Large Language Models are Locally Linear Mappings

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 the inference operation of LLMs can be mapped to an equivalent linear system that nearly exactly reconstructs the predicted output embedding for a given input sequence. Extending techniques from image diffusion models that exhibit local or piecewise linearity, we strategically detach components of the gradient computation with respect to an input sequence for a next-token prediction such that the Jacobian of the model reproduces the output with one linear operation per input token. We demonstrate this approach across models, including Qwen 3, Gemma 3, Llama 3, Phi 4, Mistral Ministral and OLMo 2, up to Llama 3.3 70B Q4. With the singular value decomposition of the detached Jacobian, we show that these LLMs operate in extremely low-dimensional subspaces where the largest singular vectors decode to distinct concepts related to possible output tokens. We examine the equivalent linear operation of each successive layer (and its attention and multilayer perceptron components) and observe the emergence of semantic concepts. We demonstrate that the detached Jacobian of middle layer representations can be used as a steering operator to insert semantic concepts into unrelated text, which could be useful for improving safety and decreasing bias. Despite their expressive power and global nonlinearity, modern LLMs can be interpreted through locally linear decompositions that provide insights into their internal representations and 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 nearly exactly reconstructs the predicted output embedding for a given input sequence.

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 can be made exactly locally or piecewise linear through several architectural constraints and interpreted as low-dimensional adaptive linear filters.

We extend these ideas by demonstrating local linearity 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 and also almost exactly locally linear. This numerical Jacobian computation captures the complete forward operation of the model, including activation functions and attention, although it is only valid for that particular input sequence (more "pointwise" linear than piecewise linear).

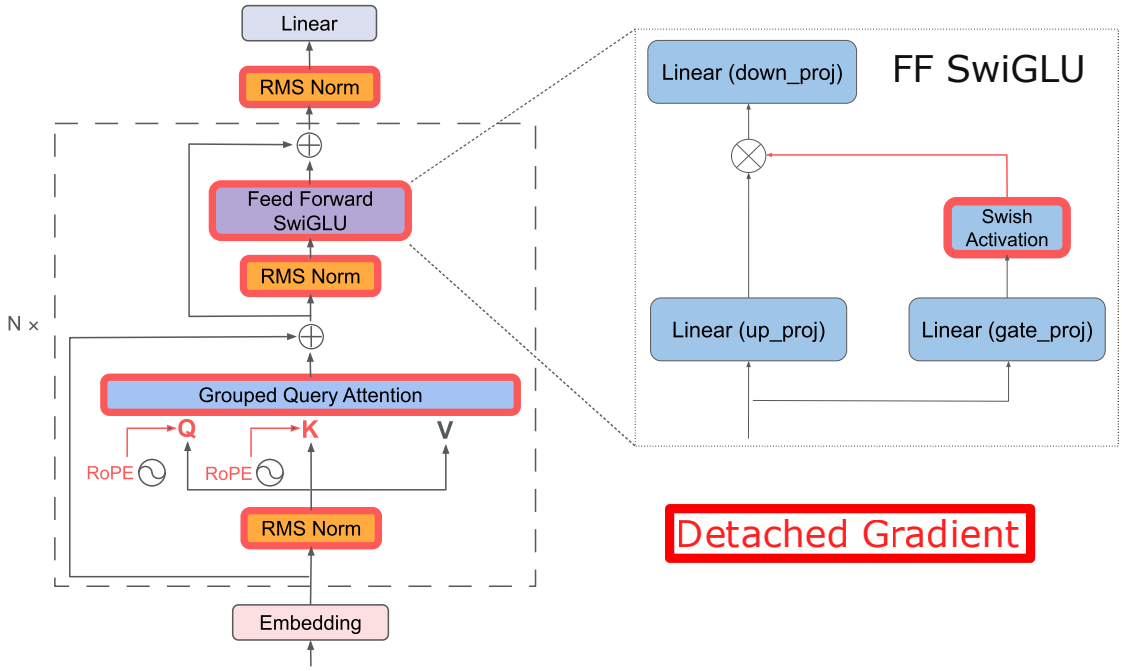

Figure 1: A schematic of the Llama 3 transformer decoder (Grattafiori et al., 2024; Nvidia, 2024). The gradient *detach* operations for components outlined in red effectively freeze the nonlinear activations for a given input sequence, creating a linear path for the gradient with respect to the input embedding vectors, but do not change the output. The output embedding prediction can be nearly exactly mapped to a 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 local 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 equivalent linear systems 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 we 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 local linearity in model families including Qwen 3, Gemma 3, Llama 3, Phi 4, Mistral Ministral, Deepseek R1 0528 Qwen 3 8B and OLMo 2, at a range of sizes up to Llama 3.3 70B Q4. This approach does not require further model training, offering a new path to interpreting a wide range of open-weight LLMs at a local level that could serve as a complement to other powerful interpretability methods.

