LaTIM: Measuring Latent Token-to-Token Interactions in Mamba Models

Hugo Pitorro, Marcos Treviso
Instituto de Telecomunicações, Lisbon
hugo.pitorro@lx.it.pt, marcos.treviso@co.it.pt

Abstract

State space models (SSMs), such as Mamba, have emerged as an efficient alternative to transformers for long-context sequence modeling. However, despite their growing adoption, SSMs lack the interpretability tools that have been crucial for understanding and improving attention-based architectures. While recent efforts provide insights into Mamba's internal mechanisms, they struggle to capture precise token-level interactions at the layer level, leaving gaps in understanding how Mamba *selectively* processes sequences across layers. In this work, we introduce LaTIM, a novel token-level decomposition method for both Mamba-1 and Mamba-2 that enables fine-grained interpretability. We extensively evaluate our method across diverse tasks, including machine translation, copying, and retrieval-based generation, demonstrating its effectiveness in revealing Mamba's token-to-token interaction patterns. Our code is available at `https://github.com/deep-spin/latim`.

1 Introduction

State space models (SSMs), such as S4 (Gu et al., 2022), have emerged as a promising alternative to transformers for long-context modeling. Unlike transformers (Vaswani et al., 2017), which explicitly compute pairwise token interactions and require quadratic memory, SSMs leverage structured recurrence mechanisms that enable more efficient sequence processing. Among them, the Mamba architecture (Gu and Dao, 2023; Dao and Gu, 2024) has demonstrated strong performance in language modeling and other modalities while significantly reducing runtime and memory requirements (Xu et al., 2024). Additionally, hybrid architectures that integrate both Mamba and attention mechanisms often outperform purely homogeneous models by combining the efficiency of recurrence with the expressivity of attention (Lenz et al., 2025; Dong et al., 2025; Pitorro et al., 2024). While these findings highlight the relevance of Mamba models, their internal decision-making processes remain opaque, hindering their reliability.

Interpretability techniques have played a key role in the widespread adoption of transformers, enabling researchers to analyze token interactions and information flow (Mohebbi et al., 2024; Ferrando et al., 2024). However, in contrast to transformers, where attention scores offer a direct visualization of how the model distributes importance across tokens, Mamba lacks an explicit mechanism to reveal where it is "attending" at each step. Existing interpretability efforts for Mamba attempt to bridge this gap by reformulating its computations into implicit attention matrices (Ali et al., 2024) or rely on layer-wise propagation analysis to track gradient flow (Jafari et al., 2024). However, these methods have difficulties in capturing precise, fine-grained individual token-wise contributions across layers, leaving a gap in understanding how Mamba *selectively* processes sequences.

In this work, we bridge this gap by introducing LaTIM, a novel token-level decomposition method for both Mamba-1 and Mamba-2. Our approach reformulates the SSM computation to enable token-by-token analysis, allowing us to adapt attention-based interpretability techniques, such as ALTI (Ferrando et al., 2022), to the Mamba architecture. We extensively evaluate our method across diverse tasks, including the copying task (Jelassi et al., 2024) in §4.1, which features a well-defined diagonal attention pattern; machine translation in §4.2, where precise source↔target alignment is essential; and retrieval-based generation (Hsieh et al., 2024) in §4.3, where ground-truth context allows direct evaluation of token importance. Our method not only improves Mamba's interpretability but also defines a robust framework for analyzing token interactions in SSMs, paving the way for more transparent models.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 24478–24493

2 Background

2.1 Transformers

A key component in the transformer architecture is the attention mechanism, which is responsible for mixing input sequences $\boldsymbol{X} = \langle \boldsymbol{x}_1, ..., \boldsymbol{x}_N \rangle$, where each $\boldsymbol{x}_i \in \mathbb{R}^D$. Concretely, given query $\boldsymbol{Q}^h = \boldsymbol{X}\boldsymbol{W}_q^h \in \mathbb{R}^{N \times D'}$, key $\boldsymbol{K}^h = \boldsymbol{X}\boldsymbol{W}_k^h \in \mathbb{R}^{N \times D'}$, and value $\boldsymbol{V}^h = \boldsymbol{X}\boldsymbol{W}_v^h \in \mathbb{R}^{N \times D'}$ matrices as input, where $1 \leq h \leq H$ is the head dimension, the *multi-head attention mechanism* is defined as follows (Vaswani et al., 2017):

$$\mathsf{Attn}(\boldsymbol{X})_h = \underbrace{\pi\left(\frac{\boldsymbol{Q}^h \boldsymbol{K}^{h^\top}}{\sqrt{D'}}\right)}_{\boldsymbol{A}^h \in \mathbb{R}^{N \times N}} \boldsymbol{V}^h \in \mathbb{R}^{N \times D'}, \quad (1)$$

where π maps rows to distributions, with $\pi :=$ softmax being a common choice.

Transformer block. The attention is combined with other modules in order to form a transformer block. The full block, with pre LayerNorm (LN, Ba et al. 2016), can be described as follows:

$$\begin{aligned} \boldsymbol{X}_l &= \mathsf{LN}(\boldsymbol{X}) && \in \mathbb{R}^{N \times D}, \quad (2) \\ \boldsymbol{Y}_a &= \mathsf{Concat}(\mathsf{Attn}(\boldsymbol{X}_l)_h \boldsymbol{W}_o^h), && \in \mathbb{R}^{N \times D}, \\ \boldsymbol{Y} &= \boldsymbol{Y}_a + \boldsymbol{X} && \in \mathbb{R}^{N \times D}, \end{aligned}$$

where we denote $\mathsf{Concat}(\cdot)$ as the concatenation of all heads $1 \leq h \leq H$, and $\boldsymbol{W}_o \in \mathbb{R}^{D' \times D}$. In words, the attention output is projected through $\boldsymbol{W}_O^h$ and, together with a residual stream and pre-layer norm, forms the output of the block.

2.2 Attention Decomposition

Transformers benefit from attention maps for interpretability, but these do not fully capture token influence on predictions. Token attribution methods address this by decomposing the forward pass into token-wise contributions (Kobayashi et al., 2021). This section presents two key approaches—direct token-to-token decomposition and logit attribution—which motivate our interpretability method for Mamba.

Token Contributions. To determine the influence of token j on the representation of token i, we express the output at a certain layer as follows:[1]

$$\boldsymbol{y}_i = \sum_{j=1}^{N} T_i(\boldsymbol{x}_j) \in \mathbb{R}^D, \quad (3)$$

where the transformed contribution of $\boldsymbol{x}_j$ to $\boldsymbol{y}_i$ is

$$T_i(\boldsymbol{x}_j) = \sum_{h=1}^{H} \boldsymbol{W}_o^h \boldsymbol{A}_{i,j}^h \boldsymbol{W}_v^h \cdot \mathsf{LN}(\boldsymbol{x}_j) + \delta_{i,j} \boldsymbol{x}_i, \quad (4)$$

with $\delta_{i,j}$ denoting the Kronecker delta.

