TensorLens: End-to-End Transformer Analysis via High-Order Attention Tensors

Ido Andrew Atad **Itamar Zimerman** **Shahar Katz** **Lior Wolf**
Blavatnik School of Computer Science and AI, Tel Aviv University
{idoatad,zimerman1,shaharkatz3}@mail.tau.ac.il, wolf@cs.tau.ac.il

Abstract

Attention matrices are fundamental to transformer research, supporting a broad range of applications including interpretability, visualization, manipulation, and distillation. Yet, most existing analyses focus on individual attention heads or layers, failing to account for the model's global behavior. While prior efforts have extended attention formulations across multiple heads via averaging and matrix multiplications or incorporated components such as normalization and FFNs, a unified and complete representation that encapsulates all transformer blocks is still lacking. We address this gap by introducing TensorLens, a novel formulation that captures the entire transformer as a single, input-dependent linear operator expressed through a high-order attention-interaction tensor. This tensor jointly encodes attention, FFNs, activations, normalizations, and residual connections, offering a theoretically coherent and expressive linear representation of the model's computation. TensorLens is theoretically grounded and our empirical validation shows that it yields richer representations than previous attention-aggregation methods. Our experiments demonstrate that the attention tensor can serve as a powerful foundation for developing tools aimed at interpretability and model understanding.

https://github.com/idoatad/TensorLens

1 Introduction

Transformer-based architectures (Vaswani et al., 2017) have revolutionized deep learning by exhibiting remarkable scaling properties, enabling effective models with millions or even billions of parameters that can be trained on extensive datasets containing trillions of tokens. This advancement has led to breakthroughs that include large language models (LLMs) such as ChatGPT (Brown et al., 2020), Vision Transformers (Dosovitskiy et al., 2020), Diffusion Transformers (Peebles and Xie, 2023), and others. The core component of the Transformer responsible for capturing interactions between tokens is the self-attention mechanism.

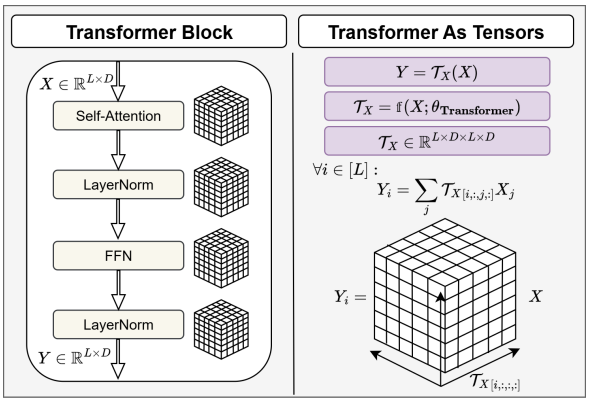


Figure 1: Transformers are re-formulated as data-controlled linear operators, characterized by an input-dependent high-order attention tensor $\mathcal{T}$. This formulation enables a unified self-attention representation that captures the entire Transformer architecture, including sub-components such as FFN layers, normalization, embedding layers, and residual connections.

Self-attention can be viewed as a data-controlled linear operator (Poli et al., 2023; Massaroli et al., 2020) that is represented by an input-dependent attention matrix. Due to the row-wise softmax normalization, these attention matrices are somewhat interpretable, offering insight into how each layer updates the output representations through a weighted linear combination of input value vectors. As a result, attention matrices have been employed in a wide range of research domains, including (i) explainability and interpretability through attribution methods and model analysis (Abnar and Zuidema, 2020; Katz et al., 2024), (ii) model editing and intervention techniques (Chefer et al., 2023; Katz and Wolf, 2025; Ali et al., 2025a), (iii) distillation and training techniques (Touvron et al.,

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3452–3468

2021; Zhang et al., 2024), (iv) inductive bias and regularization methods (Li et al., 2018; Attanasio et al., 2022; Zimerman and Wolf, 2024), among others.

To push these applications further, substantial effort has been invested in developing extended representations of attention that go beyond individual attention matrices. A prominent example is the attention rollout technique (Abnar and Zuidema, 2020), which averages attention matrices across heads within the same layer and then integrates across the layers by applying multiplication. More recent approaches propose improved aggregation methods across heads. For example, Kobayashi et al. (2020) leverages the output projection layer to aggregate heads more precisely, and subsequent work incorporates the feed-forward layer into the formulation (Kobayashi et al., 2023). Additionally, in non-transformer models, implicit attention formulations have been introduced even for several architectures, including Mamba (Ali et al., 2025b; Dao and Gu, 2024), RWKV and Griffin (Zimerman et al., 2025), and others. Following this line of work, we ask: What is the most comprehensive formulation that attention can encompass in Transformers? Is it possible to represent the entire Transformer as a data-controlled linear operator that captures all of its parameters and is theoretically grounded, rather than relying on heuristically aggregated attention matrices?

We fundamentally address this question by reformulating the entire Transformer model, including all of its components (feed-forward networks (FFNs), activation functions, LayerNorm, skip connections, embedding layers, and others) as a single data-controlled linear operator as visualized in Figure 1. A key insight of our work is that such a formulation requires high-order tensor attention tensors, not just matrices, to fully encompass the model's behavior. Our formulation is theoretically grounded, and our empirical analysis shows that it better reflects the model than previously proposed attention forms. Moreover, we demonstrate that the tensor structure can approximate linear relations (Hernandez et al., 2024) better than the matrix alternatives, underscoring its capacity to reveal LLM functionalities previously explored in mechanistic interpretability research.

Our main contributions are as follows: (i) We introduce TensorLens, a novel high-order tensor formulation that represents the entire Transformer as a data-controlled linear operator, yielding generalized attention maps that can replace standard attention matrices and their cross-layer aggregations. (ii) We provide theoretical justification showing that this formulation is principled, more precise than prior attention variants, and encompasses all model parameters. (iii) We empirically show that TensorLens better reflects model behavior through perturbation-based evaluations. Finally, (iv) we demonstrate that TensorLens provides a robust foundation for mechanistic interpretability tools, such as approximating linear relations from LLM embeddings.

2 Background & Related Work

This section provides the scientific context for discussing our approach to precisely aggregating attention matrices via high-order tensors.

2.1 Extended Attention Matrices

Due to their importance, extended formulations of attention matrices have been widely explored over the years. In particular, Kobayashi et al. (2020) demonstrated that attention analysis can be refined by incorporating the output projection matrix when analyzing Transformer heads. Additionally, Kobayashi et al. (2021) proposed a further refinement by incorporating the residual connections and normalization layers into the attention formulation, resulting in more precise formulation. These approaches were further extended by Kobayashi et al. (2023) who also incorporated the FFN sublayer into the attention analysis. Moreover, Abnar and Zuidema (2020) introduced the attention rollout technique, which aggregates attention weights across multiple layers by multiplying the per-layer attention matrices. The rollout method was applied by Modarressi et al. (2022) to aggregate the extended attention matrices of Kobayashi et al. (2021) across layers. Finally, most similarly to our work, Elhage et al. (2021) analyze 2 layer attention-only transformers using 4th order tensors to describe the end-to-end function of the model. Our approach builds on these works by proposing a more precise formulation that explicitly captures all Transformer blocks and their sub-components.

2.2 Attention as High-Order Tensors

Several prior works have proposed architectures that extend the matrix-based self-attention mechanism to higher-order tensors (Omranpour et al., 2025; Ma et al., 2019; Gao et al., 2020; Zhang et al., 2025), primarily to enhance expressivity (Sanford

et al., 2023). However, these approaches often come at the cost of reduced efficiency, prompting efforts to improve their computational performance (Liang et al., 2024). While related, this line of work focuses on architectural modifications to the Transformer, rather than reinterpreting the vanilla self-attention mechanism through a tensor-based formulation, as done in this work.

2.3 Explainability Attribution Methods

Attribution methods aim to explain the decisions of neural networks (NNs) by quantifying the contribution of each neuron or input feature to the model's output (Das and Rad, 2020). These tools are primarily used for interpretability and are crucial for making NNs more trustworthy and understandable (Doshi-Velez and Kim, 2017). Attribution can be either class-specific, where the explanation targets a particular output class (for example, why the model predicted "cat" over "dog"), or class-agnostic, where the method provides a general explanation of the model's behavior regardless of any specific output (Hassija et al., 2024). While class-specific methods are valuable for understanding individual decisions, class-agnostic methods offer insights into the model's global processing, emergent patterns, and internal representations. Both perspectives are complementary and play a central role in building explainable AI systems.