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

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

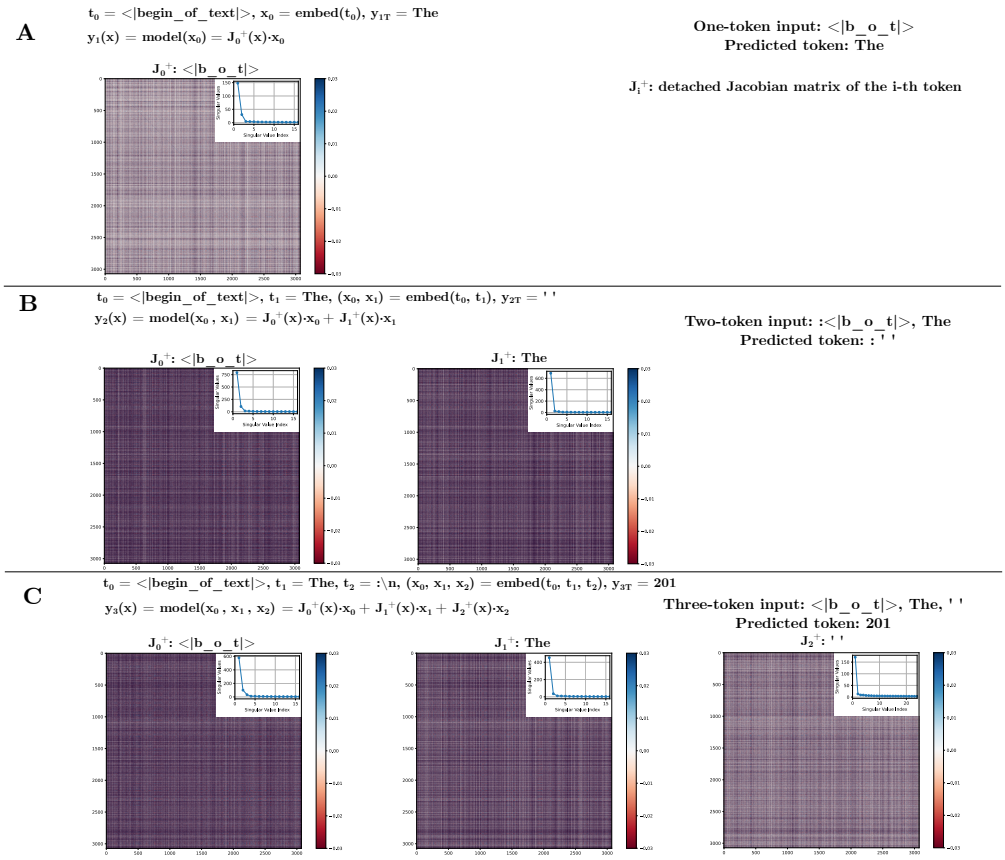

Figure 2: An overview of next-token prediction in the Llama 3.2 3B transformer decoder and decomposition of the predicted embedding vector computation using the detached Jacobian. Generating three tokens with only $< |BoT| >$ as input produces "The 201". For each prediction, each input token $\mathbf{t_i}$ 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 $\mathbf{J_i^+}(\mathbf{x})$ 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 $\mathbf{J_0^+}$ 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.

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).

2.2 The Jacobian of a transformer decoder

Many open weight LLMs also use linear layers with zero bias, like the *ReLU* network 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 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, zero-bias $RMSNorm$ layers and softmax attention blocks. We further demonstrate that for a given input sequence the entire transformer decoder is an adaptive homogeneous function of order 1 where we can apply necessary gradient detachments and numerically compute a linear system that nearly exactly 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).

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. Fig. 2 shows how each matrix of the detached Jacobian operates on its corresponding input embedding vector to provide a nearly-exact reconstruction of the LLM forward operation (with a relative error of 10^{-6} (the standard deviation of the reconstruction error divided by the standard deviation of the output embedding), see Fig. 3).

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

While the traditional Jacobian $\mathbf{J}$ for a particular input sequence $\mathbf{x}^*$ is a locally linear approximation of the nonlinear LLM forward operation, it does not generate an exact reconstruction at $\mathbf{x}^*$ since the transformer function is not linear. The detached Jacobian $\mathbf{J}^+$ evaluated at $\mathbf{x}^*$ is the result of an alternative gradient path through the same network which is linear for the input $\mathbf{x}^*$. The detached Jacobian $\mathbf{J}^+$ only generates a near-exact reconstruction at $\mathbf{x}^*$ and not in the local neighborhood due to the strong nonlinearity 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 locally linear operations

In order to achieve local 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) 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 disconnects the variance term from the network's computational graph. Their batch-norm layer returns the same values as the standard batch-norm layer, but it is locally linear at inference as the nonlinear operation is removed from the gradient computation.

