ongoing. Significant insights include sparse autoencoders for conceptual activations in LLMs (Bricken et al., 2023), "white-box" alternative architectures (Yu et al., 2023), minimally sufficient architectures (He & Hofmann, 2023) and analytic results on generalization (Cowsik et al., 2024). While transformers are complex globally nonlinear functions of their input, we demonstrate how to compute an equivalent linear system that reconstructs the predicted output embedding for a given input sequence up to double floating-point precision.

Our approach builds on two previous results: Elhage et al. (2021) found that attention-only networks with basic language generation abilities transfer semantic information across the network with interpretable circuits, including the "induction head", and Kadkhodaie et al. (2023) showed that powerful image denoising diffusion models with several architectural constraints are piecewise linear functions and interpreted as low-dimensional adaptive linear filters.

We extend these ideas by demonstrating equivalent linear systems for modern LLMs with gated linear activations. For many open-weight LLMs, the gradient operation with respect to the input can be manipulated at inference such that the output prediction is unchanged but has a linear equivalent. This numerical Jacobian computation captures the complete forward operation of the model, including activation functions and attention modules, although it is only valid for that particular input sequence (not piecewise linear but "pointwise" linear).

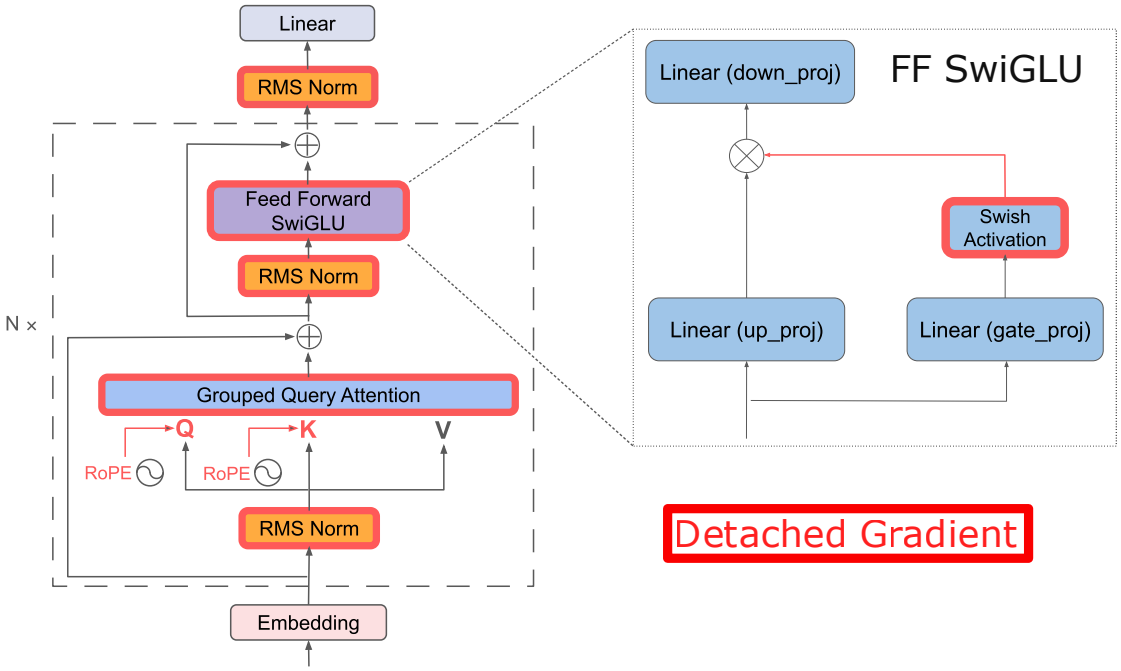

Figure 1: A schematic of the Llama 3 transformer decoder (Grattafiori et al., 2024; Nvidia, 2024). The PyTorch gradient detach operations for components outlined in red effectively freeze the nonlinear activations for a given input sequence, creating a linear path for the gradient with respect to the input embedding vectors, but do not change the output. The output embedding prediction can be mapped to an equivalent linear system by the Jacobian autograd operation. The feedforward module with a SwiGLU activation function is shown in expanded form to demonstrate how the nonlinear Swish term can be detached from the gradient to form a linear path, achieving linearity for a given input. The RMSNorm layers and and softmax attention blocks also must be detached from the gradient.

This approach allows us to analyze entire models, from input embeddings to predicted output embedding, as an equivalent linear system for a particular input sequence. By examining the singular value decomposition (SVD) of the equivalent linear system, we can measure the local dimensionality of the learned manifolds involved in next-token prediction and can decode the singular vectors into output tokens. This analysis can also be done layer by layer, or for individual attention and multilayer perceptron (MLP) modules, in order to observe how these models compose next-token predictions.

We demonstrate equivalent linearity in model families including Qwen 3, Gemma 3, Llama 3, at a range of sizes up to Qwen 3 14B. This approach offers a new path to interpreting a wide range of open-weight LLMs for specific inputs that could serve as a complement to other powerful interpretability methods. When applied across next-token prediction for entire datasets, this equivalent linear approach could shift interpretability away from the hard problem of reverse-engineering globally nonlinear functions toward the more tractable problem of analyzing many equivalent linear systems in a manifestation of the bitter lesson (Sutton, 2019). While computationally intensive, this approach does not require additional model training, and offers mathematical exactness and broad applicability across model architectures, suggesting that interpretability insights may come through computational scale rather than hand-crafted methods for reverse engineering.

2 Method

2.1 The Jacobian of a deep ReLU Network

Mohan et al. (2019) observed that deep *ReLU* networks for image denoising which utilize zero-bias linear layers are "adaptive linear" functions due to their homogeneity of order 1 at a given fixed input, which

enables interpretation as an exactly equivalent linear system. Given the homogeneity at a fixed input, the network's output can be reproduced by numerically computing the Jacobian matrix of the network at a particular input image $\mathbf{x^*_{im}}$ and multiplying it by $\mathbf{x^*_{im}}$.

$$\mathbf{y^*_{im}} = \mathbf{J}(\mathbf{x^*_{im}}) \cdot \mathbf{x^*_{im}} \tag{1}$$

Due to the global nonlinearity of the network, the Jacobian must usually be computed again at every input of interest. The Jacobian may be the same for similar inputs in the same piecewise region of the response (Balestriero & Baraniuk, 2021) (but this will be demonstrated to not be the case for transformer architectures).

2.2 The Jacobian of a transformer decoder

Many open weight LLMs also use linear layers with zero bias, as required for linearity in the architecture of Mohan et al. (2019). A transformer decoder predicts an output token embedding $\mathbf{y}$ given a sequence of k input tokens $\mathbf{t} = (\mathbf{t_0}, \mathbf{t_1}..., \mathbf{t_k})$ mapped to input embedding vectors $\mathbf{x} = (\mathbf{x_0}, \mathbf{x_1}..., \mathbf{x_k})$, where $\mathbf{t}^*$ and $\mathbf{x}^*$ represent a particular sequence. The output embedding prediction is a nonlinear function of the input embedding vectors $\mathbf{x_0}, \mathbf{x_1}$,...$\mathbf{x_k}$, as LLMs utilize nonlinear gated activation functions for layer outputs (SwiGLU for Llama 3, GELU for Gemma 3 and Swish for Qwen 3) as well as normalization and softmax attention blocks.

Gated activations like $Swish(\mathbf{x}) = \mathbf{x} \cdot sigmoid(\mathbf{x})$, with a linear term and a nonlinear term, are also an "adaptive" linear function or, more generally, an adaptive homogeneous function of order 1 (Mohan et al., 2019). If the $sigmoid(\mathbf{x})$ term that gives rise to the nonlinearity is frozen for a specific numerical input, e.g. an embedding vector $\mathbf{x^*_0}$ (Elhage et al., 2021) (or equivalently detached from the computational graph with respect to the input), then we have a linear function valid only at $\mathbf{x^*_0}$ where (1) holds and we can numerically compute a Jacobian matrix that carries out $Swish(\mathbf{x^*_0})$ as a linear operation.

Below we show that computing the Jacobian after effectively substituting specific values for the nonlinear terms also works for other gated activation functions, normalization layers and softmax attention blocks. We further demonstrate that for a given input sequence we can apply necessary gradient detachments so that the entire transformer decoder is an adaptive homogeneous function of order 1, and numerically compute the equivalent linear system that reproduces the transformer output embedding $\mathbf{y}^*$.

The Jacobian $\mathbf{J}(\mathbf{x})$ of a transformer is the set of matrices generated by taking the partial derivative of the decoder inference function $\mathbf{y}(\mathbf{x}) = f(\mathbf{x_0}, \mathbf{x_1}..., \mathbf{x_k})$, with respect to each element of each $\mathbf{x_i}$ (where $\mathbf{x_i}$ for Llama 3.2 3B has length 3072, for example, and therefore the Jacobian matrix for each embedding vector is a square matrix of this size). If a transformer decoder were naturally a homogeneous function of order 1, this Jacobian would produce an exact reconstruction of the output.