Popular class-specific attribution methods include gradient-based techniques such as Input $\times$ Gradient (Shrikumar et al., 2017; Baehrens et al., 2010), and Layer-wise Relevance Propagation (LRP) (Bach et al., 2015; Achtibat et al., 2024; Bakish et al., 2025). In contrast, common class-agnostic methods include activation maximization (Erhan et al., 2009), probing techniques (Alain and Bengio, 2016), and the extraction of attention maps (Abnar and Zuidema, 2020). This paper focuses on developing a class-agnostic explainability method for Transformers, based on a more generalized and insightful formulation of attention matrices via Tensors.

3 Method: TensorLens

A standard Transformer architecture with N layers and hidden representations $X^n \in \mathbb{R}^{L\times D}$ for any $n \in [N]$ is defined as follows:

$$\forall n \in [N] : X^{n+1} = \text{Transformer}^n(X^n)\,, \quad (1)$$

where each Transformer block is defined by:

$$Z^n = \text{LayerNorm}^n(\text{Attention}^n(X^n) + X^n)\,, \quad (2)$$

$$X^{n+1} = \text{LayerNorm}^n(\text{FFN}^n(Z^n) + Z^n)\,. \quad (3)$$

Here, LayerNorm denotes the layer normalization operation (Ba et al., 2016), FFN is the feed-forward layer, and Attention is the self-attention layer. The superscript n indicates the signals, parameters or operations corresponding to the n-th layer, and each intermediate representation is a matrix in $\mathbb{R}^{L\times D}$, where L is the sequence length and D is the hidden dimension. We also assume that the input and output are multiplied by an embedding matrices E_{in} and E_{out}.

Intuition. Our key insight is that each subcomponent of the Transformer can be represented as a data-controlled linear operator defined by a data-dependent matrix. However, while some components, such as attention, mix interactions between tokens, others, like the FFN, mix across dimensions. As a result, their combination cannot be represented by a single matrix. Instead, it requires a tensor-based operator to capture both types of interactions.

To materialize our insight, we show that each sub-layer in Transformers can be represented as a tensor-based data-control linear operator in Section 3.2, followed by how these tensors can be aggregated to represent each block and the entire model in Sections 3.3, 3.4, and 3.5. A schematic visualization of the method is presented in Figure 2.

3.1 Prerequisites

Our formulation of the Transformer as tensors builds on the following rules for vectorizing matrix operations and tensor calculations (Itskov, 2007):

Bilinear Map. A bilinear map AXB can be vectorized using the Kronecker product $\otimes$ as:

$$\text{vec}\big[\underbrace{A}_{L\times L}\underbrace{X}_{L\times D}\underbrace{B}_{D\times D}\big] = \underbrace{\left(B^\top \otimes A\right)}_{LD\times LD}\underbrace{\text{vec}\,[X]}_{LD}\,.$$

Matrix Multiplication. A matrix multiplication XM can be vectorized as:

$$\text{vec}\big[\underbrace{I_L}_{L\times L}\underbrace{X}_{L\times D}\underbrace{M}_{D\times D}\big] = \underbrace{\left(M^\top \otimes I_L\right)}_{LD\times LD}\underbrace{\text{vec}\,[X]}_{LD}\,.$$

Element-wise Hadamard Product. An element-wise Hadamard product is vectorized as:

$$\text{vec}\big[\underbrace{H}_{L\times D} \odot \underbrace{X}_{L\times D}\big] = \underbrace{\text{diag}\,(\text{vec}\,[H])}_{LD\times LD}\underbrace{\text{vec}\,[X]}_{LD}\,.$$

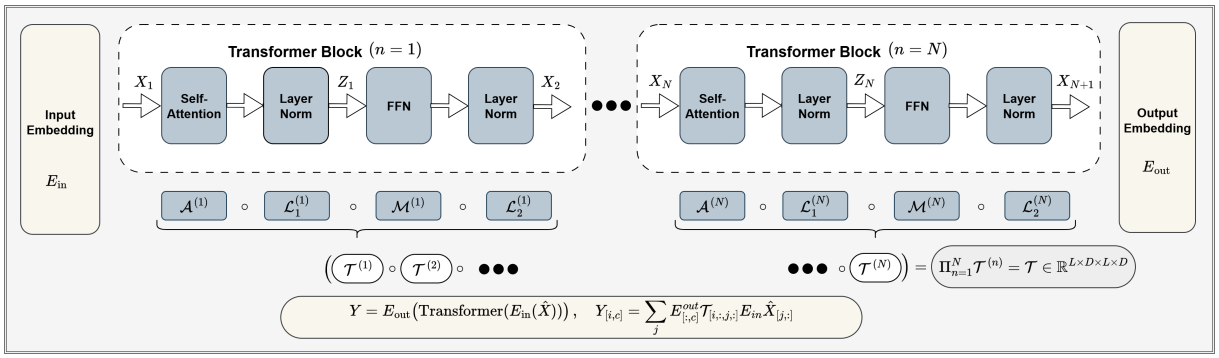


Figure 2: **Method:** A schematic visualization of our method, where each sub-component of the transformer architecture, including self-attention, LayerNorm, FFNs, input and output embedding layers, and the residual connection (which is omitted here for simplicity), is formulated as a data-controlled linear operator represented by high-order tensor in $\mathbb{R}^{L\times D\times L\times D}$. These tensors are composed into per-block tensors $\mathcal{T}^{(n)}$ for each layer $n \in [N]$, which are then used to construct the final linear operator representing the entire Transformer.

Tensor Contractions. For an input matrix $X \in \mathbb{R}^{L\times D}$ and a 4th order tensor $\mathcal{T} \in \mathbb{R}^{L\times D\times L\times D}$, we define the tensor contraction $\mathcal{T}(X)$ as:

$$\forall i \in [L]: \ \mathcal{T}(X)_{[i,:]} = \sum_{j=1}^{L} \underbrace{\mathcal{T}_{[i,:,j,:]}}_{D\times D} \underbrace{X_{[j,:]}}_{D} \in \mathbb{R}^D . \tag{4}$$

Unfolding the tensor into a matrix $\mathcal{T}_{\text{mat}} \in \mathbb{R}^{LD\times LD}$, the vectorized tensor contraction follows

$$\text{vec}\left[\mathcal{T}(X)\right] = \mathcal{T}_{\text{mat}} \text{vec}\left[X\right] . \tag{5}$$

In the following sections we overload notation, referring to $\mathcal{T}_{\text{mat}}$ as $\mathcal{T}$.

3.2 Block-by-Block Tensorization

We now show how each sub-layer in the Transformer architecture (LayerNorm, self-attention, FFN, residual) can be "Tensorized" into a linear tensor form. For simplicity, in this section we omit superscripts and weight biases, derivation including biases is in Appendix B.

Tensorized Self-Attention. Recall that given an input X, the multi-head self-attention layer with H heads is parameterized by key $W_{k,h}$, query $W_{q,h}$, value $W_{v,h}$, and output $W_{o,h}$ projections for each head $h \in [H]$, and is defined by:

$$\text{Attn}(X) = \sum_{h=1}^{H} A_h \, X \, W_{v,h} \, W_{o,h} , \tag{6}$$

$$A_h = \text{softmax}(Q_h K_h^\top) , \tag{7}$$

$$Q_h = XW_{q,h}, \quad K_h = XW_{k,h} . \tag{8}$$

Vectorising and grouping heads gives the following attention tensor $\mathcal{A}$:

$$\text{vec}[\text{Attn}(X)] = \underbrace{\sum_{h=1}^{H} \left((W_{v,h} W_{o,h})^\top \otimes A_h \right)}_{\mathcal{A}} \text{vec}[X] . \tag{9}$$

Tensorized LayerNorm. Recall that LayerNorm applies an affine transformation based on the input statistics and operates independently on each token:

$$\text{LN}(X) = \gamma \odot \tfrac{X-\mu}{\sigma} + \beta , \tag{10}$$

where $\gamma \in \mathbb{R}^D$ and β are learnable parameters, and μ and $\sigma \in \mathbb{R}^L$ are the per-token statistics , all broadcasted to match $X \in \mathbb{R}^{L\times D}$. With pre-computed variance σ^2, the LayerNorm can be tensorized by:

$$\begin{aligned} \text{vec}\big[\text{LN}(X)\big] &= \text{vec}\big[\text{diag}\big(\tfrac{1}{\sigma}\big) X (I_D - \tfrac{\mathbf{1}\mathbf{1}^\top}{D}) \text{diag}(\gamma)\big] \\ &= \underbrace{\big[(I_D - \tfrac{\mathbf{1}\mathbf{1}^\top}{D})\text{diag}(\gamma)\big]^\top \otimes \text{diag}\big(\tfrac{1}{\sigma}\big)}_{\mathcal{L}} \text{vec}[X] , \end{aligned} \tag{11}$$

where $(I_D - \frac{\mathbf{1}\mathbf{1}^\top}{D}) \in \mathbb{R}^{D\times D}$ is the mean centering function in matrix form, with $\mathbf{1} \in \mathbb{R}^D$ a column vector of all ones.