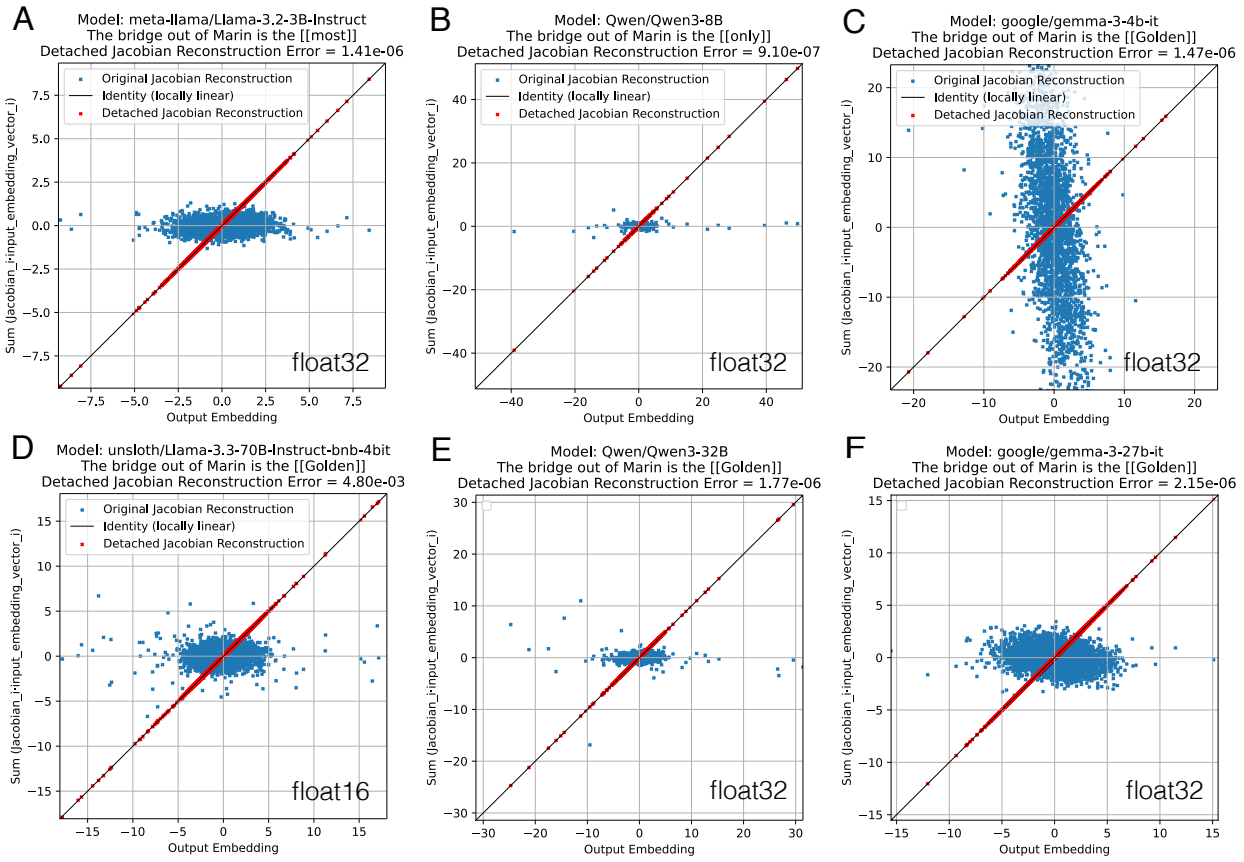

Figure 3: For the input sequence "The bridge out of Marin is the", the elements of the predicted output embedding vector of the model compared to the elements from the Jacobian reconstruction for both the original Jacobian (blue points) and detached Jacobian operations (red points), shown for a range of model families and sizes. Note that the detached Jacobian reconstructions nearly exactly match the predicted embedding in each case, with relative error (the standard deviation of the reconstruction error divided by the standard deviation of the output embedding) on the order of 10^{-3} for $float16$ precision and 10^{-6} for $float32$ precision. Llama 3 models are shown in A) and D), Qwen 3 models in B) and E), and Gemma 3 models in C) and F). These plots demonstrate the near-exact reconstruction of the detached Jacobian and therefore the local linearity of the mapping.

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 $autograd.functional.jacobian$. The gradient of the function is then computed at $\mathbf{x}^*$ (we assume for simplicity a sequence of length 1).

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

We define the detached Jacobian as follows:

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

We can rewrite the locally linear $RMSNorm$ as follows:

$$norm_{LL}(\mathbf{x}^*) = \mathbf{J}^{+}_{\mathbf{n_{LL}}}(\mathbf{x}^*) \cdot \mathbf{x}^* \tag{6}$$

At inference, we now have a locally 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 $\mathbf{J}^{+}_{\mathbf{n_{LL}}}$, which we can use to reconstruct the normalization output exactly as a linear system; no higher-order terms are needed.

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 locally linear, 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 local linearity, Llama 3.2 uses $SwiGLU$ (Shazeer, 2020), Gemma 3 uses approximate $GELU$ (Hendrycks & Gimpel, 2016) and Qwen 3 uses $Swish$ for activation functions. Fortunately, 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 locally linear $SwiGLU$ layer with respect to the input $\mathbf{x}$. Below, $Swish(\mathbf{x}) = \mathbf{x} \cdot sigmoid(\mathbf{x})$ and $\otimes$ is element-wise multiplication.

$$\text{SwiGLU}\left(\mathbf{x}\right) = \text{Swish}\left(\mathbf{W}\mathbf{x}\right) \otimes \left(\mathbf{Z}\mathbf{x}\right) \tag{7}$$

$$\text{SwiGLU}_{LL}\left(\mathbf{x}\right) = [\text{Swish}\left(\mathbf{W}\mathbf{x}\right)]|_{\mathbf{x}=\mathbf{x}^*} \otimes \left(\mathbf{Z}\mathbf{x}\right) \tag{8}$$

$$\text{SwiGLU}_{LL}\left(\mathbf{x}^*\right) = ([\frac{\partial}{\partial \mathbf{x}} \text{SwiGLU}_{LL}(x)]|_{\mathbf{x}=\mathbf{x}^*}) \cdot \mathbf{x}^* \tag{9}$$

$$\text{SwiGLU}_{LL}\left(\mathbf{x}^*\right) = \mathbf{J}^{+}_{\mathbf{SwiGLU_{LL}}}(\mathbf{x}^*) \cdot \mathbf{x}^* \tag{10}$$

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

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, $\mathbf{Q} = \mathbf{W_Q}\mathbf{x}$, $\mathbf{K} = \mathbf{W_K}\mathbf{x}$ and $\mathbf{V} = \mathbf{W_V}\mathbf{x}$.

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

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

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

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

The linear $\mathbf{x}$ term within $\mathbf{V}$ makes it possible for the attention block to be locally 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 locally 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.

3 Results

3.1 Local linearity of the predicted output

In order to validate whether the detached Jacobian achieves local linearity, we can compare the predicted output embedding vector for a given input token sequence to the Jacobian reconstruction of the output.