Token-to-Token Importance. Using this decomposition, we can obtain token-to-token importance scores via vector norms (Kobayashi et al., 2021):

$$C_{i,j} = \left\| T_i(\boldsymbol{x}_j) \right\|_2, \quad (5)$$

or via ALTI's contextual mixing approach (Ferrando et al., 2022):

$$C_{i,j} = \frac{[\|\boldsymbol{y}_i\|_1 - \|\boldsymbol{y}_i - T_i(\boldsymbol{x}_j)\|_1]_+}{\sum_k [\|\boldsymbol{y}_i\|_1 - \|\boldsymbol{y}_i - T_i(\boldsymbol{x}_k)\|_1]_+}, \quad (6)$$

where $[\cdot]_+$ represents the ReLU function.

Logit Contributions. While token-wise decomposition methods capture interactions within a layer, they do not measure a token's direct contribution to the final output. To bridge this gap, ALTI-Logit (Ferrando et al., 2023) traces token contributions through the residual stream up to the final prediction. Formally, given a token $w(i) \in \mathcal{V}$, the contribution of token j at layer l is given by:

$$\Delta_{i,j}^{(l)} = T_i^{(l)}(\boldsymbol{x}_j^{(l-1)})^\top \boldsymbol{U}_{w(i)}, \quad (7)$$

where $\boldsymbol{U} \in \mathbb{R}^{|\mathcal{V}| \times D}$ is the output embedding matrix. Let $\boldsymbol{R}^{(l)} = \boldsymbol{P}^{(l)} \cdots \boldsymbol{P}^{(2)} \boldsymbol{P}^{(1)}$ denote the residual stream at layer l, where $P_{i,j}^{(l)}$ refers to the contribution of $\boldsymbol{x}_i^{(l-1)}$ to $\boldsymbol{x}_j^{(l)}$ such that $\sum_j P_{i,j}^{(l)} = 1$. Then, the final pairwise contribution score aggregated from all L layers is

$$C_{i,j} = \sum_{l=1}^{L} \Delta_{i,j}^{(l)} \boldsymbol{R}_j^{(l-1)}. \quad (8)$$

ALTI-Logit provides a final-layer attribution score, making it particularly useful for output-sensitive interpretability. In Section 3, we follow these principles to design attribution methods for Mamba.

2.3 State Space Models (SSMs)

SSMs (Gu et al., 2020) are a type of sequence mixing layer that process sequences through a linear recurrence. Letting $\boldsymbol{H}_i \in \mathbb{R}^{R \times D}$ denote the "state"

[1] We ignore the bias terms for clarity (w.l.o.g). Moreover, in a decoder-only model we have $1 \leq j \leq i$.

at the i^{th} time step, a discrete SSM can be formulated as follows (Pitorro et al., 2024):[2]

$$\begin{aligned} \boldsymbol{H}_i &= \boldsymbol{A}\boldsymbol{H}_{i-1} + \boldsymbol{b}\boldsymbol{x}_i^\top && \in \mathbb{R}^{R\times D}, \\ \boldsymbol{v}_i &= \boldsymbol{H}_i^\top \boldsymbol{c} + \boldsymbol{D}\boldsymbol{x}_i && \in \mathbb{R}^{D}, \end{aligned} \tag{9}$$

where $\boldsymbol{A} \in \mathbb{R}^{R\times R}$, $\boldsymbol{b} \in \mathbb{R}^R$, $\boldsymbol{c} \in \mathbb{R}^R$, $\boldsymbol{D} \in \mathbb{R}^{D\times D}$ are (discrete) parameters shared for all i.

Mamba-1. The first version of Mamba (Gu and Dao, 2023) extends the previous formulation into an *input-dependent* SSM by turning the parameters into learnable projections of the current input $\boldsymbol{x}_i$:

$$\begin{aligned} \boldsymbol{H}_i &= \boldsymbol{A}_i \odot \boldsymbol{H}_{i-1} + \boldsymbol{B}_i \odot \boldsymbol{X}_i && \in \mathbb{R}^{R\times D}, \\ \boldsymbol{v}_i &= \boldsymbol{H}_i^\top \boldsymbol{c}_i + \boldsymbol{D}\boldsymbol{x}_i && \in \mathbb{R}^{D}, \end{aligned} \tag{10}$$

where $\boldsymbol{X}_i = \boldsymbol{1}_r \boldsymbol{x}_i^\top \in \mathbb{R}^{R\times D}$ is an R-sized stack of the input, $\boldsymbol{A}_i \in \mathbb{R}^{R\times D}$ represents D diagonal matrices of size $R \times R$, $\boldsymbol{B}_i \in \mathbb{R}^{R\times D}$, $\boldsymbol{c}_i \in \mathbb{R}^R$, and $\odot$ is the Hadamard product.

Mamba-1 block. Analogously to transformers, the Mamba-1 model is a collection of stacked blocks containing a sequence mixing layer and a gating mechanism. Concretely, the sequence mixing layer can be fully described as:

$$\begin{aligned} \boldsymbol{\Psi} &= \mathsf{Conv1D}(\boldsymbol{X}\boldsymbol{W}_x) && \in \mathbb{R}^{N\times 2D}, \\ \boldsymbol{\Phi} &= \mathsf{SiLU}(\boldsymbol{\Psi}) && \in \mathbb{R}^{N\times 2D}, \\ \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} &= \mathsf{Linear}(\boldsymbol{\Phi}) && \in \mathbb{R}^{N\times R}, \\ \boldsymbol{\Upsilon} &= \mathsf{SSM}(\boldsymbol{\Phi}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}) && \in \mathbb{R}^{N\times 2D}, \end{aligned} \tag{11}$$

where $\boldsymbol{W}_x \in \mathbb{R}^{D\times 2D}$ and $\mathsf{Linear} : \mathbb{R}^{N\times 2D} \to \mathbb{R}^{N\times R}$ represents a set of low-rank projections. The gating mechanism is employed as follows:

$$\begin{aligned} \boldsymbol{Z} &= \mathsf{SiLU}(\boldsymbol{X}\boldsymbol{W}_z) && \in \mathbb{R}^{N\times 2D}, \\ \boldsymbol{U} &= \boldsymbol{\Upsilon} \odot \boldsymbol{Z} && \in \mathbb{R}^{N\times 2D}, \\ \boldsymbol{Y} &= \boldsymbol{U}\boldsymbol{W}_o && \in \mathbb{R}^{N\times D}, \end{aligned} \tag{12}$$

where $\boldsymbol{W}_z \in \mathbb{R}^{D\times 2D}$ and $\boldsymbol{W}_o \in \mathbb{R}^{2D\times D}$.