However, this is not the case. In order to numerically compute an equivalent linear representation, we introduce a "detached" Jacobian $\mathbf{J}^+$, which is a set of matrices that captures the full nonlinear forward computation for a particular input sequence $\mathbf{x}^*$ as a linear system. The detached Jacobian is the numerical Jacobian of the LLM forward operation when its gradient includes a specific set of $detach()$ operations for the nonlinear terms in the normalization, activation and attention operations that force the function to be "adaptively" homogeneous of order 1. Fig. A1 shows how each matrix of the detached Jacobian operates on its corresponding input embedding vector to provide a reconstruction of the LLM forward operation (validated by the PyTorch "allclose" function for absolute and relative tolerances of 10^{-14}), see Fig. 2).

$$\mathbf{y}^* = \sum_{i=0}^{k} \mathbf{J^+_i}(\mathbf{x}^*) \cdot \mathbf{x^*_i} \tag{2}$$

The conventional Jacobian $\mathbf{J}$ for a particular input sequence $\mathbf{x}^*$ (as in Mohan et al. (2019)) does not generate an accurate reconstruction the nonlinear LLM forward operation since the transformer function is not homogeneous of order 1. The detached Jacobian $\mathbf{J}^+$ evaluated at $\mathbf{x}^*$ is the result of an alternative gradient path through the same network which is homogeneous with resepct to the input $\mathbf{x}^*$. The detached Jacobian $\mathbf{J}^+$ only generates an accurate reconstruction at $\mathbf{x}^*$ and not in the local neighborhood due to the strong non-

linearity of the decoder inference function. The detached Jacobian matrices differ for every input sequence and must be computed numerically for every sequence.

2.3 Nonlinear layers as linear operators for a given input

In order to achieve linearity, modifications must be made to the gradient computations of the RMSNorm operation, the activation function (SwiGLU in Llama 3.2) and the softmax term in the attention block output.

2.3.1 Normalization

Normalization layers like LayerNorm (Xu et al., 2019) or RMSNorm (Zhang & Sennrich, 2019) with zero bias are nonlinear with respect to their input because they include division by the square root of the variance of the input.

$$norm(\mathbf{x}) = \frac{\mathbf{x}}{\sqrt{var(\mathbf{x})}} \tag{3}$$

Mohan et al. (2019) devised a novel bias-free batch-norm layer which detaches the variance term from the network's computational graph (see their code implementation). Their batch-norm layer returns the same values as the standard batch-norm layer, but it is linear at inference as the nonlinear operation is removed from the gradient computation. This is also similar to the "freezing" of nonlinear terms in attention-only transformers from Elhage et al. (2021).

We make a similar change for Llama 3.2 3B by altering how the gradient with respect to the input is computed at inference for RMSNorm. This is accomplished by substituting the value for the input vector $\mathbf{x}^*$ for only the variance term as in (4). In PyTorch, this is accomplished by cloning and detaching the $\mathbf{x}$ tensor within the variance operation, so its value will be treated as a constant. The gradient operation is still tracked for $\mathbf{x}$ in the numerator, so that term will be treated as a variable by the PyTorch autograd function for computing the Jacobian. The gradient of the function is then computed at $\mathbf{x}^*$ (we assume for simplicity an input sequence of length 1).

$$norm(\mathbf{x}) = \frac{\mathbf{x}}{\sqrt{var(\mathbf{x}^*)}} \tag{4}$$

We define the detached Jacobian as follows:

$$\mathbf{J}_{\mathbf{n}}^{+} = [\frac{\partial}{\partial \mathbf{x}} norm(\mathbf{x})]|_{\mathbf{x}=\mathbf{x}^*} \tag{5}$$

We can rewrite the pointwise linear RMSNorm as follows:

$$norm(\mathbf{x}^*) = \mathbf{J}_{\mathbf{n}}^{+}(\mathbf{x}^*) \cdot \mathbf{x}^* \tag{6}$$

At inference for a given input, we now have a linear RMSNorm whose output is numerically identical to the one used in training. However, when we take the gradient with respect to the input vector $\mathbf{x}$ in *eval* mode, the numerical output is the detached Jacobian matrix $\mathbf{J}_{\mathbf{n}}^{+}$, which we can use to reconstruct the normalization output as a linear system.

The goal is to apply this same approach for other nonlinear functions in the decoder such that the entire computation from the input embedding vectors to the predicted output is linear for a given input, and we can compute and interpret the set of detached Jacobian matrices.

2.3.2 Activation functions

While Mohan et al. (2019) relied on *ReLU* activation functions, which do not require any changes to achieve linearity, Llama 3.2 3B uses SwiGLU (Shazeer, 2020), Gemma 3 uses approximate GELU (Hendrycks & Gimpel, 2016) and Qwen 3 uses Swish for activation functions. There is a linear $\mathbf{x}$ term in each of these, and the gradients can be cloned and detached from the nonlinear terms. This manipulation produces a pointwise linear Swish layer with respect to the input $\mathbf{x}$.

$$\text{Swish}\,(\mathbf{x}) = \mathbf{x} \cdot \text{sigmoid}(\mathbf{x}) \tag{7}$$

$$\text{Swish}\,(\mathbf{x}^*) = \mathbf{x} \cdot \text{sigmoid}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} \tag{8}$$

$$\text{Swish}\,(\mathbf{x}^*) = ([\frac{\partial}{\partial \mathbf{x}}\text{Swish}(\mathbf{x})]|_{\mathbf{x}=\mathbf{x}^*}) \cdot \mathbf{x}^* \tag{9}$$

$$\text{Swish}\,(\mathbf{x}^*) = \mathbf{J}^+_{\mathbf{Swish}}(\mathbf{x}^*) \cdot \mathbf{x}^* \tag{10}$$

Detaching the gradient from the Swish output thus allows for a pointwise linear form of Swish at inference. A similar procedure may be carried out for SwiGLU with Llama 3 and GELU with Gemma 3 (see supplement, eq. 17).

2.3.3 Attention

The softmax operation at the output of the attention block can also be detached, with the linear relationship preserved through the subsequent multiplication with $\mathbf{V}$, which is a linear function of $\mathbf{x}$. Below, $\mathbf{Q} = \mathbf{W_Q x}$, $\mathbf{K} = \mathbf{W_K x}$ and $\mathbf{V} = \mathbf{W_V x}$.

$$Attn(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{QK}^T}{\sqrt{d_k}}) \cdot \mathbf{V} \tag{11}$$

$$Attn(\mathbf{x}) = [softmax(\frac{\mathbf{QK}^T}{\sqrt{d_k}})|_{\mathbf{Q}=\mathbf{Q}^*,\mathbf{K}=\mathbf{K}^*}] \cdot \mathbf{W_V x} \tag{12}$$

$$Attn(\mathbf{x}^*) = ([\frac{\partial}{\partial \mathbf{x}} Attn(x)]|_{\mathbf{x}=\mathbf{x}^*}) \cdot \mathbf{x}^* \tag{13}$$

$$Attn(\mathbf{x}^*) = \mathbf{J}^+_{\mathbf{Attn}}(\mathbf{x}^*) \cdot \mathbf{x}^* \tag{14}$$

The linear $\mathbf{x}$ term within $\mathbf{V}$ makes it possible for the attention block to be pointwise linear at inference, as the gradient for the softmax output is detached.

2.3.4 The Transformer Decoder

With the the above gradient detachments for the normalization layers, activation functions and attention blocks, the transformer decoder network is linear with respect to $\mathbf{x}^*$ when evaluated at $\mathbf{x}^*$ (shown here for length k).

$$\mathbf{y}^* = \sum_{i=0}^{k} \mathbf{J}^+_{\mathbf{i}}(\mathbf{x}^*) \cdot \mathbf{x}^*_{\mathbf{i}} \tag{15}$$

The output of the network incorporating the above gradient detachments is unchanged from the original architecture but has an equivalent linear representation.

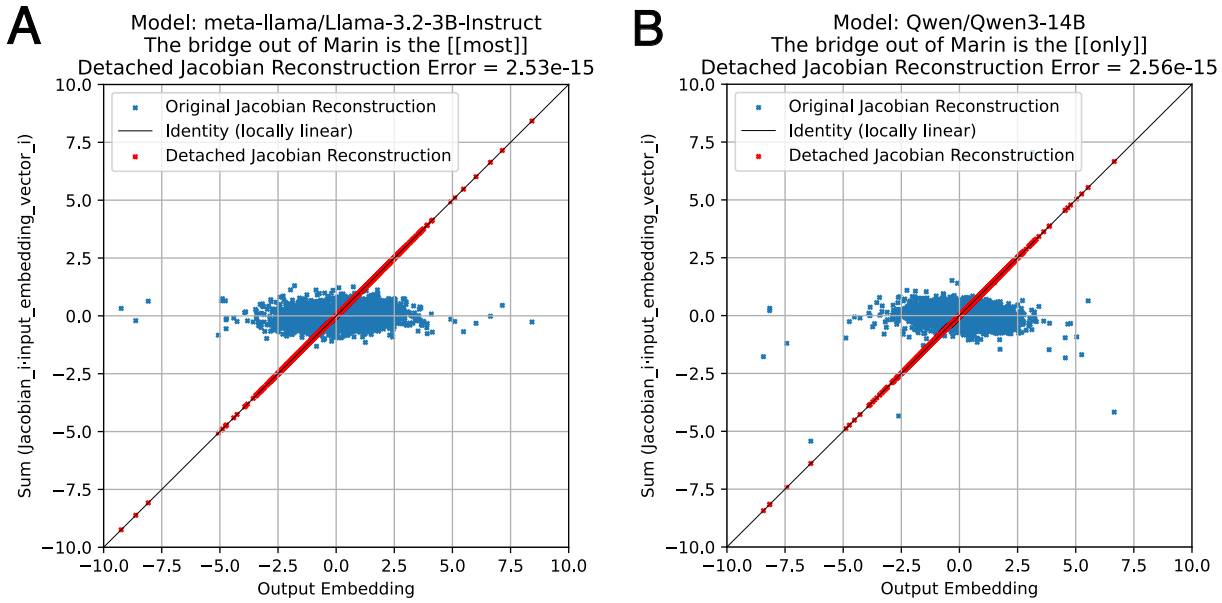

Figure 2: For the input sequence "The bridge out of Marin is the", the elements of the predicted output embedding vector of the model compared to the elements from the Jacobian reconstruction for both the original Jacobian (blue points) and detached Jacobian operations (red points), shown for A) Llama 3.2 3B and B) Qwen 3 14B. Note that the detached Jacobian reconstructions match the predicted embedding, with relative error (the norm of the reconstruction error divided by the norm of the output embedding) on the order 10^{-14} for $float64$ precision.