Fig. 3 examines the original output and the output when incorporating the appropriate gradient detachments for local linearity for two sizes of Llama 3, Qwen 3 and Gemma 3. The reconstruction of the output embedding with the detached Jacobian matrices falls close to the identity line when compared with the output embedding, which is evidence for local linearity. The same comparison with the original Jacobian does not fall on the identity line. The standard deviation of the difference for the detached Jacobian divided by the standard deviation of the output embedding vector is on the order of only 10^{-6}, which quantifies how exactly the detached Jacobian reproduces the output). These plots therefore validate the local linearity of Llama 3, Qwen 3 and Gemma 3 with the appropriate gradient detachments for a particular input.

The numerical computation of the top k singular vectors of the Jacobian takes on the order of 10 seconds for an input sequence of 8 tokens for Llama 3.2 3B on a T4 GPU. At the other end of the spectrum, the singular vectors for the same sequence with Llama 3.3 70B Q4 on an H100 GPU takes more than a minute. (A method for computing the top singular vectors without forming the full matrix with Lanczos iteration is under development.)

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 mangitude 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. 4 (in appendix A below) 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. 4A.

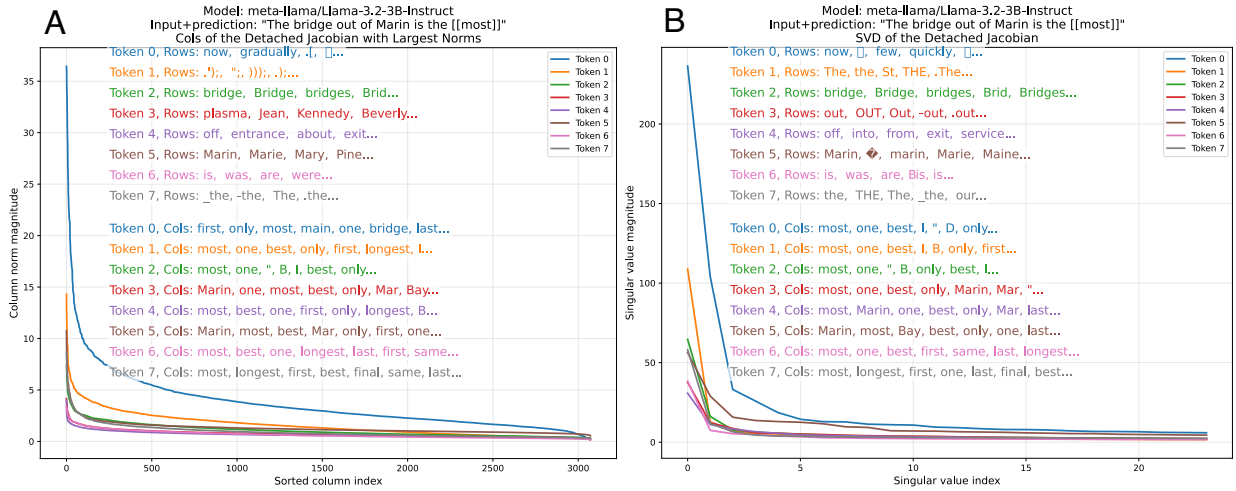

Figure 4: Given the sequence “The bridge out of Marin is the", the most likely prediction is “most" for Llama 3.2 3B. The detached Jacobian matrices for each token represent a near-exact local linearization of the predicted output embedding. A) We show the features which drive large responses in single 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 blue list of words at the top are the ordered decoded “feature" input tokens from the largest rows of the detached Jacobian matrix for the beginning of sequence token, and the different colors show the decoded feature tokens for the other 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 features 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 left and right singular vectors U and V to output and input embeddings, shown in colors. The singular value spectrum is extremely low rank, and decoding the U singular vectors returns candidate output tokens: “most" and “one" appear frequently. 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 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. 4B, the singular vectors are decoded for three different models of two sizes each, from $3B$ to $70B$ parameters. 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 almost identical 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.