Tensorized FFN. Given an activation function ϕ, the FFN is defined by two linear layers as follows:

$$\text{FFN}(X) = \phi(XM_1)M_2. \tag{12}$$

The element-wise activation can be converted to an input-dependent hadamard product $\frac{\phi(Z)}{Z} \odot Z$,

and tensorized as:

$$\mathrm{vec}\big[\phi(Z)\big] = \underbrace{\mathrm{diag}\left(\mathrm{vec}\left[\frac{\phi(Z)}{Z}\right]\right)}_{\Psi}\mathrm{vec}[Z]\,. \quad (13)$$

Resulting in the full vectorized form of the FFN as follows:

$$\mathrm{vec}\big[\mathrm{FFN}(X)\big] = \underbrace{\big(M_2^\top \otimes I_L\big)\Psi\big(M_1^\top \otimes I_L\big)}_{\mathcal{M}}\mathrm{vec}[X]\,, \quad (14)$$

which is characterized by a tensor $\mathcal{M}$.

Tensorized Residual. For some sub-layer g, the residual connection can be written as:

$$\mathbf{Y}_{\text{res}} = X + g(X)\,. \quad (15)$$

Vectorizing this equation yields:

$$\mathrm{vec}\big[\mathbf{Y}_{\text{res}}\big] = \big(I + \mathcal{G}\big)\mathrm{vec}\big[X\big]\,, \quad (16)$$

where $\mathcal{G}$ is the tensor associated with g (e.g. $\mathcal{A}$ or $\mathcal{M}$ for attention and FFN accordingly) and $\mathcal{I} \in \mathbb{R}^{LD\times LD}$ an identity matrix.

3.3 Transformer Block as Tensor

A Transformer block is defined in Eq.1 and is obtained by stacking the sub-layers according to Eq.2 and Eq. 3. Thus, stacking the tensors obtained from the self-attention, residual, normalizations and FFNs as in Eqs. (9), (11), (14) (16), produces the tensor $\mathcal{T}^n$ associated with the n-th block:

$$\mathcal{T}^n = \mathcal{L}_2^n\left(\mathcal{M}^n + \mathcal{I}\right)\mathcal{L}_1^n\left(\mathcal{A}^n + \mathcal{I}\right)\,, \quad (17)$$

for a post-layernorm block. For a similar derivation for a pre-layernorm block, see Appendix B.

3.4 Entire Transformer as Tensor

Given the tensor formulation of a single Transformer block in Section 3.3 (Eq. 17), we now construct the full model as a composition of such block tensors. Let $\mathcal{T}^n$ denote the tensor representation of the n-th block, the entire Transformer function $\mathcal{F}$ can be expressed as a nested application of block tensors over the input sequence denoted as $X^0 = X$, yielding the following recursive structure:

$$\mathcal{F}(X) = \mathcal{T}^{(N)} \circ \mathcal{T}^{(N-1)} \circ \cdots \circ \mathcal{T}^{(1)}(X)\,. \quad (18)$$

Thus, the entire model is fully represented as a chain of high-order tensor transformations:

$$\mathrm{vec}[\mathcal{F}(X)] = \mathrm{vec}[\mathcal{T}(X)] = \left(\prod_{n=1}^{N}\mathcal{T}^n\right)\mathrm{vec}\big[X\big]\,, \quad (19)$$

which completes the transition from individual layer operations to a unified tensor-based view of the full Transformer expressed by $\mathcal{T}(X)$.

Interpretation as Generalized Attention. We denote by $\mathcal{T}$ the linearized representation of the entire Transformer, expressed as a 4th-order tensor of dimensions $L \times D \times L \times D$, where L is the sequence length and D is the hidden dimension. This tensor captures the influence of each input token–channel pair on every output token–channel pair. Conceptually, $\mathcal{T}$ can be interpreted as a generalization of the conventional attention matrix to a *higher-order attention tensor*, modeling both inter-token dependencies and intra-token (cross-channel) interactions. In the unvectorized form, each position $i \in [L]$ in the output is obtained by a sum of linear transformations of the input, which are defined by slices of the overall tensor:

$$\forall i \in [L] : \mathcal{F}(X)_{[i,:]} = \sum_j \underbrace{\mathcal{T}_{[i,:,j,:]}}_{D\times D}\underbrace{X^\top_{[j,:]}}_{D}\,. \quad (20)$$

Our goal with generalized attention is not to propose a new architecture, but to formulate attention in a way that yields a representation encapsulating more components and computations, following the line of work described in Section 2.1. Importantly, our method is not limited to end-to-end linearization alone. By restricting the composition to a chosen subset of layers or heads, we can obtain generalized attention matrices at any desired granularity.

3.5 From Tensor to Matrix by Collapsing

While the tensor representation provides a richer and more comprehensive view of Transformer computations, it is often less interpretable and more difficult to visualize due to its 4th-order structure and the sheer number of elements (L^2D^2 in total). To address this, we propose a simple yet effective technique for collapsing the tensor into a more compact, matrix-like form, akin to the standard attention matrix. Specifically, we reduce the $D \times D$ channel dimensions using the following three approaches: **(i) Norm over feature dimensions:** By taking the

norm of the dimension related to the channel as follows:

$$\forall i,j \in [L] : T^{\text{Norm}}_{i \leftarrow j} = \left\| \mathcal{T}_{[i,:,j,:]} \right\|_2 , \quad (21)$$

resulting in a matrix $T^{\text{Norm}} \in \mathbb{R}^{L \times L}$. **(ii) Projection using output and input embedding vectors:** Let $X^0, X^N \in \mathbb{R}^{L \times D}$ be the hidden states inserted and extracted from the Transformer layers. We contract over the channel dimensions using X^0, X^N as both input and output projection weights:

$$\forall i,j \in [L] : T^{\text{IO}}_{i \leftarrow j} = X^N_{[i,:]} \mathcal{T}_{[i,:,j,:]} X^0_{[j,:]} . \quad (22)$$

Following Eq. 20, taking an inner product with $X^N_{[i,:]}$ on both sides yields:

$$X^{N\top}_{[i,:]} X^N_{[i,:]} = X^N_{[i,:]} \sum_j \mathcal{T}_{[i,:,j,:]} X^0_{[j,:]} , \quad (23)$$

$$\left\| X^N_{[i,:]} \right\|^2 = \sum_j T^{\text{IO}}_{i,j} ,$$

meaning $T^{\text{IO}}_{i,j}$ reflects the contribution of the input $X^0_{[j,:]}$ to the output $X^N_{[i,:]}$.

This approach can be applied to the entire Transformer or to a selected subset of blocks. **(iii) Class-specific projection using output embedding matrices:** For a chosen output class/token c, let $E^{out}_{[:,c]} \in \mathbb{R}^D$ be the corresponding column in the classification head/unembedding matrix, we contract the tensor over the channel dimensions using X^0, $E^{out}_{[:,c]}$ to get:

$$\forall i,j \in [L] : T^{\text{CLS}}_{(c,i) \leftarrow j} = E^{out}_{[:,c]} \mathcal{T}_{[i,:,j,:]} X^0_{[j,:]} . \quad (24)$$

Similarly to Eq. 23, for each class c and output position i, the input contributions $T^{\text{CLS}}_{(c,i) \leftarrow j}$ sum up to the logit of the class c (excluding biases, see Appendix B).

Eq. 24 provides a comprehensive view of the Transformer as a linear operator. It encapsulates all model parameters, including the Transformer and embedding layers, and serves as a direct local approximation of the full Transformer computation. As the first formulation that explicitly captures all model parameters within a unified tensor-based representation, it offers a principled foundation for analyzing and interpreting Transformer computations through the lens of high-order linear operators.