3 Results

3.1 Pointwise linearity of the predicted output

In order to validate whether the detached Jacobian achieves reconstruction with a linear representation, we can compare the predicted output embedding vector for a given input token sequence to the reconstruction of the output. As a baseline, we can also compute the reconstruction using the conventional Jacobian as in Mohan et al. (2019) and examine its accuracy. Given the above argument that the appropriate gradient detachments are necessary to achieve output reconstruction, we expect the detached Jacobian to accomplish reconstruction, but the conventional Jacobian to fail.

Fig. 2 compares the network output to both the conventional and detached Jacobian reconstructions for Llama 3.2 3B and Qwen 3 14B. The reconstruction of the output embedding with the detached Jacobian matrices falls on the identity line when compared with the output embedding, showing accurate reconstruction, while the reconstruction with the conventional Jacobian is not at all close to the output. This comparison therefore demonstrates the validity of the reconstruction with the linear system of the detached Jacobian for Qwen 3 14B for a particular input.

In order to examine the fidelity of the detached Jacobian reconstruction, we compared the reconstruction against the network output using PyTorch function allclose with varying tolerance levels. The reconstructions achieved numerical agreement within a relative and absolute tolerance of 10^{-14}. This tolerance is approximately 50 times the machine epsilon of $2.2 \cdot 10^{-16}$ for 64-bit floating-point numbers, indicating high-fidelity reconstruction that is numerically equivalent to the reference implementation for practical purposes. As an additional metric, the norm of the detached Jacobian reconstruction error divided by the norm out of the output is also on the order of 10^{-14}.

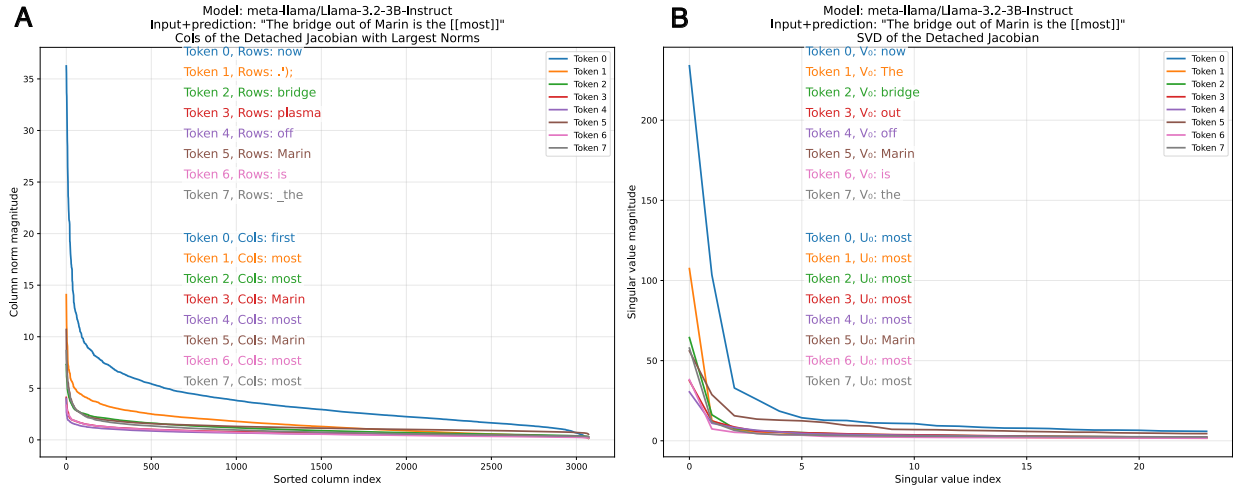

Figure 3: Given the sequence "The bridge out of Marin is the", the most likely prediction is "most" for Llama 3.2 3B. The detached Jacobian matrices for each token represent an equivalent linear system that computes the predicted output embedding. A) We show the features which drive large responses in single units in the last decoder layer, which are the rows of the detached Jacobian with the largest norm values, and decode each of those into the most likely input embedding token. The block of words at the top shows the ordered decoded "feature" input tokens from the largest rows of the detached Jacobian matrix for the input tokens. A similar operation is carried out for columns of the largest norm values, which are decoded to the output token space. Note that the activation distribution of column magnitudes is fairly sparse, with only a few units driving the response. B) We take the singular value decomposition of the detached Jacobian matrix corresponding to each input token, which summarizes the modes driving the response, and decode the right and left singular vectors V and U to input and output embeddings, shown in colors. The singular value spectrum is extremely low rank, and decoding the U singular vectors returns candidate output token, including "most" and "first". Decoding the V singular vectors returns variants of the input tokens like "bridge", "Marin" and "is", as well as others that are not clearly related to the input sequence.

The numerical computation of the full detached Jacobian matrix takes on the order of 10 seconds for an input sequence of 8 tokens for Llama 3.2 3B on a GPU with 16 GB VRAM. In contrast, the full Jacobian matrix for the same sequence at float64 precision with Qwen 3 14B on a GPU with 40 GB VRAM takes 20 seconds. A method for computing the top k singular vectors of the detached Jacobian without forming the full matrix utilizing Lanczos iteration has also been implemented in Jax for Gemma 3 4B, allowing for the efficient computation of the top singular vectors of the detached Jacobian for up to 400 input tokens on a GPU with 40 GB VRAM.

3.2 Single-unit feature selectivity and invariance

Since the detached Jacobian applied to the input embedding reproduces the predicted output embedding vector, and the elements of the predicted output embedding vector are the units of the last transformer layer, the rows of the detached Jacobian matrices represent the input features to which the last layer units are selective and invariant for that particular input sequence (Kadkhodaie et al., 2023; Mohan et al., 2019).

The activation of a particular unit in the last layer is determined by the inner product of a row of the detached Jacobian and the input embedding vector. We can sort by the magnitude of row norms, then map the largest-magnitude rows of the detached Jacobian back to the input embedding space (by finding the nearest-neighbor embedding vectors from the tokenizer) to determine the tokens that cause each unit to be strongly positive or negative. We can see in Fig. 3A that the units respond strongly to the words of the prompt, including "bridge", "Marin" and "is". Decoding of the rows of the detached Jacobian for each token as well as the distribution of activations for this sequence is shown in Fig. 3A.

The columns of the Jacobian can also be decoded to the output token space, and these turn out to be tokens that could be predicted, which include words like “most” or “first”, which could be acceptable outputs.

3.3 Singular vectors of the detached Jacobian

An alternative approach is to look at the singular value decomposition of the detached Jacobian $\mathbf{J_i^+} = \mathbf{U\Sigma V^T}$, following Mohan et al. (2019). Since the detached Jacobian represents the forward computation, and there are no higher order terms, the fact that the SVD is very low rank shows the entire forward computation can be approximated with only a few singular vectors operating on the input embeddings.

In Fig. 3B, the singular vectors are decoded for Llama 3.2 3B (and for other models in supplemental Fig. A2). The right singular vectors V are decoded to input tokens in the same way the rows of the detached Jacobian were above, and we see similar decoding of the top tokens to the features driving the most active single units. The left singular vectors U can be decoded to output embedding tokens, and “most” is the strongest, as it was in the columns of the detached Jacobian matrices.