3.4 Singular vectors across model families

Fig 5 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

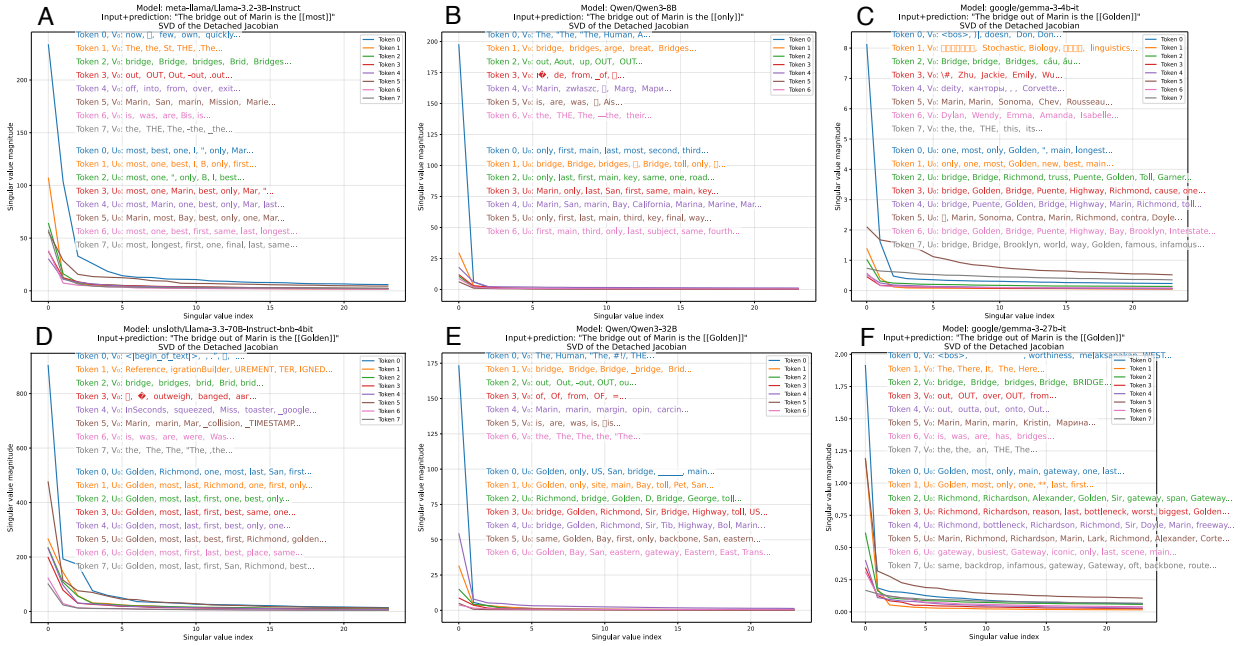

Figure 5: Singular value decomposition of the detached Jacobian for different families and sizes of language models (from 3B to 70B parameters) evaluating the input sequence "The bridge out of Marin is the", followed by a predicted token. The left singular vectors decode to tokens related to bridges 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. A1 for Deepseek R1 0528 Qwen 3 8B Distill, Phi 4, Mistral Ministral and OLMo 2.

magnitude in each spectrum reflecting how the positional encoding is entangled with semantic information in the detach Jacobian representation.

3.5 Layer output singular vectors

Tables 1, 3 and 2. show the decodings of the top three singular vectors of the detached Jacobians of the layer outputs for Llama 3.1 8B, Gemma 3 4B and Qwen 3 8B. Similar patterns emerge across models: the words "bridge" (and its variants), "Golden", "highway", "exit", "most" and "only" are highlighted to show their appearances in singular vector decodings. Early layers are excluded as the decodings are unintelligible. The emergence of intelligible tokens in later layers is shown in the tables as something like a phase change in the representation. Llama 3.1 shows a progression from generic bridge terms to specific San Francisco geographic terms and generating "Golden". Gemma 3 includes more diverse geographic terms including international locations but still results in "Golden". Qwen 3 produces more infrastructure and engineering concepts before producing "only".

Fig. 6A shows the normalized singular value spectra of the detached Jacobian at the output of every layer. Llama 3.2 3B has 28 transformer 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. 6B, 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 $W_{0\_to\_k}$, 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 $W_{0\_to\_k} = \prod_{i=0}^{k} W_i$ 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.

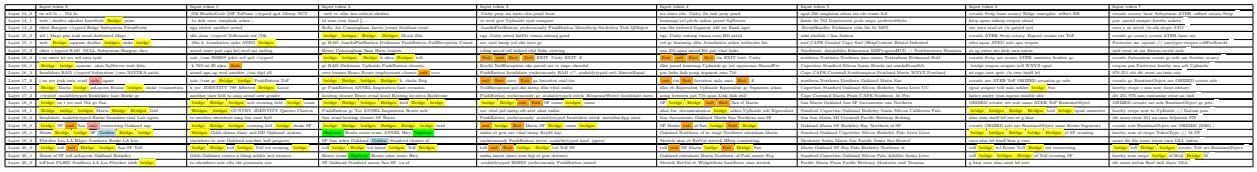

Table 1: The top three singular vectors of the detached Jacobian for the layer outputs from Llama 3.1 8B for the sequence "The bridge out of Marin is the" with the prediction [[Golden]]. Legend: "Bridge", "only", "highway", "exit", "Golden", "most". This is a selection of the middle layers, see A.4 for full tables that extend to the last layer.

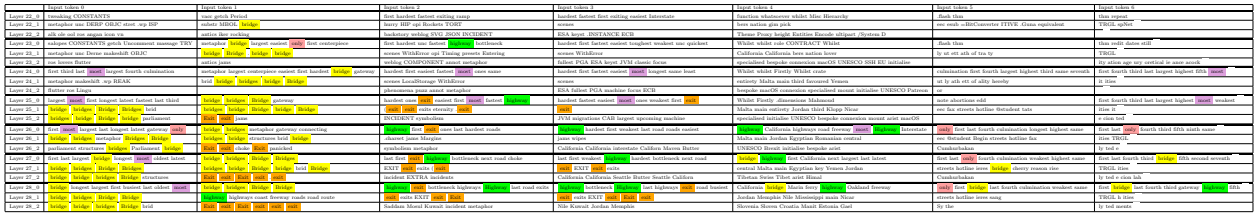

Table 2: The top three singular vectors of the detached Jacobian for the layer outputs from Qwen 3 8B for the sequence "The bridge out of Marin is the" with the prediction [[only]]. Legend: "Bridge", "only", "highway", "exit", "Golden", "most".

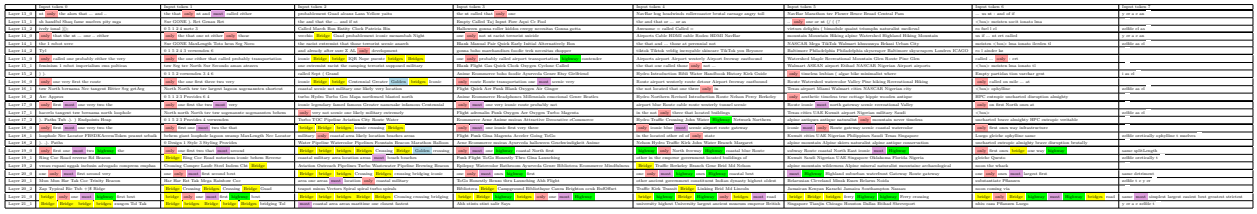

Table 3: The top three singular vectors of the detached Jacobian for the layer outputs from Gemma 3 4B for the sequence "The bridge out of Marin is the" with the prediction [[Golden]]. Legend: "Bridge", "only", "highway", "exit", "Golden", "most".