3.6 Theoretical Analysis

Our method relies on a linearization of the *entire* Transformer computation at a given input. A natural question arises: how well does this linearization approximate the original function locally? We address this in Proposition 1, where we provide a data and model-dependent bound on the forward approximation error. It is important to note that alternative methods, which either do not incorporate all model parameters in their formulation or avoid tensor-level operations, are generally not capable of producing such bounds. Even when they can be applied, the resulting approximations are typically significantly looser.

Proposition 1. *The **approximation error** of the tensor $\mathcal{T}_X$ computed on input X, when evaluating the transformer function $\mathcal{F}$ at $(X + \epsilon)$ is bounded by:*

$$\left\| \mathcal{T}_X (X + \epsilon) - \mathcal{F}(X + \epsilon) \right\|_2 \leq \quad (25)$$
$$\left\| \mathcal{T}_X \right\|_2 \left\| \epsilon \right\|_2 + \left\| \mathcal{F}(X + \epsilon) - \mathcal{F}(X) \right\|_2 ,$$

where $\left\| \mathcal{T}_X \right\|_2$ is bounded by constants of the transformer weights.

The complete proof is provided in Appendix E. The core intuition is that each sub-component of the Transformer can be linearly approximated using tensor operations, enabling the error to be bounded by recursively applying standard first-order linear approximation techniques, which can be composed across layers to yield a global bound.

4 Experiments

We empirically assess the representation power of our tensor formulation as a proxy for Transformer behavior, comparing it to other attention aggregation techniques via perturbation tests in Section 4.1. Then, in Section 4.2, we demonstrate that the tensor representation is a valuable tool for mechanistic interpretability and model understanding.

4.1 Perturbation Tests

To assess the representational power of our attention aggregation method, we adopted an input perturbation scheme similar to Chefer et al. (2021); Ali et al. (2022). This evaluation strategy gradually masks input tokens in the order determined by their computed relevance scores. When the highest-scoring tokens are masked first (positive perturbation), we expect the model's accuracy to rapidly decline. We assess explanation quality using the Area Under the Curve (AUC) metric, which

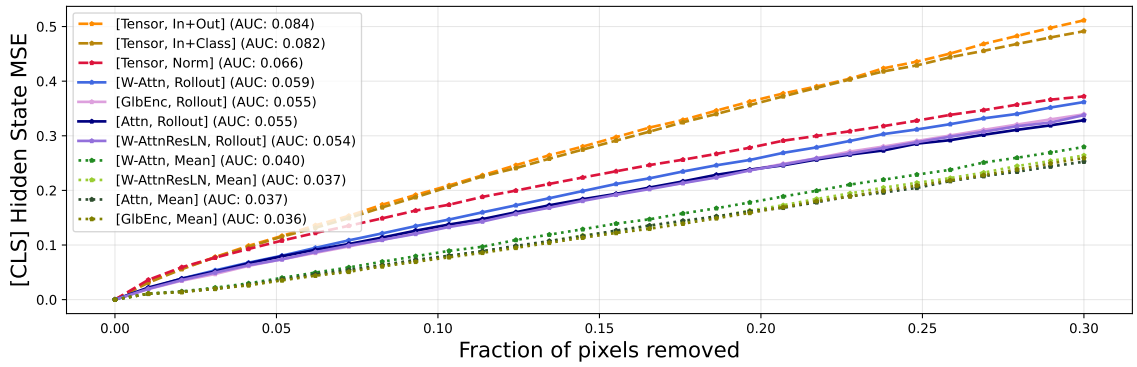


Figure 3: **Perturbation Tests in Vision:** Effect of perturbations on final hidden representations of DeiT-Base. Measured by the mean squared error between the last hidden-state of the [CLS] token in the original and perturbed input (higher is better).

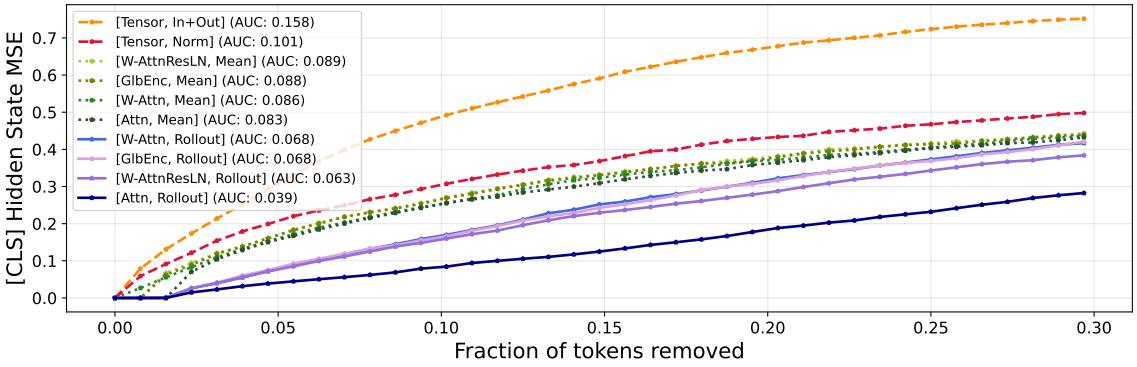


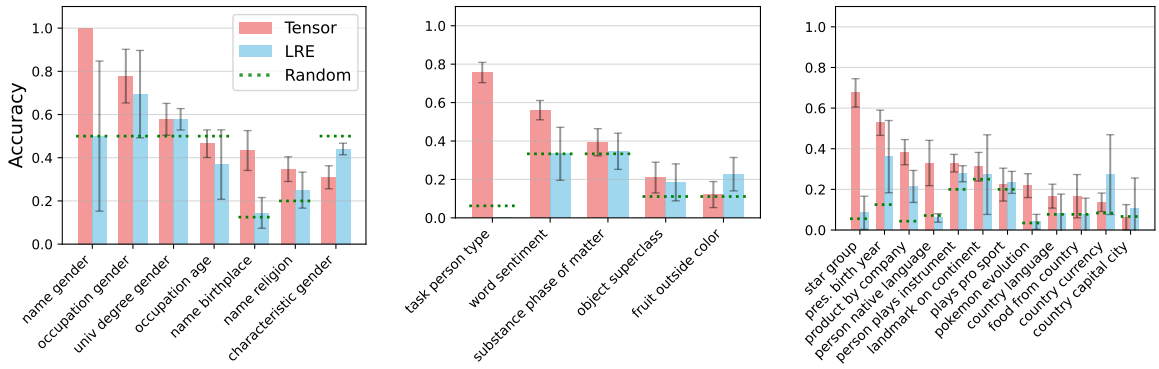
Figure 4: **Perturbation Tests in NLP:** Effect of token perturbations on final hidden representations of BERT-Base.

captures the model's accuracy as a function of the percentage of masked input elements, ranging from 0% to 30%.

As baselines, we use eight aggregation variants that combine two methods for cross-layer aggregation and four methods for intra-layer aggregation. For cross-layer aggregation, we apply either multiplicative composition, as in Attention Rollout (Abnar and Zuidema, 2020) ("Rollout"), or simple averaging ("Mean"). For intra-layer aggregation, we use the following four methods: (i) averaging of attention matrices ("Attn"), (ii) value-weighted attention as proposed by Kobayashi et al. (2020) ("W. Attn"), (iii) value-weighted attention that also includes the residual connection and LayerNorm as proposed by Kobayashi et al. (2021) ("W. AttnResLN"), and (iv) a global encoding variant ("GlbEnc") that further incorporates the second LayerNorm into the formulation (Modarressi et al., 2022). We compare these class-agnostic baselines with the variants defined in Eqs. 21 and 22, denoted as "Tensor,Norm" and "Tensor,In+Out", respectively.

Perturbation in Vision. In the vision domain, we evaluate our methods using DeiT by Touvron et al. (2021), on the ImageNet-1K test set, considering both the base and small model sizes. Results for the base model are shown in Fig. 3. As can be seen, across all perturbation levels, tensor-based aggregation methods consistently outperform the baselines. When incorporating both input and output embeddings ('Tensor,In+Out'), the total AUC exceeds 0.82, and reaches 0.66 when using the tensor norm ('Tensor,Norm'). In contrast, all aggregation methods that are not based on tensors fall below 0.6, highlighting the superior robustness of our formulation. The results for DeIT-small, which follow a similar trend are provided in Appendix A.