Mamba-2. Mamba-2 (Dao and Gu, 2024) introduces a simpler SSM formulation by defining $\boldsymbol{A}$ as a scalar times identity $\boldsymbol{A}_i = a_i \boldsymbol{I}_{R\times R}$. This leads to the following *input-dependent* model:

$$\begin{aligned} \boldsymbol{H}_i &= \boldsymbol{A}_i \boldsymbol{H}_{i-1} + \boldsymbol{B}_i \odot \boldsymbol{X}_i && \in \mathbb{R}^{R\times D}, \\ \boldsymbol{v}_i &= \boldsymbol{H}_i^\top \boldsymbol{c}_i + \boldsymbol{D}\boldsymbol{x}_i && \in \mathbb{R}^{D}. \end{aligned} \tag{13}$$

In contrast to Mamba-1 (*c.f.* Equation 10), the input-dependent parameter $\boldsymbol{A}_i \in \mathbb{R}^{R\times R}$ represents a single diagonal matrix.

[2] A discretization step is required to obtain discrete parameters (e.g., via the zero-order hold rule); however, we follow Pitorro et al. (2024) and omit this step for clarity.

Mamba-2 block. Regarding block structure, Mamba-2 draws the parameters $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ directly from the initial input $\boldsymbol{X}$, and further introduces a GroupNorm layer (Wu and He, 2018) after the gating mechanism for additional stability:

$$\boldsymbol{U} = \mathsf{GroupNorm}(\boldsymbol{\Upsilon} \odot \boldsymbol{Z}) \quad \in \mathbb{R}^{N\times 2D}. \tag{14}$$

2.4 Hidden Attention in Mamba

As noted by Ali et al. (2024) and Dao and Gu (2024), by unrolling Mamba's recurrence we can interpret the sequence mixing layer as multiplying a lower-triangular matrix $\boldsymbol{M}$ with the entire input $\boldsymbol{\Upsilon} = \boldsymbol{M}\boldsymbol{X}$ (independently for each channel/head). More generally, by unrolling Mamba-1's recurrence defined in Equations 10, we can show that $\boldsymbol{M}_{i,j} \in \mathbb{R}^{D\times D}$ has the following form:

$$\boldsymbol{M}_{i,j} = \mathsf{Diag}\left(\left[\left(\bigodot_{k=j+1}^{i} \boldsymbol{A}_k \right) \odot \boldsymbol{B}_j \right]^\top \boldsymbol{c}_i \right), \tag{15}$$

for all $j \le i$, and $\boldsymbol{M}_{i,j} = \boldsymbol{0}$ otherwise. A similar expression can be derived for Mamba-2 by noticing that $\boldsymbol{A}_k \in \mathbb{R}^{R\times R}$ is, by definition, a diagonal matrix. Importantly, for each dimension $d \in [D]$, this is an *implicit attention* matrix akin to transformers' attention matrix. We provide more details on this derivation in App. A.

3 LaTIM

While the attention mechanism found in transformers allows us to decompose the contributions of different input tokens, decomposing individual token contributions is challenging for Mamba. Additionally, in Mamba-1 the channel dimensionality is often large in practice, and therefore manual inspection of all attention maps per layer and sample quickly becomes unfeasible (e.g., a 370M model has 48 layers with $D = 1\,024$). Although Mamba-2 alleviates this issue by using a smaller number of heads, it remains unclear how to obtain a single attention plot for each layer or for an entire sample. Overall, our goal is to rearrange the forward pass from both Mamba-1 and Mamba-2 such that we can measure the total contribution of token $\boldsymbol{x}_j$ towards the output $\boldsymbol{y}_i$, akin to the definition of $T_i(\boldsymbol{x}_j)$ in Equation 4 tailored for transformers.

3.1 Mamba-1 Decomposition

In this direction, we start by revisiting Mamba's forward pass at step i in Equation 11. The first component of Mamba-1 block is the 1D convolution layer. Concretely, letting $w \in \mathbb{N}$ denote the kernel size, the 1D causal convolution output for a token i can be described as:

$$\boldsymbol{\psi}_i = \mathrm{Conv1D}\left(\boldsymbol{X}\boldsymbol{W}_x; w\right)_i \tag{16}$$

$$= \sum_{k=1}^{w} \boldsymbol{W}_c^{(k)} \left(\boldsymbol{W}_x^\top \boldsymbol{x}_{i-w+k}\right) + \boldsymbol{b}_c, \tag{17}$$

where $\boldsymbol{W}_c^{(k)} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}_c$ represents the convolution kernel and bias, respectively. Next, $\boldsymbol{\psi}_i$ is transformed via a SiLU activation $\boldsymbol{\phi}_i = \mathsf{SiLU}(\boldsymbol{\psi}_i)$, which, in turn, is passed to the SSM module, $\boldsymbol{v}_i = \mathsf{SSM}(\boldsymbol{\phi}_i)$. Therefore, in order to compute the contribution of token-to-token interactions, we first need to unroll the SSM recurrence from Equation 10. To that end, we leverage the tensor $\boldsymbol{M}$ defined in Equation 15 and treat the term $\boldsymbol{D}\boldsymbol{\phi}_i$ as a skip-connection, leading to:

$$\boldsymbol{v}_i = \sum_{j=1}^{i} (\boldsymbol{M}_{i,j} + \delta_{i,j}\boldsymbol{D}) \underbrace{\boldsymbol{\phi}_j}_{\mathsf{SiLU}(\boldsymbol{\psi}_j)}, \tag{18}$$

where $\delta_{i,j}$ is the Kronecker delta. Unfortunately, the non-additivity of the SiLU activation prevents the decomposition of $\boldsymbol{v}_i$ as a sum of previous token interactions. That is, we cannot rearrange the above expression such that we use the j^{th} token only at the j^{th} iteration, prohibiting us from deriving token-to-token contributions as done in transformers (see Section 2.2). However, if we assume the existence of an additive function f that approximates well the SiLU activation, we can decompose $\boldsymbol{\phi}_j$ as follows:[3]

$$\boldsymbol{\phi}_j = \sum_{k=1}^{w} f(\underbrace{\boldsymbol{W}_c^{(k)}\boldsymbol{W}_x^\top \boldsymbol{x}_{j-w+k} + \delta_{k,0}\boldsymbol{b}_c}_{\boldsymbol{\varphi}_j^{(k)}}). \tag{19}$$

This decomposition allows us to derive a more *interpretable* output for Mamba's recurrent module:

$$\boldsymbol{v}_i = \sum_{j=1}^{i}\sum_{k=1}^{w} (\boldsymbol{M}_{i,j+k} + \delta_{i,j+k}\boldsymbol{D})\, \boldsymbol{\varphi}_j^{(k)}. \tag{20}$$

Importantly, we can modify the above expression in order to obtain the vector representation that stems from interactions with the j^{th} token as follows:

$$\boldsymbol{v}_{i\leftarrow j} = \sum_{k=1}^{w} (\boldsymbol{M}_{i,j+k} + \delta_{i,j+k}\boldsymbol{D})\, \boldsymbol{\varphi}_j^{(k)}. \tag{21}$$

[3]We explicitly include $\delta_{j,0}$ into the expression to account for the convolution bias, which is only added once per channel.