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). Here we utilize the detached Jacobian from an intermediate layer for a phrase like "The Golden Gate", after the "Golden Gate Claude" demo (Templeton et al., 2024). The model predicts "Bridge", and this detached Jacobian is used as an operator 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 embedding vectors $\mathbf{x}^*_{\mathbf{new}}$ are scaled by λ and multiplied by the detached Jacobian for the steering concept $\mathbf{J}^+_{\mathbf{L}}(\mathbf{x}^*_{\mathbf{steer}})$, and added to the layer activation $\mathbf{f}_{\mathbf{Li}}$ from the new input.

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

This intermediate representation is then fed back 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

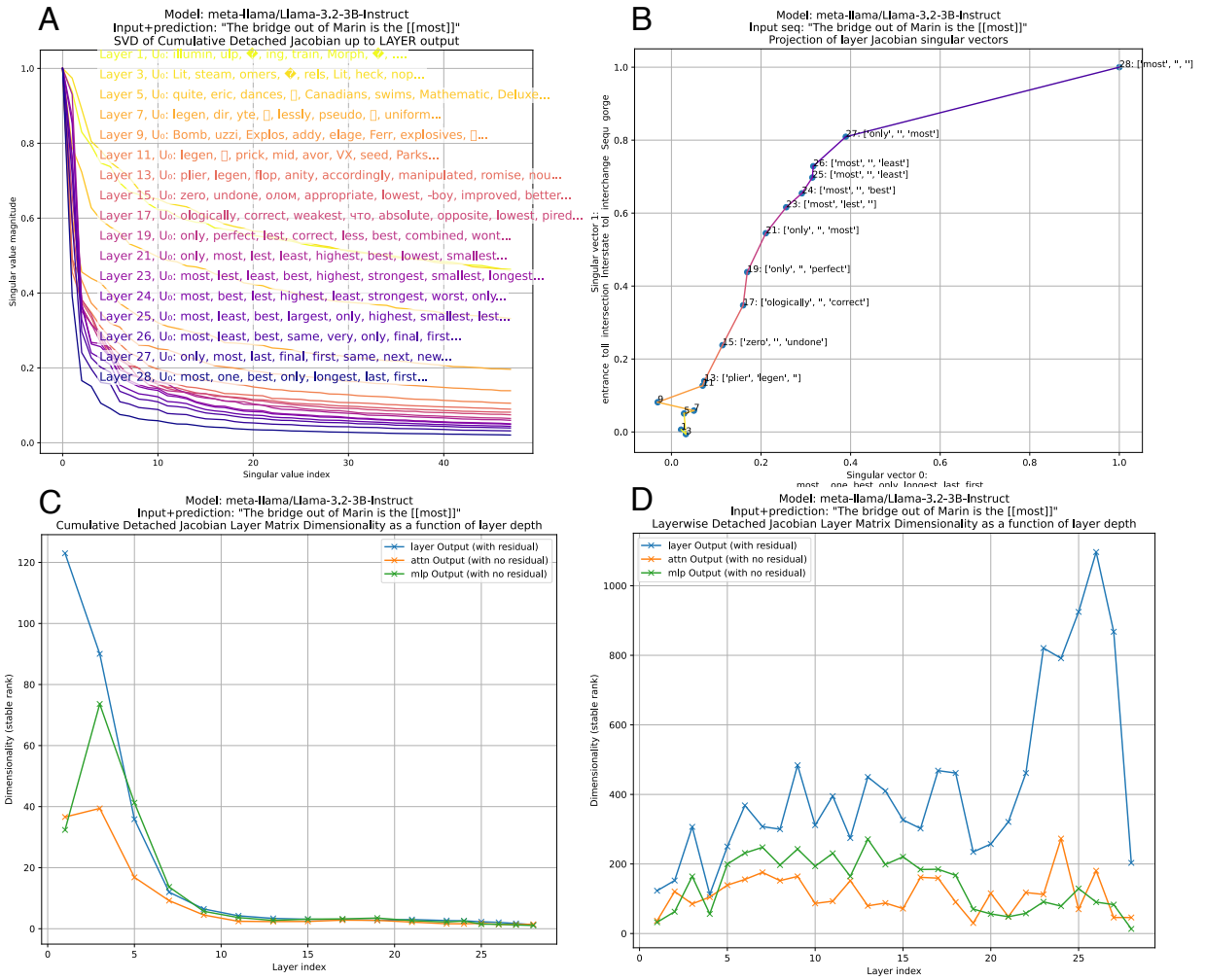

Figure 6: Since the transform representing the model forward operation is locally linear, we can also decompose each transformer layer as a linear operation as well. A) The singular value spectrum for the cumulative transform up to layer i. Note that later layers are lower rank than earlier layers. The 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.

this method is rather efficient. Table 4 shows how the detached Jacobian from an intermediate layer imposes the Golden Gate Bridge as the semantic output coherent with the rets 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.

4 Discussion

The detached Jacobian approach allows for locally linear representations of the transformer decode to be found for each input sequence, without changing the output. The intermediate outputs of each layer and sub-component are also exactly reproduced by the detached Jacobian function.

The detached Jacobian operation is only nearly exact at the specific operating point at which the matrices were computed by autograd. A short distance away in the input embedding neighborhood, the detached Jacbian 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 are not obvious needs for exploring the local neighborhood beyond token embedding vectors). The manifold is not piecewise linear, but only has a linear equivalent exactly at the operating point, which can be found numerically for every input sequence.

Model	Layer intervention	Input sequence	Normal response	Steered response
Llama 3.1 8B IT	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 IT	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 IT	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 4: Detached Jacobian matrices as steering operators, pilot results with Llama 3.1 8B, Qwen 3 8B and Gemma 3 12B.