Perturbations in NLP. In the NLP domain, we evaluate our method across several models on sequences of length 128, including both encoder-only and decoder-only architectures. For the encoder-only setting, we conduct experiments with BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) on the IMDB dataset. Results for



(a) Bias (b) Common Sense (c) Factual

Figure 5: **Relation Decoding:** Accuracy relative to original model computation, for different relation categories on Pythia-1B, with $m = 3$ training samples per relation. Results are averaged across 6 train-test splits, with standard deviation shown in error bars. Random baselines shown as horizontal dashed lines.

BERT are shown in Figure 4, where tensor-based aggregation consistently outperforms all baselines across all perturbation levels. When incorporating both input and output embeddings ('Tensor,In+Out'), the total AUC exceeds 0.158, and reaches 0.101 when using only the tensor norm ('Tensor,Norm'). In contrast, all non-tensor aggregation methods fall below 0.09, underscoring the superior robustness of our formulation. Moreover, in Appendix A, we also report results for RoBERTa in Table 1, as well as for the more recent ModernBert (Warner et al., 2024) and Gemma3 (Team et al., 2025) in Table 2. In these figures, the observed pattern closely aligns with that of BERT.

When evaluating decoder-only models, the results are less clear-cut. In this setting, we tested several LLMs, including Pythia-1B (Biderman et al., 2023), Pico-570M (Martinez et al., 2024), and Phi-1.5 (Li et al., 2023), using the WikiText-103 dataset. As shown in Table 1 (Appendix A), although our method consistently achieved the top or second-best AUC scores across benchmarks, the overall findings are less conclusive. One possible explanation is that auto-regressive language models are trained to predict the next token and therefore tend to exhibit inherently local behavior (Fang et al., 2024). This characteristic may reduce the informativeness of perturbation-based evaluations, making the results appear less definitive.

4.2 Approximation of Relation Decoding

The tensor formulation $\mathcal{T}$, which we uncover from the forward pass of the model, mathematically describes a linear transformation between tokens in the same sentence. In this section we evaluate the quality of our method as a local approximation for the Transformer computation through the lens of *linear relation decoding*. Introduced by Hernandez et al. (2024), linear relation decoding examines sets of relations, such as *"A teacher typically works at a school"*, composed of triplets (s, r, o) connecting a subject s to an object o via relation r. Hernandez et al. (2024) illustrate how to produce transformation between tokens embeddings, such as ones that output *"school"* for *"teacher"* or *"hospital"* for *"doctor"*. Their method was based on approximating the Jacobian matrix of the model's prediction relative to the subject token "s". Since our tensor formulation is a multi-linear transformation that describes such input-output relations, our goal is to examine to what extent it can match the linear representation's performances of Hernandez et al. (2024) which were tailored for this task.

In order to create a per-relation transformation, we compute the mean tensor extracted from m examples $X_i = (s_i, r, o_i)$ of a relation r:

$$\widetilde{\mathcal{T}}_r = \frac{1}{m} \sum_{i=1}^{m} \mathcal{T}_{X_i}\,, \tag{26}$$

and measure the similarity of the tensor function $\widetilde{\mathcal{T}}_r(X)$ to that of the original model, on a held-out test set of subject-object pairs of the same relation.

For experimental setup, we follow Hernandez et al. (2024) and prepend each example X_i in the mean calculation with the remaining $m - 1$ train examples as few-shot examples, so that the model is more likely to generate the answer o given a s under the relation r over other plausible tokens. Further experimental details are described in Appendix D. We report the approximation accuracy

as the percentage of examples in which the top-predicted object o matches the original output.

As seen in Figure 5, approximating the model's computation using our tensor method achieves higher accuracy than the LRE baseline of Hernandez et al. (2024) on most relations examined. In some tasks, such as *occupation-age*, we found both methods to achieve results close to that of a random guess, which we associate with the inherent limitation of describing the model's internal processes solely via linear transformations of the input.

Overall, it is evident that our multi-linear approximation provides better capacity than previous linear methods to describe the function of the *entire* model as a whole. We find these results to strengthen our claim that the tensor formulation reflects the model's internal representations.

5 Conclusions

This work presents a technique for aggregating attention matrices across both Transformer blocks and all sub-components within each block. The resulting formulation is theoretically grounded and more comprehensive than prior approaches and it is based on representing the Transformer as a high-order data-controlled linear operator. This formulation captures the internal interactions of the model, including contributions from components such as the FFN, embeddings, LayerNorm, and others. Practically, we emphasize that this formulation can be used as a drop-in replacement for attention matrices and their aggregations, in order to enhance many existing interpretability, analysis, and intervention techniques. An example of direct application to mechanistic interpretability and model understanding is demonstrated in our relation-based analysis.

Limitations

While TensorLens offers a more precise formulation of the Transformer through a self-attention-based representation compared to prior work, it has several limitations. First, some of the linearization techniques are chosen for their simplicity rather than being derived from the intrinsic properties of optimally approximated tensors. For example, this includes the activation decomposition in Eq. 13. Second, the high-order tensor representation is GPU-memory intensive. We partially mitigate this with a memory-optimized computation method as described in Appendix C, however, our experiments are limited to models up to 1B parameters and moderate input lengths. Third, although the formulation is comprehensive and practical for visualization and model interpretation, the full potential of the tensor-based approach remains underexplored. In particular, it opens the door to new perspectives on rank collapse, sparsity, and training dynamics through the lens of tensor properties.

Ethics Statement

This work focuses on developing a theoretically grounded and interpretable representation of Transformer-based models via high-order attention tensors. Our research does not involve human subjects, personally identifiable data, or the generation of potentially harmful content. All evaluations are conducted on publicly available datasets such as ImageNet, IMDB, and WikiText-103, adhering to their respective licenses and intended usage.

We acknowledge that improved model interpretability tools, such as those proposed in this work, may be used both to enhance trust in machine learning systems and to expose or exploit model vulnerabilities. We believe that the positive implications including greater transparency, accountability, and error analysis, outweigh potential misuse. Nonetheless, we encourage responsible use of our methods in alignment with ethical AI principles.

Acknowledgments

This work was supported by a Tel Aviv University Center for AI and Data Science (TAD) grant and by the Blavatnik Family foundation. This research was also supported by the Ministry of Innovation, Science & Technology, Israel (1001576154) and the Michael J. Fox Foundation (MJFF-022407).

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*.

Reduan Achtibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. 2024. Attnlrp: attention-aware layer-wise relevance propagation for transformers. *arXiv preprint arXiv:2402.05602*.

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

Ameen Ali, Thomas Schnake, Oliver Eberle, Grégoire Montavon, Klaus-Robert Müller, and Lior Wolf. 2022. Xai for transformers: Better explanations through conservative propagation. In *International conference on machine learning*, pages 435–451. PMLR.

Ameen Ali Ali, Shahar Katz, Lior Wolf, and Ivan Titov. 2025a. Detecting and pruning prominent but detrimental neurons in large language models. In *Second Conference on Language Modeling*.

Ameen Ali Ali, Itamar Zimerman, and Lior Wolf. 2025b. The hidden attention of mamba models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1516–1534.

Giuseppe Attanasio, Debora Nozza, Dirk Hovy, and Elena Baralis. 2022. Entropy-based attention regularization frees unintended bias mitigation from lists. *arXiv preprint arXiv:2203.09192*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *stat*, 1050:21.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

D Baehrens, T Schroeter, S Harmeling, M Kawanabe, K Hansen, and K-R Müller. 2010. How to explain individual classification decisions. *Journal of Machine Learning Research*.

Yarden Bakish, Itamar Zimerman, Hila Chefer, and Lior Wolf. 2025. Revisiting lrp: Positional attribution as the missing ingredient for transformer explainability. *arXiv preprint arXiv:2506.02138*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. 2023. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *ACM transactions on Graphics (TOG)*, 42(4):1–10.

Hila Chefer, Shir Gur, and Lior Wolf. 2021. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 782–791.

Tri Dao and Albert Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*.

Arun Das and Paul Rad. 2020. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and 1 others. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, and 1 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2009. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. 2024. What is wrong with perplexity for long-context language modeling? *arXiv preprint arXiv:2410.23771*.

Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2020. Kronecker attention networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 229–237.

Vikas Hassija, Vinay Chamola, Atmesh Mahapatra, Abhinandan Singal, Divyansh Goel, Kaizhu Huang, Simone Scardapane, Indro Spinelli, Mufti Mahmud, and Amir Hussain. 2024. Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, 16(1):45–74.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2024. Linearity of relation decoding in transformer language models. In *The Twelfth International Conference on Learning Representations*.