	Input token 0	Input token 1	Input token 2	
Layer 23_0	salopes CONSTANTS getch Uncomment massage TRY	metaphor bridge largest easiest only first centerpiece	first hardest unc fastest highway bottleneck	
Layer 23_1	metaphor unc Derne makeshift OBJC	bridge Bridge . bridge bridge	scenes WithError opi Timing presets Entering	
Layer 23_2	ros lovers flutter	antics jams	weblog COMPONENT annot metaphor	
Layer 24_0	first third last most largest fourth culmination	metaphor largest centerpiece easiest first hardest bridge gateway	hardest first easiest fastest most ones same	
Layer 24_1	metaphor makeshift .wp REAK	brid bridge bridges bridge Bridge	scenes LocalStorage WithError	
Layer 24_2	flutter ros Lingu	phenomena puzz annot metaphor		
Layer 25_0	largest most first longest latest fastest last third	bridge bridges Bridge gateway	hardest ones exit easiest first most fastest highway	
Layer 25_1	bridge bridges Bridge Bridges brid	bridges bridge Bridge bridge Bridge	(exit exit exits eternity . exit	
Layer 25_2	bridges bridge Bridge bridge parliament	Exit exit jams	INCIDENT symbolism	
Layer 26_0	first most largest last longest latest gateway only	bridge bridges metaphor gateway connecting	highway first exit ones last hardest roads	
Layer 26_1	bridge bridges metaphor Bridges Bridge	bridges bridge structures brid bridge	.charset jams Margins	
Layer 26_2	parliament structures bridges Parliament bridge	Exit exit choke Exit panicked	symbolism metaphor	
Layer 27_0	first last largest bridge longest most oldest latest	bridge bridges Bridge Bridges	last first exit highway bottleneck next road choke	
Layer 27_1	bridge bridges Bridge Bridges	bridges bridge Bridge bridge brid Bridge	EXIT exit exits (exit	
Layer 27_2	bridge bridge bridges Bridge structures	Exit exit Exit	exit . exit incident EXTRA incidents	
Layer 28_0	bridge longest largest first busiest last oldest most	bridge bridges Bridge Bridge	highway exit bottleneck highways Highway last road exits	
Layer 28_1	bridge bridges Bridge Bridge	highway highways coast freeway roads road route	exit exits EXIT exit Exit	
Layer 28_2	bridge bridge bridges Bridge brid	Exit exit Exit exit exit	exit Saddam Mosul Kuwait incident metaphor	
Layer 29_0	bridge only fourth last third longest fifth most	bridge bridges Bridge Bridges	only last first highway third highways exit fourth	
Layer 29_1	bridge bridges Bridge Bridges	coast highway road driveway coastline roads highways freeway	exits exit EXIT	
Layer 29_2	bridge bridges bridge structures brid structure	Exit exit Highway Exit	Saddam Mosul Elvis metaphor incident	
Layer 30_0	bridge most longest fourth third last only fifth	bridge bridges Bridge Bridge	highway only bridge last first Highway road highways	
Layer 30_1	bridge bridges Bridges Bridge	coast freeway highway coastline road roads highways	bridge Bridge bridges bridge brid	
Layer 30_2	bridge structure structures bridges bridge brid	sail seab sailing Bermuda ship	Memphis Kuwait Jordan Saddam Iowa	
Layer 31_0	bridge most only last longest first third largest	bridge bridges Bridge Bridge	only last highway first bridge exit Highway most	
Layer 31_1	coast airlines Interior airline interior Lua Speedway	coast coastline coastal Coast Coastal route	bridge Bridge bridges bridge underwater brid	
Layer 31_2	bridge bridges bridge brid Bridge structure	ship sail sailing dock seab	Jordan Memphis Kuwait Mississippi	
Layer 32_0	bridge most only first last longest third largest	bridge Bridge bridges Bridge bridge	only last first highway most main route exit	
Layer 32_1	interior airline steam airlines Trail breed vacuum	coast coastal coastline route Coast Route beach	bridge span underwater connecting deck public member	
Layer 32_2	bridge bridge bridges Bridge brid Bridge	ship sail dock sailing seab	Kuwait Jordan Memphis Edmonton Mississippi Nile	
Layer 33_0	only first last most third main second subject	bridge Bridge bridges Bridge only	only last first key main same most exit	
Layer 33_1	planet interior cabin floors roots	coast coastline coastal Coast route beach Coastal	span public member library platform floating intervening deck	
Layer 33_2	bridge bridge structure bridges brid Bridge	ship orbit aircraft sail vessel	Kuwait Nile Edmonton Saskatchewan Tulsa	

Table 1: The top-8 token decodings of the largest three singular vectors of the detached Jacobian for the layer outputs from Qwen 3 14B for the sequence “The bridge out of Marin is the” with the prediction [[only]] (note: some tokens do not appear due to unicode display issues). Legend: “Bridge”, “only”, “highway”, “exit”, “most”. This is a selection of the middle layers, see supplemental Table **??** for the full table that extends to the last layer.

3.4 Comparative Analysis of Singular Vectors in Llama 3 and Qwen 3

A direct comparative analysis of the singular vectors derived from the detached Jacobians of Llama 3 3.2B and Qwen 3 4B offers a lens through which to view not only the shared computational principles of modern LLMs but also their distinct data-driven approaches. While both models demonstrate a consistent hierarchical organization of their predictive computations, they diverge significantly in their semantic richness, their approach to multi-lingual representations, and their tokenization strategies. These differences are made visible by the SVD of their equivalent linear mappings and reveal unique styles that likely reflect their underlying training datasets.

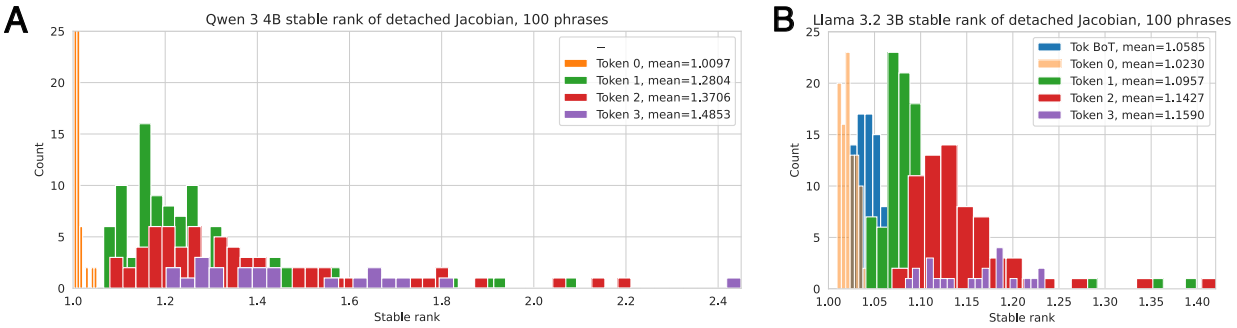

Figure 4: For 100 short input phrases, the stable rank distribution as a function of input token number. Note that Llama 3.2 3B uses a $< |BoT| >$ token and Qwen 3 4B does not.

In terms of their singular value spectra over 100 examples, Fig. 4 shows that both Llama 3 and Qwen 3 are consistently low-rank. The first token for Qwen 3 has a low average rank at 1.01 than Llama 3 at 1.06, but Qwen's next singular vectors are all higher rank than those of Llama. Llama's "beginining of text" token is surprisingly of lower rank than the first text token.

In terms of the semantic content of the singular vectors, both Llama 3 and Qwen 3 employ a similar hierarchical strategy. The first singular vector U_0 with largest magnitude establishes the foundational layer of prediction. This vector primarily contains high-frequency tokens that provide grammatical structure or represent the most probable continuations. For example, in "Should have known," both models place "better" and common punctuation in their U_0 vectors. This shared pattern reinforces the hypothesis that the dominant computational axis in transformers is dedicated to establishing a coherent structural and high-probability scaffold upon which more nuanced semantic meaning can be built. See section A.5 in appendix for more examples of each of these analyses. **21 phrases out of 100** fit this category.

Llama 3 (Abstract Semantics) vs. Qwen 3 (Direct Semantics)

A distinction in semantic processing is pronounced in the secondary singular vectors (U_1 and U_2). Llama 3 consistently demonstrates a rich and abstract English-centric semantic space. For the input "Will break," its U_1 vector contains a diverse set of conceptual possibilities like "confidentiality," "independence," "promises," and "ground." This indicates a capacity to reason about abstract concepts that can be "broken." Qwen 3's vectors for the same phrase are more direct and action-oriented, featuring tokens like "ties," "neck," and "dance," alongside Chinese characters for "stiff" and "can't." This highlights Llama 3's deep modeling of the nuances and abstractions within the English language. **14 phrases out of 100** fit this category.

Qwen 3's Multilingual Reasoning

Perhaps the most obvious difference revealed by this analysis is Qwen 3's multilingual and cross-lingual representation capability, which is largely absent in Llama 3's vectors for the analyzed English prompts. In nearly every example, Qwen 3's secondary vectors are populated with non-English tokens—primarily Chinese, but also Russian and others that are conceptually related to the input phrase. For "The broken," Qwen 3's U_1 vector includes Chinese tokens for "bicycle," "vase," "necklace," and "window"—all concrete examples of breakable objects. This demonstrates that Qwen 3 does not operate in a constrained linguistic space; rather, it accesses a unified, cross-lingual conceptual representation to generate predictions. **38 phrases out of 100** fit this category.

Examples of sub-word Fragments in Qwen 3

We also observed a difference in tokenization and morphological strategy. Qwen 3's secondary vectors frequently contain what appear to be sub-word fragments or tokenization artifacts (e.g., "e," "eer," "ection," "ing"). The persistent recurrence of these tokens, often in the U_2 vector, suggests that part of Qwen 3's

computational process involves constructing or modifying words at a morphological level. This could be an efficient mechanism for handling its multilingual vocabulary. Llama 3 tends to operate with whole-word semantic tokens, indicating a different approach to vocabulary representation. **33 phrases out of 100** fit this category.