Method	Expression
LATIM (ℓ_p)	$C_{ij} = \lVert T_i(\boldsymbol{x}_j)\rVert_p$
LATIM (ALTI)	$C_{ij} \propto [\lVert\boldsymbol{y}_i\rVert_1 - \lVert\boldsymbol{y}_i - T_i(\boldsymbol{x}_j)\rVert_1]_+$
LATIM (ALTI-Logit)	$C_{ij} = T_i(\boldsymbol{x}_j)^\top \boldsymbol{U}_{w(i)}\boldsymbol{R}_j$

Table 1: Overview of LATIM-based methods for obtaining contribution scores for (i, j) token interactions.

Finally, after considering the gating mechanism and the output projection from Equation 12, we obtain the (i, j) contribution vector:

$$T_i(\boldsymbol{x}_j) = \boldsymbol{W}_o^\top \left(\boldsymbol{Z}_i \odot \boldsymbol{v}_{i\leftarrow j}\right). \tag{22}$$

And similarly to the way attention is decomposed in transformers (see Equations 3 and 4), the final output can be computed by integrating the contribution from all previous tokens:

$$\boldsymbol{y}_i = \sum_{j=1}^{i} T_i(\boldsymbol{x}_j). \tag{23}$$

3.2 Mamba-2 Decomposition

Recall from Equation 14 that Mamba-2 places a GroupNorm layer on the output of the SSM module. Let $\boldsymbol{v}_i \in \mathbb{R}^{2D}$ be the SSM output at token i, and define $\boldsymbol{u}_i = \boldsymbol{Z}_i \odot \boldsymbol{v}_i$. At test time, GroupNorm can be viewed as an affine map around $\boldsymbol{u}_i$,

$$\mathsf{GroupNorm}(\boldsymbol{u}_i) = \gamma_i(\boldsymbol{u}_i)\,\boldsymbol{u}_i + \beta_i, \tag{24}$$

where $\gamma_i(\boldsymbol{u}_i)$ is a (fixed) linear operator once $\boldsymbol{u}_i$ is known, and β_i is an offset.[4] Hence, if $\boldsymbol{u}_{i\leftarrow j}$ denotes the portion of $\boldsymbol{u}_i$ that originates from token j, its contribution passes through GroupNorm in the same linear fashion. Finally, applying the output projection $\boldsymbol{W}_o$ yields the token decomposition:

$$T_i(\boldsymbol{x}_j) = \boldsymbol{W}_o^\top \Big[\gamma_i\big(\boldsymbol{u}_i\big)\,\boldsymbol{u}_{i\leftarrow j}\Big]. \tag{25}$$

As we are interested in obtaining token-to-token interpretability scores, we can apply various scalar aggregation functions to $T_i(\boldsymbol{x}_j)$. Common examples include ℓ_1 or ℓ_2 norms (Kobayashi et al., 2021), as well as the ALTI (Ferrando et al., 2022) and ALTI-Logit (Ferrando et al., 2023) approaches. We provide a summary of LATIM variants that leverage these aggregations in Table 1.

[4]We follow Ferrando et al. (2022) and ignore the offset term as it not attributed to any token.

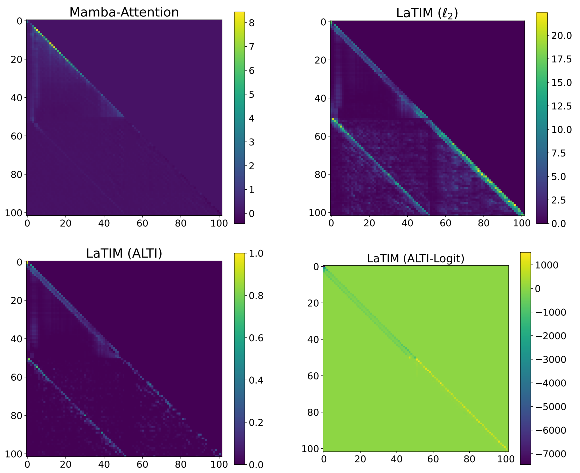

Figure 1: Heatmaps generated by different interpretability methods for Mamba-2. The interaction between source and copied tokens (along the diagonal line) is more clearly highlighted with LATIM.

3.3 Decomposition Error

Approximated Strategy. Unlike attention decomposition in transformers, Mamba requires an additive function f in Equation 19 to linearly decompose pairwise interactions. Ideally, f should closely approximate the original non-additive expression $\phi_i = \mathsf{SiLU}(\psi_i)$. To assess this, we explore different approximation strategies in Appendix C, including first- and second-order Taylor expansions around zero. Surprisingly, we find that directly setting f as SiLU yields the lowest approximation error across all layers. Therefore, unless explicitly stated otherwise, LATIM refers to our decomposition method using $f := \mathsf{SiLU}$.

Exact Strategy. While a well-chosen approximation function f enables interpretability without requiring model retraining, it does not fully recover the exact Mamba block's output. To eliminate this discrepancy, we suggest a modified version of Mamba that removes the SiLU activation, simplifying the computation to $\phi_i = \psi_i$, which effectively turns f into the identity function in Equation 19. Though this approach requires an extra training step, we demonstrate in Section 4.4 that it achieves zero decomposition error while maintaining the same level of interpretability and task performance.

4 Experiments

Tasks and Metrics. We adopt a diverse set of tasks to provide a rigorous evaluation. Following Jelassi et al. (2024) we experiment on the Copying Task, a synthetic benchmark that tests sequence recall and allows us to faithfully assess how different methods capture token interactions (Bastings et al., 2022). Next, we follow (Kobayashi et al., 2020) and (Ferrando et al., 2022) and analyze machine translation (MT), where we use the alignment error rate (AER) metric to quantitatively compare the performance of interpretability approaches. Finally, we explore retrieval-based generation, leveraging the RULER benchmark (Hsieh et al., 2024) to investigate Mamba's selective processing in real-world recall-intensive tasks.

Method	**AUC**	**AP**	**R@K**
Mamba-1:			
Mamba-Attention	0.84	0.36	0.22
Mamba-Attribution	0.83	0.31	0.19
MambaLRP	0.40	0.22	0.20
LATIM (ℓ_2)	**0.88**	0.41	0.27
LATIM (ALTI)	0.86	**0.47**	**0.36**
LATIM (ALTI-Logit)	0.85	0.44	0.31
Mamba-2:			
Mamba-Attention	0.79	0.49	0.39
Mamba-Attribution	0.79	0.47	0.39
LATIM (ℓ_2)	**0.98**	**0.86**	**0.74**
LATIM (ALTI)	0.85	0.71	0.63
LATIM (ALTI-Logit)	0.87	0.70	0.61

Table 2: Faithfulness evaluation on the copying task in terms of Area Under the Curve (AUC), Average Precision (AP), and Recall at K (R@K).

Models. For machine translation and retrieval-based generation, we use pre-trained versions of Mamba-1 and Mamba-2. For the copying task, we train our models from scratch. Training details for all tasks are provided in Appendix B.

Methods. To evaluate the effectiveness of LATIM, we conduct both qualitative and quantitative assessments, comparing it against existing interpretability techniques for Mamba. Namely, we compare our approach against MambaLRP (Jafari et al., 2024) when using Mamba-1,[5] and with Mamba-Attention/Attribution (Ali et al., 2024) for both Mamba-1 and Mamba-2. Regarding LATIM, we experiment with the variants shown in Table 1.