Mikhail Itskov. 2007. *Tensor algebra and tensor analysis for engineers: with applications to continuum mechanics*. Springer.

Shahar Katz, Yonatan Belinkov, Mor Geva, and Lior Wolf. 2024. Backward lens: Projecting language model gradients into the vocabulary space. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2390–2422.

Shahar Katz and Lior Wolf. 2025. Reversed attention: On the gradient descent of attention layers in GPT. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1125–1152, Albuquerque, New Mexico. Association for Computational Linguistics.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. *arXiv preprint arXiv:2004.10102*.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2021. Incorporating residual and normalization layers into analysis of masked language models. *arXiv preprint arXiv:2109.07152*.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2023. Analyzing feed-forward blocks in transformers through the lens of attention maps. *arXiv preprint arXiv:2302.00456*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R Lyu, and Tong Zhang. 2018. Multi-head attention with disagreement regularization. *arXiv preprint arXiv:1810.10183*.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: **phi-1.5** technical report. *arXiv preprint arXiv:2309.05463*.

Yingyu Liang, Zhenmei Shi, Zhao Song, and Yufa Zhou. 2024. Tensor attention training: Provably efficient learning of higher-order transformers. *arXiv preprint arXiv:2405.16411*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. 2019. A tensorized transformer for language modeling. *Advances in neural information processing systems*, 32.

Richard Diehl Martinez, Pietro Lesci, and Paula Buttery. 2024. Tending towards stability: Convergence challenges in small language models. *arXiv preprint arXiv:2410.11451*.

Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. 2020. Dissecting neural odes. *Advances in neural information processing systems*, 33:3952–3963.

Ali Modarressi, Mohsen Fayyaz, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2022. GlobEnc: Quantifying global token attribution by incorporating the whole encoder layer in transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 258–271, Seattle, United States. Association for Computational Linguistics.

Soroush Omranpour, Guillaume Rabusseau, and Reihaneh Rabbany. 2025. Higher order transformers: Efficient attention mechanism for tensor structured data.

William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR.

Clayton Sanford, Daniel J Hsu, and Matus Telgarsky. 2023. Representational strengths and limitations of transformers. *Advances in Neural Information Processing Systems*, 36:36677–36707.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *Preprint*, arXiv:2412.13663.

Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. 2024. Lolcats: On low-rank linearizing of large language models. *arXiv preprint arXiv:2410.10254*.

Yifan Zhang, Yifeng Liu, Huizhuo Yuan, Zhen Qin, Yang Yuan, Quanquan Gu, and Andrew C Yao. 2025. Tensor product attention is all you need. *arXiv preprint arXiv:2501.06425*.

Itamar Zimerman, Ameen Ali, and Lior Wolf. 2025. Explaining modern gated-linear rnns via a unified implicit attention formulation. In *ICLR*.

Itamar Zimerman and Lior Wolf. 2024. Viewing transformers through the lens of long convolutions layers. In *Forty-first International Conference on Machine Learning*.

A Additional Perturbation Experiments

In addition to the perturbation tests presented in Section 4.1, this section presents further experiments with RoBERTa, DeiT-small, and the more recent ModernBert and Gemma3. The results are shown in Figures 6 and 7, respectively. As illustrated, across all benchmarks, our method achieves higher AUC scores, consistently outperforming the baselines for all perturbation fractions. Furthermore, in Table 1, we present perturbation results for decoder-only models, while Table 2 reports results for more modern models, including ModernBert (Warner et al., 2024) and Gemma3 (Team et al., 2025).

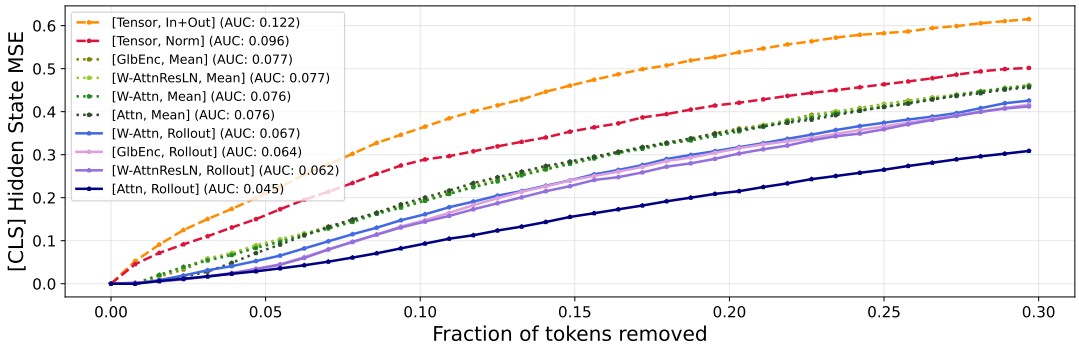


Figure 6: **Perturbation Tests in NLP:** Effect of token perturbations on final hidden representations of RoBERTa-Base. Measured by the mean squared error between the last hidden-state of the [CLS] token in the original and perturbed input (higher is better)

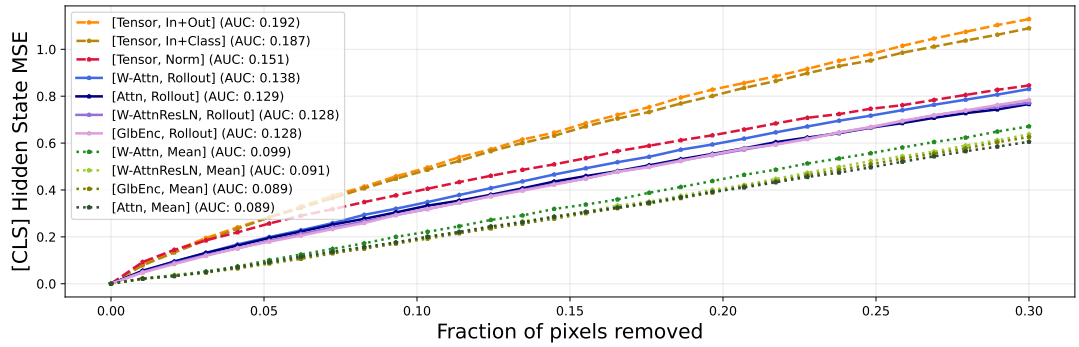


Figure 7: **Perturbation Tests in Vision:** Effect of perturbations on final hidden representations of DeiT-Small.

	Method:	Tensor (ours)			Rollout over layers				Mean over heads & layers			
LLM	Metric	Norm	In+Out	In+Class	Attn	W-Attn	W-AttnResLN	GlbEnc	Attn	W-Attn	W-AttnResLN	GlbEnc
Pyth	HS-MSE ↑	**3.708**	3.603	3.683	0.306	0.213	0.233	N.A	3.483	3.650	**3.708**	N.A
	AOPC ↑	0.147	0.146	0.147	0.044	0.035	0.037	N.A	0.141	0.147	**0.148**	N.A
Pico	HS-MSE ↑	0.222	0.223	0.22	0.03	0.014	0.090	0.036	0.211	0.22	0.222	**0.225**
	AOPC ↑	**0.129**	0.128	**0.129**	0.019	0.020	0.067	0.036	0.125	0.127	**0.129**	**0.129**
Phi	HS-MSE ↑	0.892	0.887	0.886	0.079	0.067	0.053	N.A	0.864	0.905	**0.934**	N.A
	AOPC ↑	0.141	0.143	0.143	0.028	0.023	0.026	N.A	0.133	0.141	**0.144**	N.A

Table 1: **Next Token Prediction Perturbation.** Results are AUC (higher is better) of (i) HS-MSE: Mean squared error between the last hidden-state of the final token in the original and perturbed input. (ii) AOPC: Absolute difference of the soft-maxed probability of the original predicted token, between the original and perturbed input. The GlbEnc results are not presented for the Pythia and Phi models, since their method is inapplicable for parallel-residual architectures. 'Pyth' for Pythia.

	Method:	Tensor (ours)		Rollout over layers				Mean over heads & layers			
LLM	Metric	Norm	In+Out	Attn	W-Attn	W-AttnResLN	GlbEnc	Attn	W-Attn	W-AttnResLN	GlbEnc
ModernBert	HS-MSE ↑	0.081	**0.112**	0.05	0.066	0.064	0.063	0.069	0.072	0.077	0.075
Gemma3	HS-MSE ↑	0.029	**0.049**	0.014	0.014	0.019	0.017	0.023	0.024	0.02	0.021

Table 2: **Perturbation Tests in NLP with Modern Models:** Effect of token perturbations on final hidden representations of ModernBert-Base and Gemma3-270M, trained for sentiment prediction on the IMDB dataset. Results are AUC of HS-MSE: Mean squared error between the last hidden-state of the final token in the original and perturbed input (higher is better).