3.5 Layer output singular vectors

Table 1 shows the top-8 tokens of the three largest singular vectors of the detached Jacobians of selected layer outputs for Qwen 3 14B. The words "bridge" (and its variants), "highway", "exit", "most" and "only" are highlighted to show their appearances in decoded singular vectors. Early layers are excluded as the tokens are unintelligible. The emergence of intelligible tokens in later layers is shown in the tables as something like a phase change in the representation. Qwen 3 generates infrastructure and engineering related concepts before producing "only".

Fig. 5A shows the normalized singular value spectra of the detached Jacobian at the output of every layer. Llama 3.2 3B has 28 layers, and decoding the largest singular vectors shows that the word representation of these intermediate operations is not interpretable until later layers. From the decoding of the top singular vector by layer, "only" emerges in layer 19. From the map of the progression of the projection of the top two singular vectors onto the top two singular vectors of the last layer in Fig. 5B, we first see a shift at layer 11 toward the prediction.

Since the layer-by-layer operations are only linear and have no higher-order terms, the stable rank $R = (\sum_i^L S_i^2)/S_{max}^2$ serves as a measure of the effectively dimensionality of the subspace of the representation at a particular layer.

When looking at $W_{0_\text{to}_k}$, the cumulative layer transform up through layer k, the dimensionality of the detached Jacobian steadily decreases. When considering each layer i as its own individual transform W_i (where $W_{0_\text{to}_k} = \prod_{i=0}^{k} W_i$ for the simplified scenario of a single input token; there are other cross-token terms not shown here for mid-layer detached Jacobians for longer input sequences), we also see a large peak in dimensionality near the end.

3.6 The detached Jacobian as a conceptual steering operator

Steering vectors are a well-known technique for altering LLM outputs (Liu et al., 2023) where a vector with certain properties is added to a mid-layer representation, and the sum is passed through the rest of the network to generate an output token. Here we utilize the detached Jacobian as an operator instead of an additive vector, and compute it from an intermediate layer for a "steering" phrase like "The Golden Gate" (after the "Golden Gate Claude" demo (Templeton et al., 2024)). The model predicts "Bridge", and this detached Jacobian matrix is used to steer the continuation of a new phrase toward this concept. For a new input phrase, like "Here is a painting of the", the "new" input sequence's embedding vectors $\mathbf{x^*_{new}}$ are multiplied by the detached Jacobian previously computed from the steering concept $\mathbf{J^+_L}(\mathbf{x^*_{steer}})$, scaled by λ and added to the layer activation $\mathbf{f_{Li}}$ from the "new" input.

$$\mathbf{f_{Li}}(\mathbf{x}) = \lambda \cdot \mathbf{f_{Li}}(\mathbf{x^*_{new}}) + (1-\lambda) \cdot \mathbf{J^+_{Li}}(\mathbf{x^*_{steer}}) \cdot \mathbf{x^*_{new}} \tag{16}$$

This steered intermediate representation is then put through the remaining layers of the network and the next token is decoded. The detached Jacobian must only be computed once for the steering concept, and therefore this method is rather efficient. Table 2 shows how the detached Jacobian from an intermediate layer imposes the Golden Gate Bridge as the semantic output coherent with the rest of the input sentence, even when it is difficult to make a logical connection. Beyond demonstrating practical utility, the success of the steering operator provides validation that the detached Jacobian captures actual semantic representations.

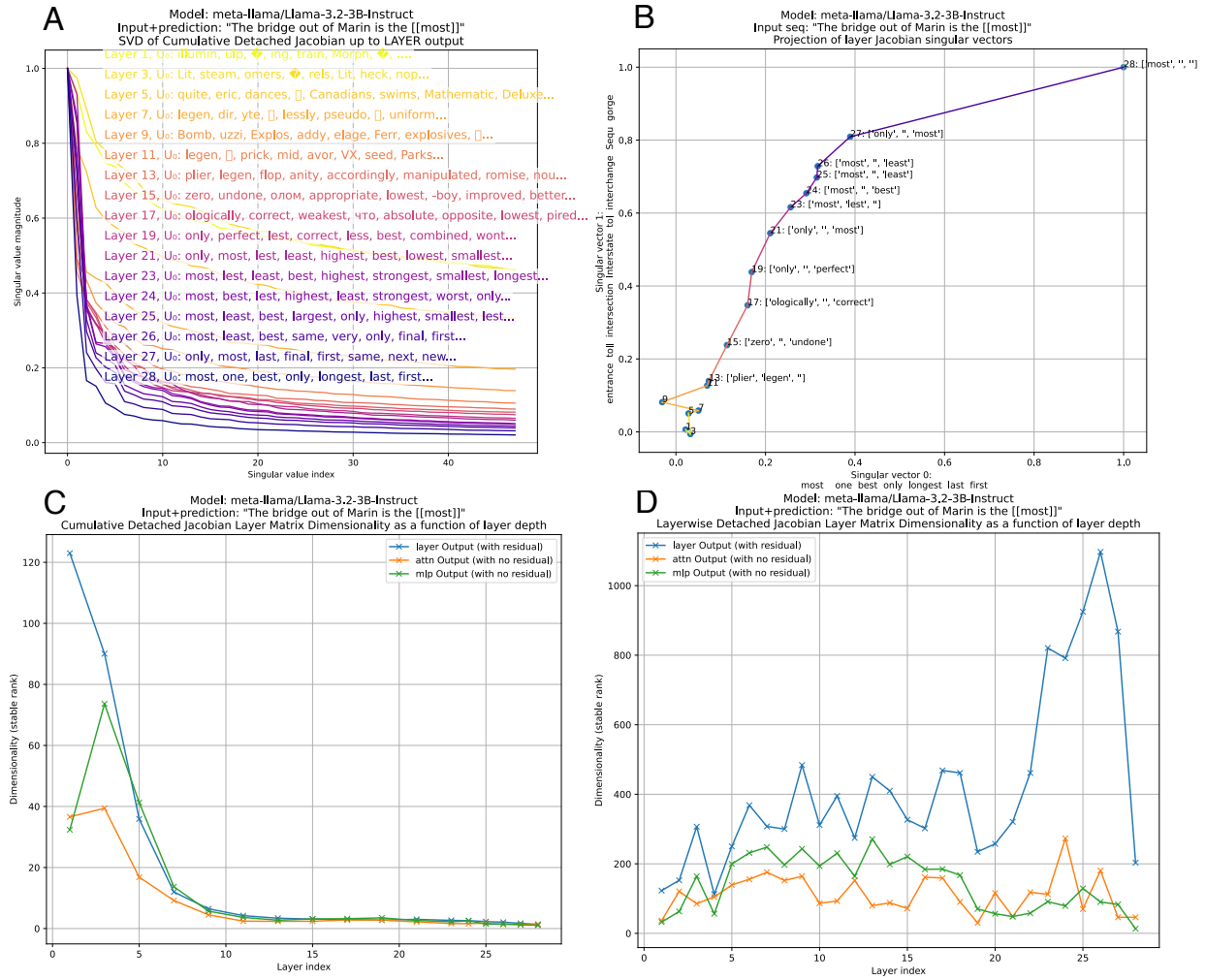

Figure 5: Since the transform representing the model forward operation is linear after detachment, we can also decompose each transformer layer as a linear operation as well. A) The singular value spectrum for the cumulative transform up to layer i. Note that later layers are lower rank than earlier layers. The top singular vectors of the later layers show a clear relation to the prediction of "most". B) The projection of the top two singular vectors onto the top two singular vectors of the final layer. The singular vectors of the first 10 layers are very different than those of the last layer, so the projections remain close to the origin. At layer 11, they begin to approach those of the output layer. C) A measurement of the dimensionality of the cumulative transform up to the output of each layer as the stable rank. Within each layer, the outputs of the attention and MLP modules (prior to adding the residual terms) can also be decomposed as linear mappings. The dimensionality decreases deeper into the network at each of these points, except for a slight increase for the attention and MLP module outputs in layer 3. D) The dimensionality of the detached Jacobian for the layer-wise transform at layer i for the layer output, as well as the attention module output and MLP module output.

4 Discussion

The detached Jacobian approach allows for linear representations of the transformer decoder to be found for each input sequence, without changing the output. The intermediate outputs of each layer and the attention and MLP modules are also accurately reproduced by the detached Jacobian function.

Model	Layer intervention	Input sequence	Normal response	Steered response
Llama 3.1 8B	24 / 36	'I'm going to arizona to see the'	'I'm going to arizona to see the Grand Canyon. I've heard it's a must see. I've also heard it's a bit of a trek to'	'I'm going to arizona to see the Grand Canyon, and I'm planning to hike the Bright Golden Gate Bridge (I think that's the name of the trail) in the Grand Canyon.'
Qwen 3 8B	24 / 36	'Here is a painting of the'	'Here is a painting of the same scene as in the previous question, but now the two people are standing on the same side of the building. '	'Here is a painting of the Golden Gate Bridge in San Francisco. The Golden Gate Bridge is one of the most famous bridges in the world. '
Gemma 3 12B	33 / 48	'I went to new york to see the'	'I went to new york to see the memorial and museum. It was a very moving and emotional experience.'	'I went to new york to see the 10th anniversary of the Broadway show, "The Golden Gate Bridge Bridge." It was a great show.'