4.1 Copying

The synthetic copying task (Jelassi et al., 2024) serves as a controlled setting for testing memory recall in SSM-based models, which traditionally struggle with maintaining long-range dependencies (Arora et al., 2024). Recent advances, such as

[5] MambaLRP is only defined for Mamba-1.

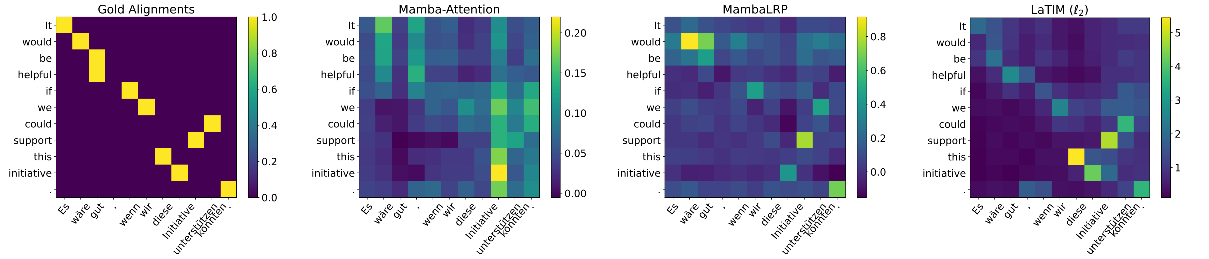

Figure 2: Interpretability heatmaps for Mamba-1 (370M) fine-tuned on DE→EN data from the IWSLT17 dataset. LATIM (ℓ_2) produces alignments that more closely match the ground truth.

	GoldAlign (DE→EN)				IWSLT17 (DE→EN)				IWSLT17 (FR→EN)			
Method	$M1_S$	$M1_L$	$M2_S$	$M2_L$	$M1_S$	$M1_L$	$M2_S$	$M2_L$	$M1_S$	$M1_L$	$M2_S$	$M2_L$
Aggregating layers:												
MambaLRP	0.50	0.47	-	-	0.65	0.68	-	-	0.65	0.66	-	-
LATIM (ALTI-Logit)	0.68	0.69	0.63	0.69	0.67	0.71	0.60	0.74	0.71	0.69	0.62	0.76
Best layer:												
Mamba-Attention	0.84	0.85	0.84	0.85	0.79	0.79	0.72	0.79	0.80	0.79	0.69	0.78
Mamba-Attribution	0.86	0.87	0.78	0.70	0.81	0.82	0.81	0.81	0.73	0.68	0.72	0.66
LATIM (ℓ_2)	**0.46**	**0.44**	**0.49**	0.52	**0.47**	**0.49**	**0.43**	0.49	**0.46**	**0.48**	**0.35**	**0.37**
LATIM (ALTI)	0.55	0.54	0.51	**0.51**	0.52	0.53	0.47	**0.47**	0.53	0.53	0.38	0.38

Table 3: Alignment Error Rate (AER) per interpretability method. M1 and M2 stand for Mamba-1 and Mamba-2, with subscript S and M denoting the small (130M) and large (370M) versions, respectively.

the mimetic initialization proposed by Trockman et al. (2024), have significantly improved Mamba's performance on this task. We replicate this setup in a smaller-scale experiment, where 13M parameter models (Mamba-1 and 2) are trained to repeat a 50-token string after a separator token: `source <SEP> copy`.

Qualitative Analysis. Our interpretability analysis focuses on whether different methods can recover the expected diagonal interaction pattern between source and copied tokens. To that end, we start by qualitatively inspecting each method's heatmap in Figure 1 for Mamba-2.[6] We observe that Mamba-Attention produces a coarse representation of the copy mechanism, lacking the precision needed to capture token-level dependencies. In contrast, all LATIM variants better highlight source→copy interactions, making it the superior choice for visualizing the copying mechanism.

Faithfulness Evaluation. To quantitatively assess the reliability of each method, we use a ground-truth matrix with ones along the three main diagonals. This means that a faithful interpretability method should produce a well-defined diagonal pattern, indicating that the model correctly attends to preceding tokens, even when shifted, during the copying process. Leveraging the interpretability metrics from Fomicheva et al. (2021), we report a faithfulness evaluation in Table 2. The results show that all variants of LATIM outperform the baselines, with LATIM (ℓ_2 and ALTI) consistently achieving the top results across all metrics for both Mamba-1 and Mamba-2.

4.2 Machine Translation

We evaluate our method in machine translation (MT) by fine-tuning Mamba models (with 130M and 370M parameters) on the IWSLT17 dataset DE↔EN (Cettolo et al., 2017a), following the setup from (Pitorro et al., 2024). This setup allows us to compare interpretability methods using the alignment error rate (AER), a widely used metric for measuring the accuracy of token alignments in translations.

Qualitative Analysis. We start by showing the alignments produced by Mamba-1 with the different approaches in Figure 2, along with the golden alignments provided by Vilar et al. (2006). We present additional heatmaps for all methods, including Mamba-2 plots, in Figure 7 (App. D.2). We find that token contribution heatmaps produced by LATIM (ℓ_2) are sparser and more informative than

[6]We empirically observed that Mamba-1 learns to copy at layer 4, while Mamba-2 shifts this behavior to layer 3. Thus, we extract heatmaps at these layers for the copying task.

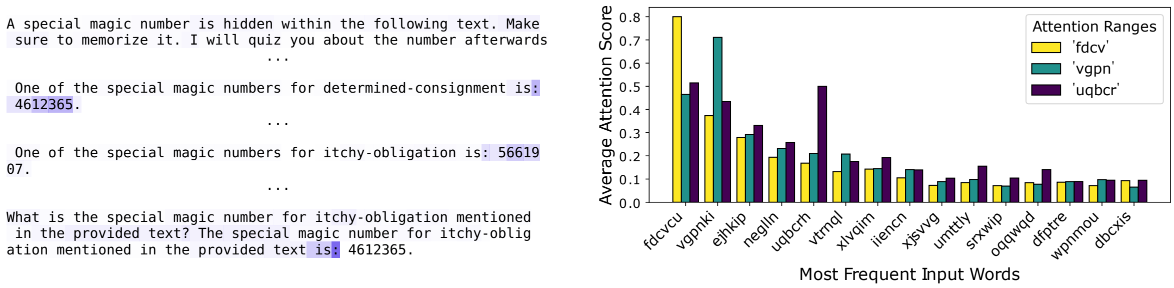

Figure 3: Left: Attention map from LATIM (ℓ_2) for a Passkey Retrieval sample where the key is "itchy-obligation" Instead of predicting 5661907, the model incorrectly produces 4612365. Right: Average contribution scores for token ranges preceding each extracted frequent word. Notably, the focus over the token ranges "fdcv" and "vgpn" aligns well with the two most frequent tokens in the sample ("fdcvcu", "vgpnki"). However, when generating "uqbcr", it fails to focus on the 3rd most frequent token, suggesting that it relies more on morphological patterns than frequency.