B Tensor Derivation with Biases

Here we reiterate the tensor derivation introduced in Section 3 while including the transformer weight biases. The perturbation experiments in Section 4.1 use the tensor without biases as described in Section 3, and the relation decoding experiments in Section 4.2 use the full affine transformation described here.

We denote the model biases as $B \in \mathbb{R}^{L\times D}$, broadcasting the original $b \in \mathbb{R}^{D}$ biases to each sequence position L. For each module f in the transformer block, we get an affine transformation of the form:

$$\mathrm{vec}\big[f(X)\big] = \mathcal{T}^{(f)}\mathrm{vec}[X] + \mathrm{vec}[B^{(f)}]$$

Tensorized Self-Attention. For an input $X \in \mathbb{R}^{L\times D}$, multi-head self attention is defined by:

$$\mathrm{Attn}(X) = \sum_{h=1}^{H} A_h \ (X\,W_{v,h} + B_{v,h})\ W_{o,h} + B_{o,h}$$

$$= \sum_{h=1}^{H} A_h \ (X\,W_{v,h})\ W_{o,h} + B_{\text{attn}}\,,$$

where $B_{\text{attn}} = B_{v,h}W_{o,h} + B_{o,h}$. The biases of the query and key projections are absorbed in the attention matrix A_h. Vectorising and grouping heads gives the attention tensor $\mathcal{A}$:

$$\mathrm{vec}[\mathrm{Attn}(X)] = \underbrace{\sum_{h=1}^{H}\Big((W_{v,h}W_{o,h})^{\top} \otimes A_h\Big)}_{\mathcal{A}}\,\mathrm{vec}[X] + \mathrm{vec}[B_{\text{attn}}] \in \mathbb{R}^{LD}\,.$$

where $\mathcal{A} \in \mathbb{R}^{LD\times LD}$ is flattened to a matrix as defined in Eq. (5).

Tensorized LayerNorm. With weights $\gamma \in \mathbb{R}^{D\times D}$ and bias $\beta \in \mathbb{R}^{L\times D}$, the LayerNorm

$$\mathrm{LayerNorm}(X) = \gamma \odot \tfrac{X-\mu}{\sqrt{\sigma^2+\varepsilon}} + \beta\,,$$

is similarly vectorized as:

$$\mathrm{vec}\big[\mathrm{LayerNorm}(X)\big] = \underbrace{\big[(I_D - \tfrac{\mathbf{1}\mathbf{1}^{\top}}{D})\mathrm{diag}(\gamma)\big]^{\top} \otimes \mathrm{diag}\big(\tfrac{1}{\sqrt{\sigma^2+\varepsilon}}\big)}_{\mathcal{L}}\,\mathrm{vec}[X] + \mathrm{vec}[\beta] \in \mathbb{R}^{LD}\,.$$

Tensorized FFN Given an activation function ϕ, the FFN is defined by two linear layers as follows:

$$\mathrm{FFN}(X) = \phi(XM_1 + B_{M_1})M_2 + B_{M_2}.$$

The element-wise activation can be converted to an input-dependent hadamard product $\frac{\phi(Z)}{Z} \odot Z$, and tensorized as:

$$\mathrm{vec}\big[\phi(Z)\big] = \underbrace{\mathrm{diag}\left(\mathrm{vec}\left[\frac{\phi(Z)}{Z}\right]\right)}_{\Psi}\,\mathrm{vec}[Z]\,.$$

Resulting in the full vectorized form of the FFN as follows:

$$\mathrm{vec}\big[\mathrm{FFN}(X)\big] = \underbrace{\big(M_2^\top \otimes I_L\big)\Psi\big(M_1^\top \otimes I_L\big)}_{\mathcal{M}}\,\mathrm{vec}[X] \;+ \underbrace{\mathrm{vec}[B_{M_2}] + \big(\mathrm{M}_2^\top \otimes \mathrm{I}_\mathrm{L}\big)\Psi\mathrm{vec}[B_{M_1}]}_{\mathrm{vec}[B_{\mathrm{FFN}}]} \in \mathbb{R}^{LD}\,.$$

which is characterized by a tensor $\mathcal{M}$.

Transformer Block as Tensor. As defined in Eq. (17), stacking the tensors obtained from the self-attention, residual, normalizations, produces the tensor $\mathcal{T}^n$ associated with the n-th post-layernorm block:

$$\mathcal{T}^n = \mathcal{L}_2^n\left(\mathcal{M}^n + \mathcal{I}\right)\mathcal{L}_1^n\left(\mathcal{A}^n + \mathcal{I}\right) + \mathrm{vec}[B_{\mathrm{block}}^n]$$

Where the bias of each sub-module is transformed by the following ones as:

$$\mathrm{vec}[B_{\mathrm{block}}^n] = \mathrm{vec}[\beta_2^n] + \mathcal{L}_2^n\left(\mathrm{vec}[B_{\mathrm{FFN}}^n] + \mathcal{M}^n\left(\mathrm{vec}[\beta_1^n] + \mathcal{L}_1^n\mathrm{vec}[B_{\mathrm{attn}}^n]\right)\right)$$

The derivation for a pre-layernorm block is obtained similarly, by changing the order of the components:

$$\mathcal{T}^n = \left(\mathcal{I} + \mathcal{M}^n\mathcal{L}_2^n\right)\left(\mathcal{I} + \mathcal{A}^n\mathcal{L}_1^n\right) + \mathrm{vec}[B_{\mathrm{block}}^n]$$

Entire Transformer as Tensor. As shown in Eq. (19), the entire model $\mathcal{F}$ is fully represented as a chain of high-order tensor transformations. Adding biases results in the final affine transformation:

$$\mathrm{vec}[\mathcal{F}(X)] = \left(\prod_{n=1}^{N}\mathcal{T}^n\right)\mathrm{vec}\big[X\big] + \mathrm{vec}[B_{\mathrm{full}}]\,,$$

where the bias of each block is recursively transformed by the following ones:

$$\mathrm{vec}[B_{\mathrm{full}}] = \mathrm{vec}[B_{\mathrm{block}}^N] + \mathcal{T}^N\left(\mathrm{vec}[B_{\mathrm{block}}^{N-1}] + \cdots\right)\,.$$

C Memory-Efficient Tensor Computation

The tensor computation introduced in Section 3 relies on multiplications of large matrices in $\mathbb{R}^{LD\times LD}$, which may be prohibitive for larger models and longer input sequences. In practice, we use a memory-efficient computation method based on the following observations. **(i)** Given an input X, patching the original Transformer function $\mathcal{F}$ to use the precomputed attention matrices, FFN activations, and LayerNorm variance from the forward pass on X yields a linear function $\widetilde{\mathcal{F}}_X$ whose Jacobian is exactly the desired tensor:

$$\mathcal{T}_X = \frac{\partial\widetilde{\mathcal{F}}_X(X)}{\partial X} \in \mathbb{R}^{L\times D\times L\times D}. \tag{27}$$

(ii) The tensor $\mathcal{T}_X \in \mathbb{R}^{L\times D\times L\times D}$ captures the influence of each input token-channel pair on every output token-channel pair. Since in both the input attribution and relation decoding experiments we are only interested in the influence on a single output position $\ell_{\mathrm{out}} \in [L]$ (either the last token or the `[CLS]` token), it suffices to compute only the 3-dimensional tensor slice corresponding to that position, i.e., $\mathcal{T}_{X[\ell_{\mathrm{out}},:,:,:]} \in \mathbb{R}^{D\times L\times D}$.

Thus, in our experiments we compute only this 3-d slice using the Jacobian of the patched Transformer function, as in Eq. (27).

If needed, the full 4-d Jacobian and tensor can be computed in a memory-efficient manner using forward-mode differentiation, such that the full tensor is never materialized on the GPU. This is done by applying the patched Transformer function $\widetilde{\mathcal{F}}_X$ to unit (basis) matrices $E^{\ell,d} \in \mathbb{R}^{L\times D}$,

$$\left(E^{\ell,d}\right)_{i,j} = \begin{cases}1, & i = \ell \wedge j = d\\ 0, & \text{else,}\end{cases}$$

such that

$$\forall\ell \in [L],\ \forall d \in [D]:\ \mathcal{T}_{X[:,:,\ell,d]} = \widetilde{\mathcal{F}}_X\left(E^{\ell,d}\right). \tag{28}$$

This allows computing the entire tensor using $L \cdot D$ (possibly batched) forward passes, trading GPU memory for compute time.