Table 2: Detached Jacobian matrices as steering operators, pilot results with Llama 3.1 8B, Qwen 3 8B and Gemma 3 12B.

The detached Jacobian operation is accurate only at the specific operating point at which the matrices were computed by the PyTorch autograd function. A short distance away in the input embedding neighborhood, the detached Jacobian will be extremely different because the manifold is highly curved. (Although local neighborhood validity is less applicable to LLMs which map tokens to embedding vectors, as inputs will only ever discretely sample the embedding space, and there is not an obvious need for exploring the local neighborhood to embedding vectors that do not represent words from the input vocabulary). The manifold is not piecewise linear, but only has a linear equivalent at the operating point, which can be found numerically for every input sequence.

5 Conclusion

This approach to LLM interpretability may scale more effectively than engineering-based methods. Rather than developing increasingly sophisticated techniques to approximate nonlinear behavior, we can compute huge sets of exact linear representations and analyze them systematically. The latter is a scaling issue and data problem for a comprehensible representation, while the former remains a much harder conceptual or mathematical challenge. While our current analysis covers a limited range of examples, the approach suggests a path toward large-scale interpretability by computing the detached Jacobian for every token prediction in major datasets and analyzing the resulting linear systems to understand semantic patterns across diverse contexts. Given the low-rank nature of the detached Jacobian, our Lanczos method which efficiently computes only the top singular vectors is a step toward making this practical. Future work should explore this scaling potential, moving toward comprehensive equivalent linear analysis of LLM behavior across tasks, domains, and model architectures.

References

Randall Balestriero and Richard Baraniuk. Fast jacobian-vector product for deep networks. *arXiv preprint arXiv:2104.00219*, 2021.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Aditya Cowsik, Tamra Nebabu, Xiao-Liang Qi, and Surya Ganguli. Geometric dynamics of signal propagation predict trainability of transformers. *arXiv preprint arXiv:2403.02579*, 2024.

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. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Bobby He and Thomas Hofmann. Simplifying transformer blocks. *arXiv preprint arXiv:2311.01906*, 2023.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Zahra Kadkhodaie, Florentin Guth, Eero P Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representation. *arXiv preprint arXiv:2310.02557*, 2023.

Sheng Liu, Lei Xing, and James Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*, 2023.

Sreyas Mohan, Zahra Kadkhodaie, Eero P Simoncelli, and Carlos Fernandez-Granda. Robust and interpretable blind image denoising via bias-free convolutional neural networks. *arXiv preprint arXiv:1906.05478*, 2019.

Nvidia. Accelerating hugging face llama 2 and llama 3 models with transformer engine. `https://docs.nvidia.com/deeplearning/transformer-engine/user-guide/examples/te_llama/tutorial_accelerate_hf_llama_with_te.html`, 2024.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38, 2019.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. (neurips), 2017. *arXiv preprint arXiv:1706.03762*, 10: S0140525X16001837, 2017.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. *Advances in neural information processing systems*, 32, 2019.

Yaodong Yu, Sam Buchanan, Druv Pai, Tianzhe Chu, Ziyang Wu, Shengbang Tong, Benjamin Haeffele, and Yi Ma. White-box transformers via sparse rate reduction. *Advances in Neural Information Processing Systems*, 36:9422–9457, 2023.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

A Appendix

A.1 Code availability

Code is provided as a zip file (and will be made available on github).

A.2 Pointwise linear GELU

Gemma 3 uses the approximate GELU activation function. Below $\gamma = 0.44715$. Here is the derivation of the pointwise linear version of GELU used for Gemma 3 in the preceding analysis.

$$\mathrm{GELU}\left(\mathbf{x}\right) = \frac{1}{2}\mathbf{x}\left(1 + \tanh\left[\sqrt{2/\pi}\left(x + \gamma\mathbf{x}^3\right)\right]\right) \tag{17}$$

$$\mathrm{GELU}\left(\mathbf{x}\right) = \frac{1}{2}\mathbf{x}\left(1 + \tanh\left[\sqrt{2/\pi}\left(x + \gamma\mathbf{x}^3\right)\right]\right)|_{\mathbf{x}=\mathbf{x}^*} \tag{18}$$

$$\mathrm{GELU}\left(\mathbf{x}^*\right) = ([\frac{\partial}{\partial\mathbf{x}}\mathrm{GELU}(x)]|_{\mathbf{x}=\mathbf{x}^*}) \cdot \mathbf{x}^* \tag{19}$$

A.3 Singular vectors across model families

Fig A2 shows this same analysis for Llama 3, Qwen 3 and Gemma 3 across two different sizes of each. Note the low-rank structure of each of the detached Jacobians, as well as the differing decoding of the top singular vectors from each input embedding vector. The first or "beginning of sequence" token has the highest magnitude in each spectrum reflecting how the positional encoding is entangled with semantic information in the detach Jacobian representation.

A.4 Additional models

Pointwise linearity for Deepseek R1 0528 Qwen 3 8B Distill, Phi 4, Mistral Ministral and OLMo 2 are shown on the following page. See Fig. A3.

A.5 Examples for comparative analysis of singular vectors in Llama 3 and Qwen 3

Shared High-Probability Tokens in U_0

This pattern shows both models using their primary singular vector (U_0) to establish a foundation of common, structurally likely next words.

For the phrase "To see," both models prioritize articles and question words.

- **Qwen 3** U_0: `the a this an all how what and`
- **Llama 3** U_0: `the a , and what an if`

For "To complete," both models identify determiners as the most probable continuations.

- **Qwen 3** U_0: `the a this his an my your`
- **Llama 3** U_0: `this , the a and an (`

For "The final result," the U_0 vectors in both models are dominated by common prepositions and linking verbs that would grammatically follow the phrase.

- **Qwen 3** U_0: `of is in for from after , was`

- **Llama 3** U_0: `, of ... ( is in and`

Both models use their primary singular vector (U_0) to propose very similar sets of common, structurally-likely next words. This highlights a shared foundational strategy of prioritizing grammatical coherence.

21 phrases out of 100 fit this category.

- **Before they:** Both suggest verbs like `were, can, could, start`.
- **While walking:** Both suggest prepositions of movement like `in, through, on, along, around`.
- **To see:** Both prioritize articles (`the, a`) and question words (`what, how`).
- **Will break:** Both suggest particles like `down` and `up`, and articles like `the, a`.
- **Must leave:** Both list determiners (`the, a, this`) and prepositions (`in, at`).
- **Should take:** Both include `a, the, into,` and `care`.
- **After reading:** Both list `the, this, a, about,` and `"`.
- **When finished:** Both suggest `,` and `with`.
- **To begin:** Both prioritize `,` and `with`.
- **May open:** Both suggest `a, the, up,` and `in`.
- **Could drive:** Both include `a, the, in,` and `,`.
- **During lunch:** Both list `„ time,` and `break`.
- **To learn:** Both prioritize `the, more, about,` and `how`.
- **The green:** Both include `and, „ light, is`.
- **The old man:** Both list linking verbs (`was, is`) and conjunctions (`and`).
- **To build they:** Both suggest modal verbs (`have, need, must, would`).
- **The fast car:** Both include `is, has, and, ,`.
- **The tall building:** Both list `is, in, has, with`.
- **To create:** Both prioritize articles `a, an, the`.
- **The response:** Both include `to, is, of`.
- **The solution:** Both list `to, of, is, for`.

Llama 3 (Abstract Semantics) vs. Qwen 3 (Direct Semantics)

This pattern illustrates how Llama 3's secondary vectors often explore a wider and more abstract conceptual space compared to Qwen 3's more direct and action-oriented suggestions.

For the phrase "Should take," Llama 3 suggests abstract responsibilities or concepts one should "take on," while Qwen 3 suggests direct objects or actions.

- **Llama 3** U_1: `utmost admission inspiration revision discipline quitting responsibility guidance`
- **Qwen 3** U_1: `refuge aways -away brib` 半天 (half-day) 午饭 (lunch) `away` 这笔 (this sum)

For "To imagine," Llama 3's vectors include abstract and philosophical concepts to imagine, whereas Qwen 3 focuses on more concrete items like "scenarios."

- **Llama 3** U_1: `reconstruct ethical erect owning peace embodied meanings yourself`
- **Qwen 3** U_1: `scenarios` 场景 (scene) `scenario` 也是一种 (is a kind of) `oha Scenario worlds Scenario`

For "The discovery," Llama 3's vectors describe the impact and nature of a discovery (revolutionary, baffling), while Qwen 3's vectors describe the event of a discovery (a journey, an unintentional bulletin).

- **Llama 3** U_1: `revolution shed bust of details vind catapult baff`
- **Qwen 3** U_1: 震惊 (shock) 轶事 (anecdote) 新西 (New West/Zealand) 了一个 (a) 之旅 (journey) 无意 (unintentional) 快报 (bulletin) 小镇 (small town)

A.5.1 Llama 3 (Abstract Semantics) vs. Qwen 3 (Direct Semantics)

Here, Llama 3's secondary vectors explore broader, more abstract concepts, while Qwen 3's are more concrete and action-oriented.

14 phrases out of 100 show this strong contrast.