Mamba-Attention and MambaLRP, which captures the general structure but lacks token-level precision. Moreover, we also note that methods that aggregate input relevances across the entire model, such as LATIM (ALTI-Logit), retain sparsity but fail to capture the gold alignments.

Alignment Error Rate. To quantitatively compare methods, we further compute AER on IWSLT17 DE→EN and FR→EN using candidate alignments generated with AwesomeAlign (Dou and Neubig, 2021). As seen in Table 3, among the layer-wise aggregation methods, we note that MambaLRP consistently outperforms LATIM (ALTI-Logit). However, when looking at layer-wise methods, we find that LATIM (ℓ_2) achieves the lowest AER among all methods, reinforcing again its effectiveness in capturing token-to-token interactions, and also suggesting that translation alignments obtained on a per-layer basis might be preferable than those collapsed into a global representation.

4.3 Retrieval-based Generation

Mamba's efficiency in handling long contexts makes it an attractive candidate for retrieval-based generation. However, its ability to selectively recall relevant information remains an open question. We investigate this issue using pre-trained Mamba-2 checkpoints with various sizes and experimenting on the RULER benchmark (Hsieh et al., 2024), focusing on two recall-intensive tasks: Passkey Retrieval and Frequent Word Extraction (FWE).

Passkey Retrieval. In this task, the model must extract a numeric value associated with a key from surrounding distractor text. In our experiments, Mamba-2 consistently performed well in the simpler, single-passkey setting. However, as shown in Table 6 (App. D.3), increasing model size, sequence length, and the number of key-value pairs leads to a significant drop in recall. When analyzing attention maps for multi-key retrieval using LATIM (ℓ_2) in Figure 3 (left), we observe that the 370M model struggles to consistently focus on the correct key, revealing a potential weakness in the multi-key setting. Further pursuing this accuracy analysis in Table 4, we uncover a very strong bias towards the first key that appears throughout the sample. Specifically, for two and four key scenarios, model accuracy respectively declines by 38% and 101%, when the gold label is different from the first key. LATIM highlights this discrepancy by consistent and increased focus over the first key.

Size	2 Passkeys		4 Passkeys	
	First	Second	First	Second+
130M	74.3	41.2	46.9	22.2
370M	65.4	47.3	53.6	26.7
780M	76.9	59.1	82.0	53.4
1.4B	81.7	43.6	64.4	31.8

Table 4: Mamba-2 accuracy (%) in the Passkey Retrieval task at recovering the correct key if the correct key is the *First* to appear or the *Second+* to appear. Computed over 1000 samples of length 1024.

Frequent Word Extraction. The FWE task requires the model to extract the three most frequent synthetic words in a passage. In App. D.3, we show that Mamba models, even at the 1.4B parameter scale, struggle with this task. Our analysis in Figure 3 (right), using LATIM (ℓ_2), reveals that the model frequently misidentifies the correct 3rd most frequent token, highlighting its difficulty in tracking long-range token occurrences. We also note that Mamba's attention on repeated words decays over time, which may explain its failure to accurately count word frequency.

4.4 Approximation Error Analysis

As noted in Section 3.3, our current method requires an approximate decomposition of Mamba's computations due to the non-linearity introduced by the SiLU activation. To measure the impact of this approximation, we experiment with alternative activations by retraining Mamba with ReLU or disabling activations entirely, which casts f as the identity function and, more importantly, yields an *exact method*. We perform continued pretraining of Mamba-2 (370M) on the FineWeb-Edu dataset (Penedo et al., 2024), followed by fine-tuning and evaluation on the IWSLT17 DE→EN dataset using AER to assess interpretability and COMET (Rei et al., 2020) to asses translation quality. Results are shown in Table 5. Interestingly, a model trained without a non-linear activation achieves not only 0 approximation error but also leads to the best AER scores along with a high COMET. As noted by Bick et al. (2024), who also disable the activation before SSM distillation, a purely linear variant of Mamba can be an effective alternative for more interpretable architectures. Nonetheless, we highlight that our approximated version with $f := \mathsf{SiLU}$ leads to similar AER and COMET scores as $f := \mathsf{identity}$.

5 Related Work

Input Attribution Methods. A large body of work focuses on interpretability via input attribution, particularly in transformers, where attention maps serve as a widely used technique (Fantozzi and Naldi, 2024). While attention weights alone can be unfaithful indicators of model decisions (Jain and Wallace, 2019; Bastings and Filippova, 2020), they remain useful in many applications, including machine translation (Wiegreffe and Pinter, 2019; Treviso and Martins, 2020). Recent methods go beyond simple attention analysis by explicitly decomposing internal model computations, such as integrating value-weighted norms (Kobayashi et al., 2020) or using vector distances to estimate token contributions (Ferrando et al., 2022). Additionally, aggregation-based techniques, including Attention Rollout (Abnar and Zuidema, 2020), DiffMask (De Cao et al., 2020), and ALTI-Logit (Ferrando et al., 2023), consolidate relevance scores across multiple layers to provide a more holistic view of information flow. While these methods have substantially improved transformer interpretability, state space models (SSMs) remain comparatively underexplored.

Feature Interactions. Understanding feature interactions is crucial for model interpretability. Janizek et al. (2021) propose Integrated Hessians, a model-agnostic approach for detecting pairwise interactions in neural networks. Eberle et al. (2022) and Vasileiou and Eberle (2024) introduced and applied BiLRP to explain interaction patterns in text similarity models. Fumagalli et al. (2024) developed KernelSHAP-IQ, a weighted least squares optimization method for efficiently computing Shapley interaction values. These methods reveal complex feature dependencies that single-feature attribution techniques miss, improving model transparency. Our method, LATIM, further extends these principles to SSMs, enabling fine-grained token-level interaction analysis specifically tailored for Mamba models.

Theoretical Insights into SSMs. Beyond interpretability, several studies have analyzed the internal mechanisms of SSMs. Vo et al. (2025) investigate the asymptotic behavior of token states, revealing conditions under which tokens either converge or diverge, affecting memory retention. Sieber et al. (2024) introduce a framework that unifies different sequence modeling paradigms, including SSMs, under a common mathematical representation. Meanwhile, Trockman et al. (2024) propose an initialization technique that improves Mamba's recall ability inspired by attention-like patterns.

Interpreting Mamba. Despite the growing adoption of Mamba, only a few works have explicitly addressed its interpretability. Ali et al. (2024) introduce Mamba-Attention and Mamba-Attribution, which approximate token interactions by extracting implicit attention patterns in Mamba-1. Similarly, MambaLRP (Jafari et al., 2024) applies Layer-wise Relevance Propagation to Mamba-1, ensuring stable attribution propagation. However, these approaches do not provide a direct decomposition of individual token contributions, leaving gaps in understanding how Mamba selectively processes information. LATIM bridges this gap by providing fine-grained, token-level interpretability for both Mamba-1 and Mamba-2. Additionally, we note that LATIM is adaptable and can be ap-

Activation	Error per Layer			AER	COMET
	0-16	16-32	32-48		
SiLU	0.21	0.45	0.57	0.47	83.4
SiLU + CP	0.21	0.43	0.54	**0.46**	**83.6**
ReLU	0.35	0.83	1.07	0.51	82.8
Identity	**0.00**	**0.00**	**0.00**	**0.46**	83.3

Table 5: Approximation error analysis with different activations for computing ϕ_i in Equation 18. CP indicates continued pretraining.

plied to other linear recurrent architectures, such as DeltaNet (Yang et al., 2024) and mLSTM (Beck et al., 2024), making it a valuable interpretability tool for long-context models.