5 Conclusion

We have demonstrated that a number of open-weight LLMs can be made locally linear at inference, where a set of detached Jacobian matrices completely describes the forward computation, and that the predicted output embedding is unchanged from that of the original model. One potential safety application of this method is to examine large singular vectors for bias, misinformation or toxic content. The initial demonstrations of model steering may eventually be useful as a way to intervene and elicit safer outputs. Although this method requires intensive computation for decomposing single-token predictions, it should be possible to scale this investigation to well-known LLM datasets.

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.

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 Locally linear approximate GELU

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

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

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

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

A.2 Code availability

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

A.3 Additional models

Local linearirty for Deepseek R1 0528 Qwen 3 8B Distill, Phi 4, Mistral Ministral and OLMo 2 are shown on the following page. See Fig. A1.

A.4 Semantic Emergence in Transformer Layers

The first singular vectors of the detached Jacobians for the layer outputs from Llama 3.1 8B, Gemma 3 4B and Qwen 3 8B for the sequence “The bridge out of Marin is the” are shown below on the following page. These are expanded versions of Tables 1, 2 and 3 from the main text.

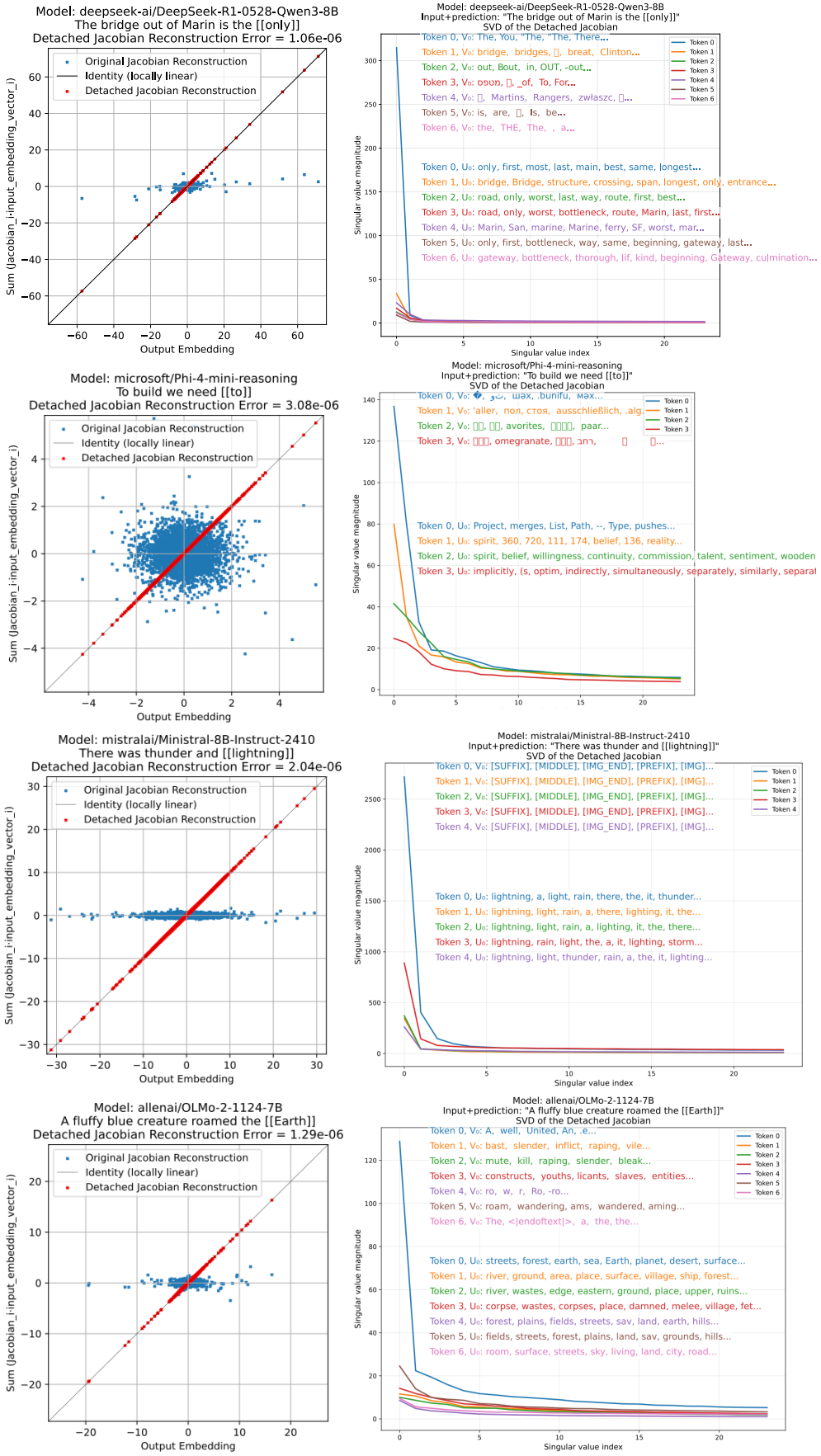

Figure A1: The detached Jacobian reconstruction error and SVD for Deepseek R1 0528 Qwen 3 8B, Phi 4 Mini 4B, Mistral Ministral 8B and OLMo2 7B.

Legend: “Bridge”, “only”, “highway”, “exit”, “Golden”, “most”.