- **Will break:** Llama `confidentiality, independence`; Qwen `ties, neck, dance`.
- **Must leave:** Llama `departing, orientation`; Qwen `immediately, room`.
- **Should take:** Llama `admission, inspiration, discipline`; Qwen `refuge, advantage`.
- **The broken:** Llama `fragments, promises, torn`; Qwen `window, clock, vase`.
- **To begin:** Llama `brainstorm, conceptual`; Qwen `start, validate`.
- **May open:** Llama `invitation, plea`; Qwen `windows, sesame`.
- **Could drive:** Llama `distracted, fleets, uninsured`; Qwen `drunk, uphill`.
- **The discovery:** Llama `revolution, catapult`; Qwen `journey, bulletin`.
- **To prevent:** Llama `vulnerability, security`; Qwen `corrosion, fires`.
- **The solution:** Llama `vector, lattice, eigen`; Qwen `set, definition`.
- **To complete:** Llama `projects, tasks`; Qwen `orders, assignment`.
- **Were planning:** Llama `launching, upcoming`; Qwen `permission, meetings`.
- **The evidence:** Llama `overwhelmingly, against`; Qwen `suggests, linking`.
- **To create:** Llama `customized, empowering`; Qwen `custom, interactive`.

Qwen 3's Multilingual Reasoning

This pattern showcases Qwen 3's unique ability to access a cross-lingual conceptual space, populating its secondary vectors with semantically relevant non-English tokens.

For the phrase "The fast car," Qwen 3's U_1 vector includes multiple Chinese words related to speed and motion.

- **Qwen 3** U_1: `overt` 的速度 (speed) 运动 (motion) `.Speed` 追赶 (chase) 速度 (speed) `riages` 超越 (surpass)

For "The fresh bread smelled," Qwen 3's U_2 vector is a list of Chinese synonyms and related concepts for "smell" and "fragrance."

- **Qwen 3** U_2: `smell smells` 嗅 (sniff/smell) 闻 (smell/hear) 香 (fragrant) 香气 (aroma/fragrance) 香味 (fragrance/scent) `smelling`

For "Should help her," the U_1 vector remarkably contains relevant concepts from multiple languages, including Chinese (career development, alleviate), Russian (cope/handle), and Vietnamese (support/help).

- **Qwen 3** U_1: 事业发展 (career development) справиться (handle/cope) hỗ trợ (support/help) 缓解 (alleviate) `unpack` 管理工作 (manage work) 过渡 (transition) 学业 (studies)

38 phrases out of 100 contain clear examples of multilingual reasoning.

Examples of Sub-word Fragments in Qwen 3

For the phrase "While walking," the second singular vector for the token "walking" is almost entirely composed of these fragments, including common suffixes.

- **Vector (Token 1,** U_2**):** `e ection ing eer ignKey cion ging eed`

For "To begin," the U_2 vector includes the common suffixes -ments and -ly, suggesting a mode for building nouns and adverbs.

- **Vector (Token 1,** U_2**):** `e ments eel hips eed s eve ly`

For "The deep water," the U_2 vector for the token "water" contains fragments like -ness and -ection.

- **Vector (Token 2,** U_2**):** `e y eer eel eus ness ection yth`

A.5.2 Qwen 3's "Word-Building" Vector

This category identifies phrases where a secondary Qwen 3 vector is dominated by sub-word fragments and morphological units (e.g., -ing, -tion, -eer, -ness).

33 phrases out of 100 clearly display a dedicated morphological vector.

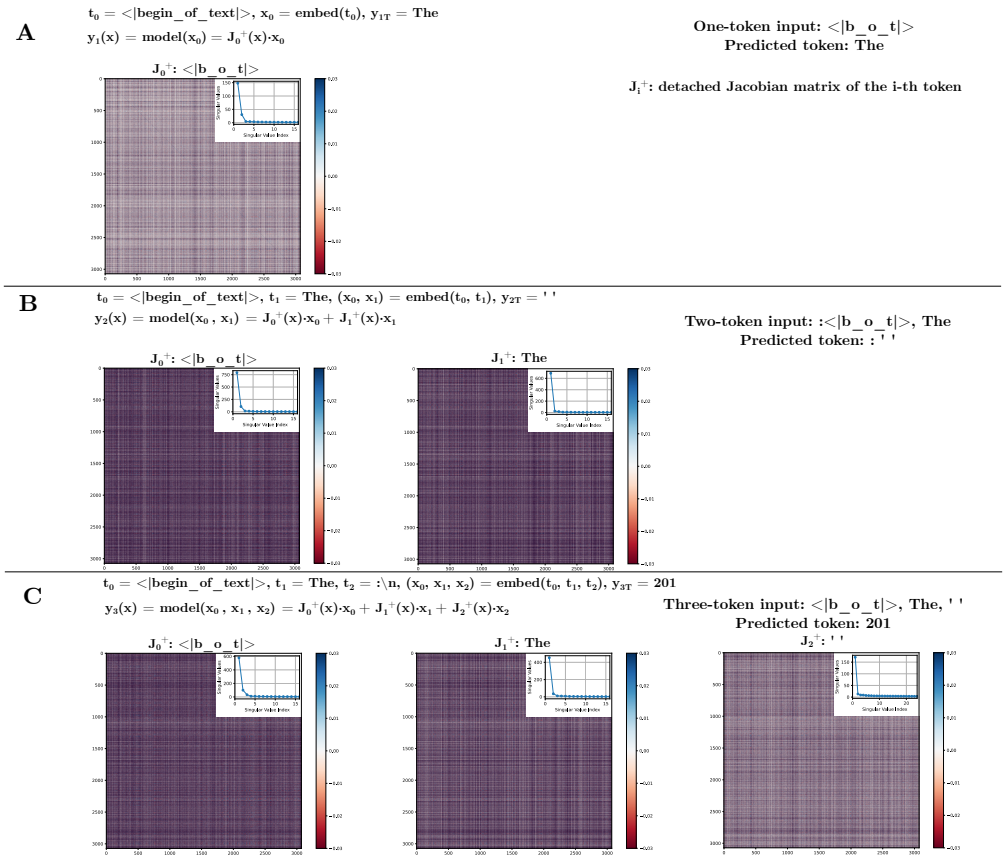

Figure A1: An overview of next-token prediction in the Llama 3.2 3B transformer decoder and decomposition of the predicted embedding vector computation using the detached Jacobian. Generating three tokens with only $< |BoT| >$ as input produces "The 201". For each prediction, each input token $\mathbf{t_i}$ is mapped to an embedding vector $\mathbf{x_i}$, and the network generates the embedding of a next token. The phrase turns out to be "The 2019-2020 season". The detached Jacobian $\mathbf{J}^+(\mathbf{x})$ of the predicted output embedding with respect to the input embeddings is composed of a matrix corresponding to each input vector. Each detached Jacobian matrix $\mathbf{J}_\mathbf{i}^+(\mathbf{x})$ is a function of the entire input sequence but operates only on its corresponding input embedding vector. The matrices tend to be extremely low rank, shown in the inset figures, and the matrix $\mathbf{J}_\mathbf{0}^+$ varies across A), B) and C) above because the input sequences differ. Since the detached Jacobian captures the entirety of the model operation in a linear system (numerically, for a given input sequence), tools like the SVD can be used to interpret the model and its sub-components.

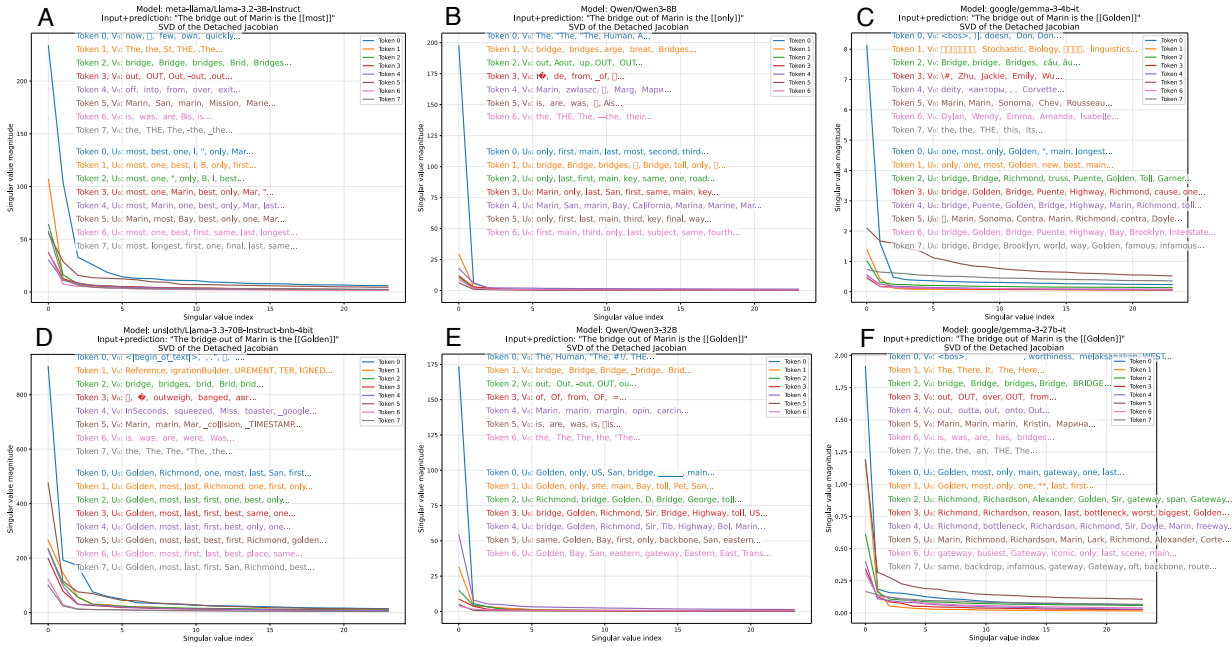

Figure A2: Singular value decomposition of the detached Jacobian for different families and sizes of language models (from 3B to 70B parameters) evaluating the input sequence "The bridge out of Marin is the", followed by a predicted token. The left singular vectors decode to tokens related to bridges and local geography, particularly the Golden Gate Bridge, while singular value spectra all have extremely low rank (see below for quantification). Each row shows top tokens associated with different singular vectors, demonstrating how models encode semantic knowledge about the input sequence and the prediction. See Fig. A3 for Deepseek R1 0528 Qwen 3 8B Distill, Phi 4, Mistral Ministral and OLMo 2.