6 Conclusion

Our experiments demonstrate that our fine-grained token-level decomposition approach significantly improves interpretability for Mamba models. Across copying, machine translation, and retrieval-based generation tasks, we show that our method, LATIM, particularly the ℓ_2 version, provides clearer insights into Mamba's selective processing mechanisms. For example, our findings suggest that Mamba's recall limitations in long-context tasks may stem from its sparse and decaying focus on relevant tokens. Moreover, our study confirms that while LATIM introduces a minimal approximation error, its exact counterpart eliminates this error entirely while maintaining interpretability and task performance. Together, these contributions improve our understanding of Mamba and open new directions for improving its reliability and effectiveness in real-world applications.

Limitations

We point out some limitations of the presented study. Our method, LATIM, relies on an approximation strategy to decompose token contributions due to the non-linearity introduced by the SiLU activation. Although our empirical analysis suggests that this approximation does not meaningfully impact interpretability quality, an exact decomposition requires model modifications, such as removing non-linearities, requiring re-training. Additionally, our evaluation focuses primarily on tasks like Copying and Machine Translation, where token interactions are well understood. In more complex tasks such as Retrieval-based Generation, assessing interpretability quality is harder, and further validation with human evaluations could provide a more robust assessment.

Furthermore, LATIM is specifically designed for Mamba-1 and Mamba-2, and while the principles behind it could easily be extended to other state space models or linear recurrent models, some additional modifications may be necessary. For example, architectures incorporating more complex gating mechanisms or hybrid attention-SSM layers might require adapted decomposition techniques for (i) deriving a sequence-mixer matrix $\boldsymbol{M}$ (as done in §2.4), and (ii) handling the non-nonlinearities present in the "SSM block" (as done in §3.3). Additionally, while LATIM helps visualize token interactions, its impact on improving model robustness and trustworthiness remains an open question.

Potential Risks

Although our token-level decomposition provides valuable insights, it may also be misused. An over-reliance on the generated token maps could lead users to assume these partial explanations capture all aspects of the model's reasoning. This false confidence may mask biases in the model or data, and encourage trust in outputs without adequate scrutiny, particularly in sensitive domains.

Additionally, exposing how Mamba selectively processes tokens could aid malicious actors in crafting targeted adversarial inputs. By identifying which tokens or positions most influence the model, adversaries could exploit these patterns to degrade performance or manipulate outputs. Such misuse risks undermining the reliability of Mamba-based systems, especially when high-stakes decisions rely on accurate and fair model predictions.

Acknowledgements

We thank André Martins, Pavlo Vasylenko, Giuseppe Attanasio and Saúl Santos for their helpful and constructive feedback. This work was supported by the Portuguese Recovery and Resilience Plan through project C645008882-00000055 (Center for ResponsibleAI), by the EU's Horizon Europe Research and Innovation Actions (UTTER, contract 101070631), by the project DECOLLAGE (ERC-2022-CoG 101088763), and by FCT/MECI through national funds and when applicable co-funded EU funds under UID/50008: Instituto de Telecomunicações.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online. Association for Computational Linguistics.

Ameen Ali, Itamar Zimerman, and Lior Wolf. 2024. The hidden attention of mamba models. *Preprint*, arXiv:2403.01590.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Re. 2024. Zoology: Measuring and improving recall in efficient language models. In *The Twelfth International Conference on Learning Representations*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *Preprint*, arXiv:1607.06450.

Jasmijn Bastings, Sebastian Ebert, Polina Zablotskaia, Anders Sandholm, and Katja Filippova. 2022. "will you find these shortcuts?" a protocol for evaluating the faithfulness of input salience methods for text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 976–991, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jasmijn Bastings and Katja Filippova. 2020. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 149–155, Online. Association for Computational Linguistics.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael K Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. 2024. xLSTM: Extended long short-term memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Aviv Bick, Kevin Li, Eric P. Xing, J Zico Kolter, and Albert Gu. 2024. Transformers to SSMs: Distilling quadratic knowledge to subquadratic models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Niehues Jan, Stüker Sebastian, Sudoh Katsuitho, Yoshino Koichiro, and Federmann Christian. 2017a. Overview of the iwslt 2017 evaluation campaign. In *Proceedings of the 14th International Workshop on Spoken Language Translation (IWSLT)*, pages 2–14.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichiro Yoshino, and Christian Federmann. 2017b. Overview of the IWSLT 2017 evaluation campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pages 2–14, Tokyo, Japan. International Workshop on Spoken Language Translation.

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

Nicola De Cao, Michael Sejr Schlichtkrull, Wilker Aziz, and Ivan Titov. 2020. How do decisions emerge across layers in neural models? interpretation with differentiable masking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3243–3255, Online. Association for Computational Linguistics.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Celine Lin, Jan Kautz, and Pavlo Molchanov. 2025. Hymba: A hybrid-head architecture for small language models. In *The Thirteenth International Conference on Learning Representations*.

Zi-Yi Dou and Graham Neubig. 2021. Word alignment by fine-tuning embeddings on parallel corpora. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2112–2128, Online. Association for Computational Linguistics.

Oliver Eberle, Jochen Büttner, Florian Kräutli, Klaus-Robert Müller, Matteo Valleriani, and Grégoire Montavon. 2022. Building and interpreting deep similarity models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1149–1161.

Paolo Fantozzi and Maurizio Naldi. 2024. The explainability of transformers: Current status and directions. *Computers*, 13(4):92.

Javier Ferrando, Gerard I. Gállego, and Marta R. Costa-jussà. 2022. Measuring the mixing of contextual information in the transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8698–8714, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Javier Ferrando, Gerard I. Gállego, Ioannis Tsiamas, and Marta R. Costa-jussà. 2023. Explaining how transformers use context to build predictions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5486–5513, Toronto, Canada. Association for Computational Linguistics.

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R Costa-jussà. 2024. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*.

Marina Fomicheva, Piyawat Lertvittayakumjorn, Wei Zhao, Steffen Eger, and Yang Gao. 2021. The

Eval4NLP shared task on explainable quality estimation: Overview and results. In *Proceedings of the 2nd Workshop on Evaluation and Comparison of NLP Systems*, pages 165–178, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fabian Fumagalli, Maximilian Muschalik, Patrick Kolpaczki, Eyke Hüllermeier, and Barbara Hammer. 2024. Kernelshap-iq: weighted least square optimization for shapley interactions. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. 2020. Hippo: Recurrent memory with optimal polynomial projections. In *Advances in Neural Information Processing Systems*, volume 33, pages 1474–1487. Curran Associates, Inc.