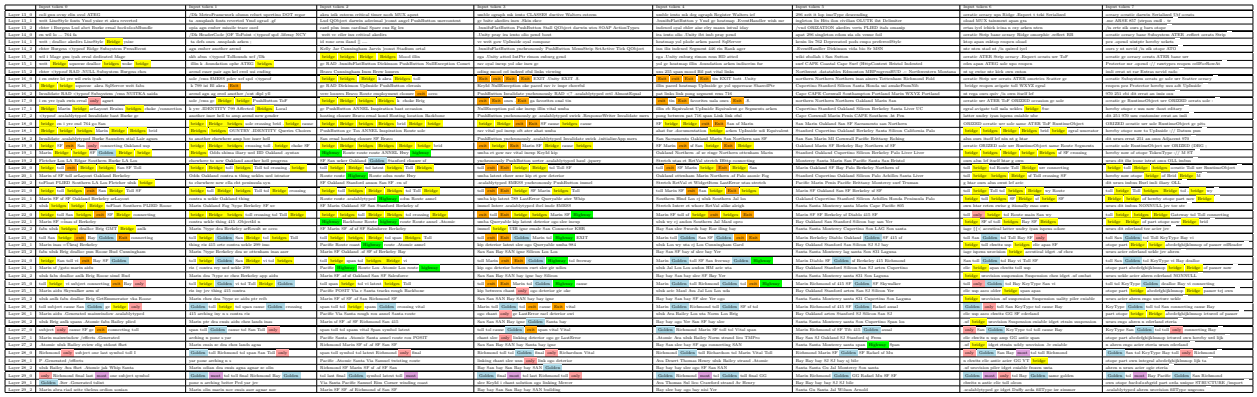

Table 5: The top three singular vectors of the detached Jacobian for the layer outputs from Llama 3.1 8B for the sequence “The bridge out of Marin is the” with the prediction [[Golden]].

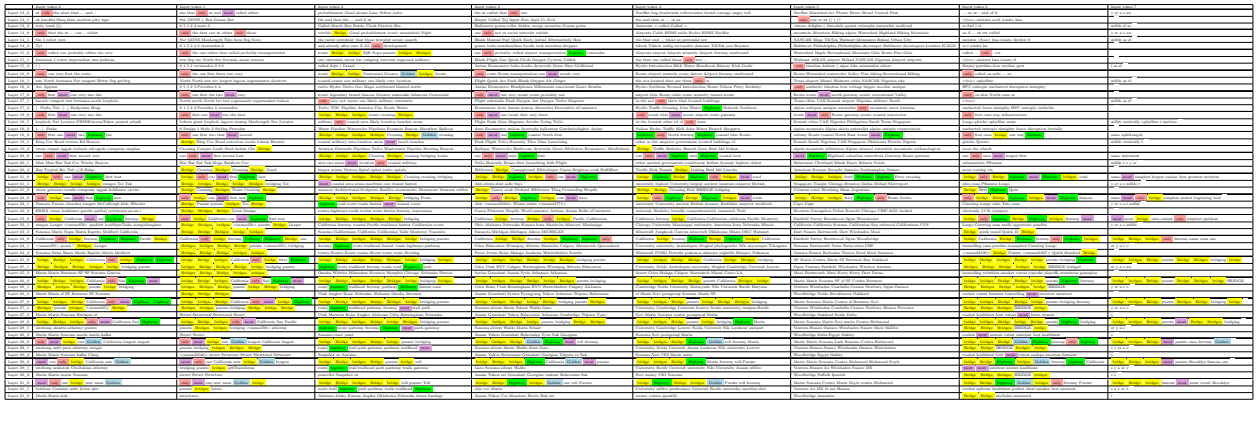

Table 6: The top three singular vectors of the detached Jacobian for the layer outputs from Gemma 3 4B for the sequence “The bridge out of Marin is the” with the prediction [[Golden]].

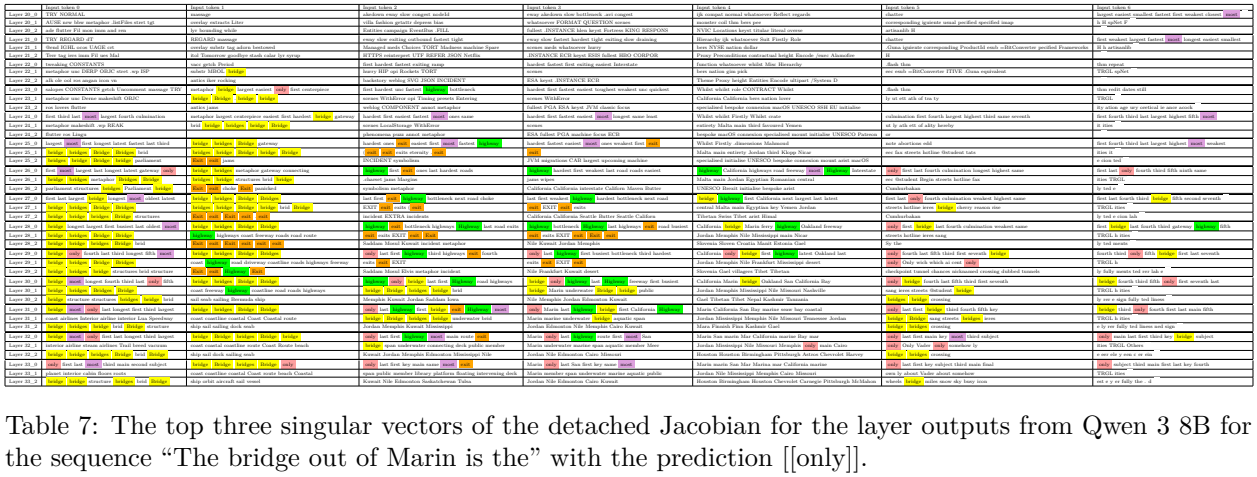

Table 7: The top three singular vectors of the detached Jacobian for the layer outputs from Qwen 3 8B for the sequence “The bridge out of Marin is the” with the prediction [[only]].