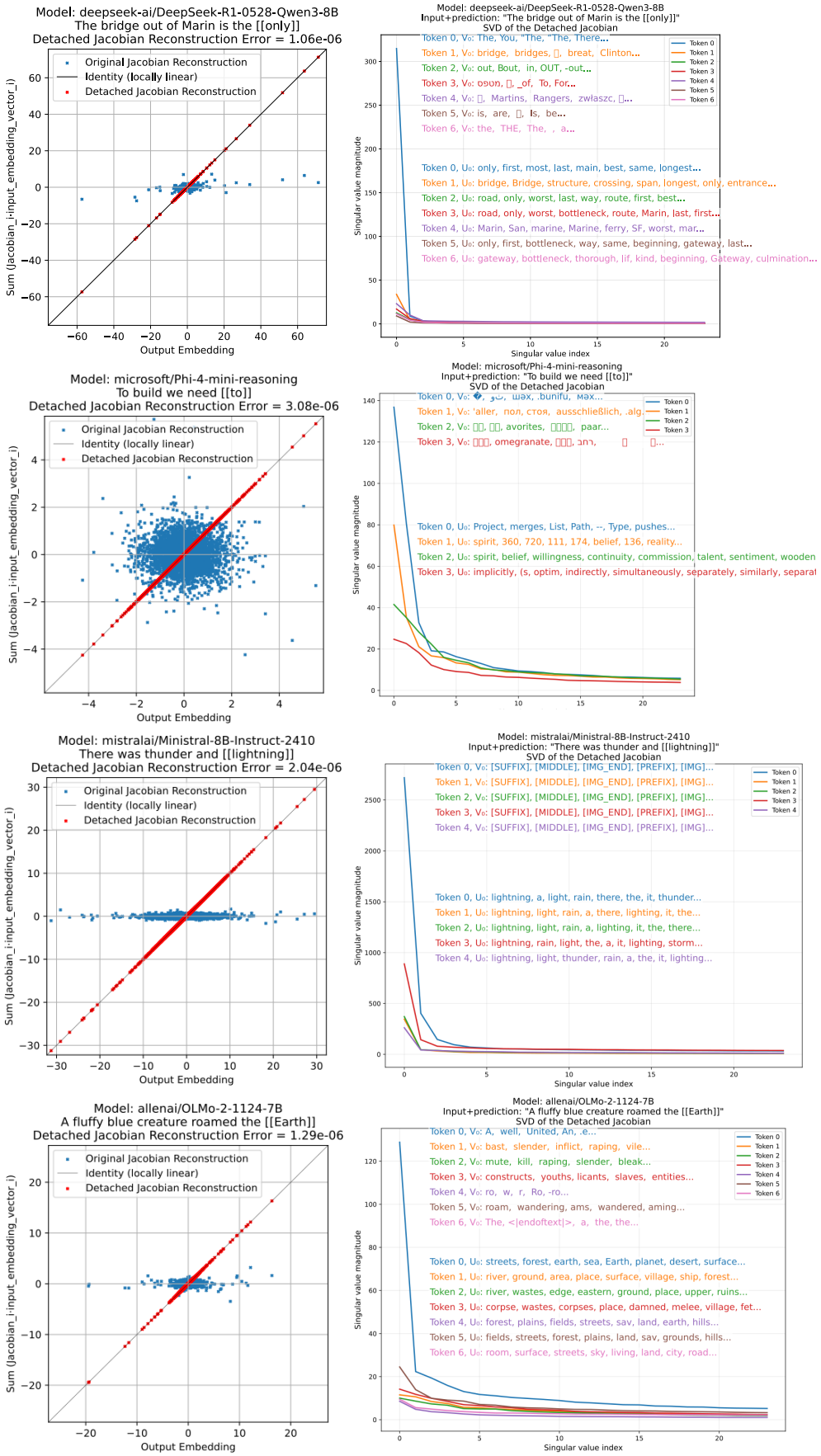

Figure A3: The detached Jacobian reconstruction error and SVD for Deepseek R1 0528 Qwen 3 8B, Phi 4 Mini 4B, Mistral Ministral 8B and OLMo2 7B.

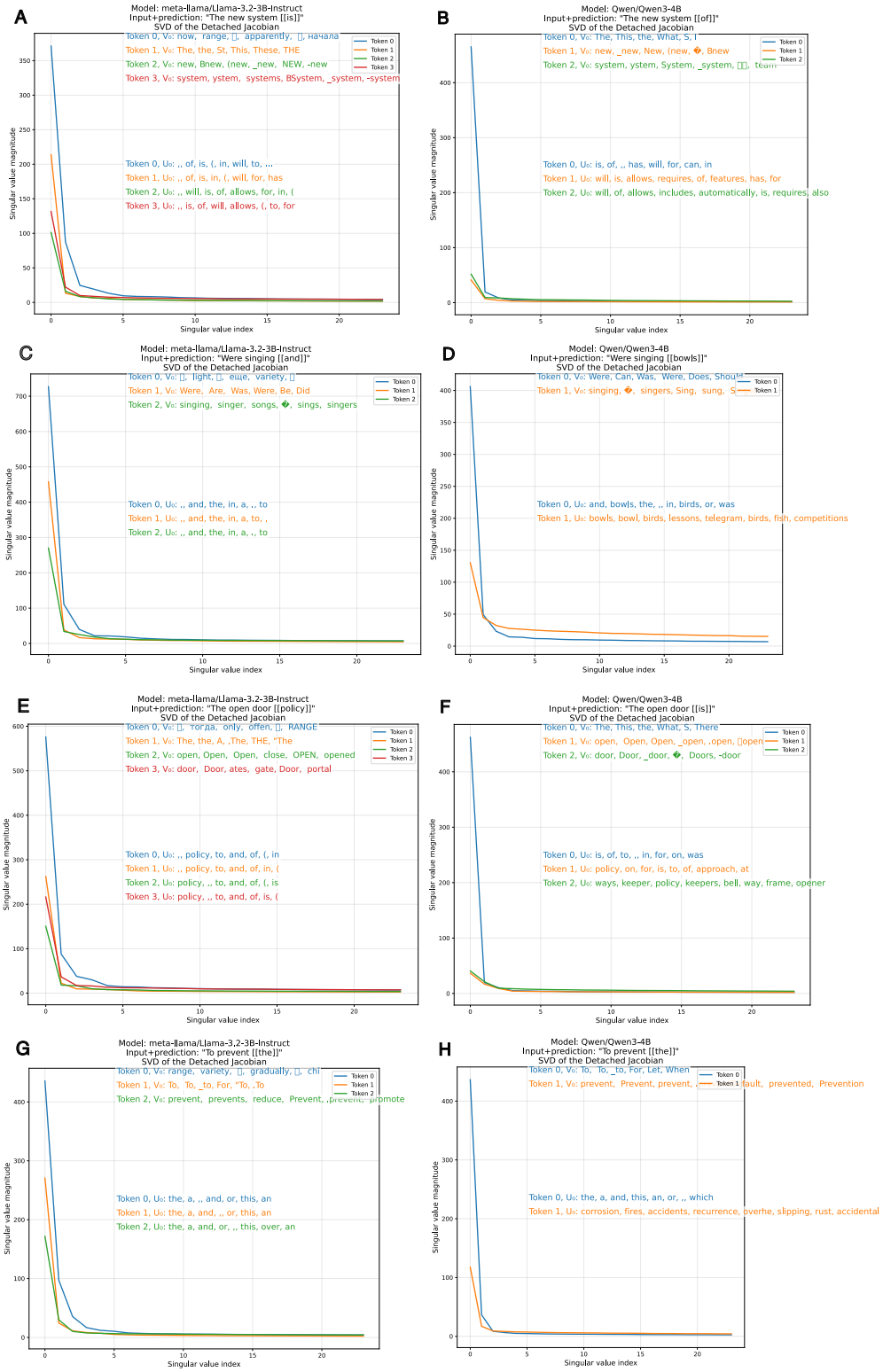

Figure A4: Comparison of detached Jacobians for the same phrase across models.