D Relation Decoding Experiment

We mostly adopt the experimental setup and relations dataset introduced in Hernandez et al. (2024), using the relation categories of *bias*, *common sense*, and *factual*. Although, in order to adapt to our tensor method we introduce several changes: **(i)** In order to perform the mean tensor approximation in Eq (26), we must filter the samples within each relation to those of the most common token length. **(ii)** Due to limited academic computational resources, we evaluate on Pythia-1B, which is a smaller model than used by Hernandez et al. (2024). **(iii)** Since we use a smaller LM, we further filter the test samples only to those in which the correct object is within the top-20 tokens predicted by the model.

For each relation type we use $m = 3$ training examples to compute the mean tensor, and the LRE weights of Hernandez et al. (2024), and report results averaged on 6 random seeds of train-test splits.

For the mean tensor calculation in Eq. (26), we use the full affine transformation with biases described in Appendix B to obtain:

$$\widetilde{\mathcal{T}}_r = \frac{1}{m}\sum_i (\mathcal{T}_{X_i} + B_i)\,.$$

This is necessary to obtain an accurate approximation of the original model.

Although the LRE method was developed for extracting linear relations from intermediate hidden-state, we compare it to ours using the input embeddings for a legitimate comparison. This method was shown as an ablation in their work. Additionally, the LRE baseline requires an additional hyper-parameter β which scales the Jacobian matrices in their method. We use their default value for Pythia's GPT-NeoX architecture of $\beta = 2.5$, which gave the best results in a grid-search of other proposed values in their repository.

E Tensor Approximation Error Bound

Proposition 1 states that the **approximation error** of the tensor $\mathcal{T}_X$ computed on input X, when evaluating the transformer function $\mathcal{F}$ at $(X+\epsilon)$ is bounded by:

$$\|\mathcal{T}_X(X+\epsilon) - \mathcal{F}(X+\epsilon)\|_2 \le \|\mathcal{T}_X\|_2 \|\epsilon\|_2 + \|\mathcal{F}(X+\epsilon) - \mathcal{F}(X)\|_2\,. \tag{29}$$

Here we prove this claim and provide the explicit bound on the spectral norm of the tensor $\|\mathcal{T}_X\|_2$, when flattened as a matrix in $\mathbb{R}^{LD\times LD}$.

First, using the full derivation with biases in Appendix B, for any input embedding $X \in \mathbb{R}^{L\times D}$ and perturbation $\epsilon \in \mathbb{R}^{L\times D}$ we have:

$$\begin{aligned}
&\|\mathcal{T}_X(X+\epsilon) - \mathcal{F}(X+\epsilon)\|_2 \\
&= \|\mathcal{T}_X \,\mathrm{vec}\,[X+\epsilon] + \mathrm{vec}\,[B_X] - \mathrm{vec}\,[\mathcal{F}(X+\epsilon)]\| \\
&= \|\mathcal{T}_X \,\mathrm{vec}\,[\epsilon] - \mathrm{vec}\,[\mathcal{F}(X) - \mathcal{F}(X+\epsilon)]\|_2 \\
&\le \|\mathcal{T}_X\|_2 \|\epsilon\|_2 + \|\mathcal{F}(X+\epsilon) - \mathcal{F}(X)\|_2
\end{aligned} \tag{30}$$

To bound $\|\mathcal{T}_X\|_2$, we bound the spectral norm of the tensor of each sub-module of the transformer block when flattened as a matrix (as defined in Section 3), and then combine the bounds within the block and across layers.

Self Attention. Denoting the combined value-output projection per head $W_{v,h}W_{o,h}$ as W^h_{vo} we get:

$$\begin{aligned}
\|\mathcal{A}\|_2 &= \left\| \sum_{h\in H} \left(W^h_{vo}\right)^\top \otimes A^h \right\|_2 \\
&\le \sum_{h\in H} \left\|W^h_{vo}\right\|_2 \sqrt{\|A^h_x\|_1 \|A^h_x\|_\infty} \\
&\le \sqrt{L} \sum_h \left\|W^h_{vo}\right\|_2
\end{aligned} \tag{31}$$

FFN. The tensor of the FFN block for input X is defined as:

$$\mathcal{M} = \left(M_2^\top \otimes I_L\right)\mathrm{diag}\left(\mathrm{vec}\left[\frac{\phi(X)}{X}\right]\right)\left(M_1^\top \otimes I_L\right),$$

with the element-wise activation function ϕ. Standard choices of ϕ such as GELU and SilU follow $\phi(x) \leq x$, so we have:

$$\left\|\mathrm{diag}\left(\mathrm{vec}\left[\frac{\phi(X)}{X}\right]\right)\right\|_2 = \left\|\frac{\phi(X)}{X}\right\|_\infty \leq 1$$

Overall for the whole FFN:

$$\|\mathcal{M}\|_2 \leq \|M_2\|_2 \|M_1\|_2 \tag{32}$$

LayerNorm. The LayerNorm tensor is defined as:

$$\mathcal{L}_X = \left[(I_D - \tfrac{\mathbf{1}\mathbf{1}^\top}{D})\mathrm{diag}(\gamma)\right]^\top \otimes \mathrm{diag}\left(\tfrac{1}{\sigma_X}\right),$$

For the left side of the mean centering matrix and γ we have:

$$\left\|I_D - \tfrac{\mathbf{1}\mathbf{1}^\top}{D}\right\|_2 \leq 1\,,\ \|\mathrm{diag}(\gamma)\|_2 = \|\gamma\|_\infty$$

And for the right side of the variance $\sigma_X \in \mathbb{R}^{L\times L}$ we have:

$$\left\|\mathrm{diag}\left(\tfrac{1}{\sigma_X}\right)\right\|_2 = \left\|\tfrac{1}{\sigma_X}\right\|_\infty = \max_{l\in L}\frac{1}{\mathrm{Var}\left[X_{[l,:]}\right]}$$

Where $\mathrm{Var}\left[X_{[l,:]}\right]$ is the variance of X at position $l \in L$. Importantly, this is the only data-dependent quantity in our bound, depending on the minimal variance of each of the hidden-states $X_{[l,:]} \in \mathbb{R}^D$ at the input to the layer norm. We denote it as

$$\xi_{(\mathrm{LN},X)} = \min_{l\in L}\mathrm{Var}\left[X_{[l,:]}\right],$$

and the overall bound for the LayerNorm tensor is:

$$\|\mathcal{L}_x\|_2 \leq \frac{\|\gamma\|_\infty}{\xi_{(\mathrm{LN},X)}}\,. \tag{33}$$

Whole Transformer. The tensor of a post-layernorm transformer layer $n \in N$ is

$$\mathcal{T}^n = \mathcal{L}_2^n\left(\mathcal{M}^n + \mathcal{I}\right)\mathcal{L}_1^n\left(\mathcal{A}^n + \mathcal{I}\right).$$

Combining the bounds of each component from Eq. (31),(32),(33) we get: for the whole transformer:

$$\begin{aligned}\|\mathcal{T}_X\|_2 &\leq \prod_{n=1}^{N}\|\mathcal{L}_2^n\|\left(\|\mathcal{M}^n\|+1\right)\|\mathcal{L}_1^n\|\left(\|\mathcal{A}^n\|+1\right)\\ &\leq \prod_{n=1}^{N}\frac{\|\gamma_2^n\|_\infty}{\xi_{(\mathrm{LN}_2^n,X)}}\left(\|M_1^n\|_2\|M_2^n\|_2+1\right)\\ &\quad\cdot\frac{\|\gamma_1^n\|_\infty}{\xi_{(\mathrm{LN}_1^n,X)}}\left(\sqrt{L}\sum_h\left\|W_{vo}^h\right\|_2+1\right)\end{aligned} \tag{35}$$

Together with Eq. 30, this completes the proof of Proposition 1.

We note that although this bound is data-dependent, the value of

$$\|\mathcal{L}_x\|_2 \leq \frac{\|\gamma\|_\infty}{\xi_{(\mathrm{LN},X)}}$$

is typically a small constant.