Albert Gu, Karan Goel, and Christopher Re. 2022. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *Preprint*, arXiv:2404.06395.

Farnoush Rezaei Jafari, Grégoire Montavon, Klaus Robert Muller, and Oliver Eberle. 2024. MambaLRP: Explaining selective state space sequence models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. 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 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Joseph D. Janizek, Pascal Sturmfels, and Su-In Lee. 2021. Explaining explanations: Axiomatic feature interactions for deep networks. *Journal of Machine Learning Research*, 22(104):1–54.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and eran malach. 2024. Repeat after me: Transformers are better than state space models at copying. In *Forty-first International Conference on Machine Learning*.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2021. Incorporating Residual and Normalization Layers into Analysis of Masked Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4547–4568, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Barak Lenz, Opher Lieber, Alan Arazi, Amir Bergman, Avshalom Manevich, Barak Peleg, Ben Aviram, Chen Almagor, Clara Fridman, Dan Padnos, Daniel Gissin, Daniel Jannai, Dor Muhlgay, Dor Zimberg, Edden M. Gerber, Elad Dolev, Eran Krakovsky, Erez Safahi, Erez Schwartz, Gal Cohen, Gal Shachaf, Haim Rozenblum, Hofit Bata, Ido Blass, Inbal Magar, Itay Dalmedigos, Jhonathan Osin, Julie Fadlon, Maria Rozman, Matan Danos, Michael Gokhman, Mor Zusman, Naama Gidron, Nir Ratner, Noam Gat, Noam Rozen, Oded Fried, Ohad Leshno, Omer Antverg, Omri Abend, Or Dagan, Orit Cohavi, Raz Alon, Ro'i Belson, Roi Cohen, Rom Gilad, Roman Glozman, Shahar Lev, Shai Shalev-Shwartz, Shaked Haim Meirom, Tal Delbari, Tal Ness, Tomer Asida, Tom Ben Gal, Tom Braude, Uriya Pumerantz, Josh Cohen, Yonatan Belinkov, Yuval Globerson, Yuval Peleg Levy, and Yoav Shoham. 2025. Jamba: Hybrid transformer-mamba language models. In *The Thirteenth International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Hosein Mohebbi, Jaap Jumelet, Michael Hanna, Afra Alishahi, and Willem Zuidema. 2024. Transformer-specific interpretability. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, pages 21–26, St. Julian's, Malta. Association for Computational Linguistics.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Hugo Pitorro, Pavlo Vasylenko, Marcos Treviso, and André Martins. 2024. How effective are state space models for machine translation? In *Proceedings of the Ninth Conference on Machine Translation*, pages 1107–1124, Miami, Florida, USA. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Jerome Sieber, Carmen Amo Alonso, Alexandre Didier, Melanie Zeilinger, and Antonio Orvieto. 2024. Understanding the differences in foundation models: Attention, state space models, and recurrent neural networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Marcos Treviso and André F. T. Martins. 2020. The explanation game: Towards prediction explainability through sparse communication. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 107–118, Online. Association for Computational Linguistics.

Asher Trockman, Hrayr Harutyunyan, J. Zico Kolter, Sanjiv Kumar, and Srinadh Bhojanapalli. 2024. Mimetic initialization helps state space models learn to recall. *Preprint*, arXiv:2410.11135.

Alexandros Vasileiou and Oliver Eberle. 2024. Explaining text similarity in transformer models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7859–7873, Mexico City, Mexico. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

David Vilar, Maja Popovic, and Hermann Ney. 2006. AER: do we need to "improve" our alignments? In *Proceedings of the Third International Workshop on Spoken Language Translation: Papers*, Kyoto, Japan.

Thieu Vo, Duy-Tung Pham, Xin T. Tong, and Tan Minh Nguyen. 2025. Demystifying the token dynamics of deep selective state space models. In *The Thirteenth International Conference on Learning Representations*.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Yuxin Wu and Kaiming He. 2018. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Rui Xu, Shu Yang, Yihui Wang, Bo Du, and Hao Chen. 2024. A survey on vision mamba: Models, applications and challenges. *arXiv preprint arXiv:2404.18861v1*.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. 2024. Parallelizing linear transformers with the delta rule over sequence length. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

A Hidden Attention Derivation in Mamba

This appendix provides a detailed derivation of the hidden-attention matrix $\boldsymbol{M}$ in both Mamba-1 and Mamba-2, showing how their element-wise recurrences can be written in the form $\boldsymbol{\Upsilon} = \boldsymbol{M}\,\boldsymbol{X}$.

A.1 Mamba-1 Derivation

Recall the Mamba-1 recurrence (ignoring skip connections) for each time step $i \geq 1$:

$$\begin{aligned} \boldsymbol{H}_i &= \boldsymbol{A}_i \odot \boldsymbol{H}_{i-1} + \boldsymbol{B}_i \odot \boldsymbol{X}_i \quad && \in \mathbb{R}^{R\times D}, \\ \boldsymbol{v}_i &= \boldsymbol{H}_i^\top \boldsymbol{c}_i && \in \mathbb{R}^{D}, \end{aligned}$$

where $\boldsymbol{X}_i = \mathbf{1}_r \boldsymbol{x}_i^\top \in \mathbb{R}^{R\times D}$ is an R-sized stack of the input, $\boldsymbol{A}_i \in \mathbb{R}^{R\times D}$ represents D diagonal matrices of size $R \times R$, $\boldsymbol{B}_i \in \mathbb{R}^{R\times D}$, $\boldsymbol{c}_i \in \mathbb{R}^{R}$, and $\odot$ is the Hadamard product. Setting $\boldsymbol{H}_0 = \mathbf{0}$, we can unroll the recurrence to see how past tokens contribute:

$$\boldsymbol{H}_1 = \boldsymbol{A}_1 \odot \mathbf{0} + \boldsymbol{B}_1 \odot \boldsymbol{X}_1.$$

$$\begin{aligned} \boldsymbol{H}_2 &= \boldsymbol{A}_2 \odot \boldsymbol{H}_1 + \boldsymbol{B}_2 \odot \boldsymbol{X}_2 \\ &= \boldsymbol{A}_2 \odot (\boldsymbol{A}_1 \odot \mathbf{0} + \boldsymbol{B}_1 \odot \boldsymbol{X}_1) + \boldsymbol{B}_2 \odot \boldsymbol{X}_2 \\ &= \boldsymbol{A}_2 \odot \boldsymbol{B}_1 \odot \boldsymbol{X}_1 + \boldsymbol{B}_2 \odot \boldsymbol{X}_2. \end{aligned}$$

$$\begin{aligned} \boldsymbol{H}_3 &= \boldsymbol{A}_3 \odot \boldsymbol{H}_2 + \boldsymbol{B}_3 \odot \boldsymbol{X}_3 \\ &= \boldsymbol{A}_3 \odot [\boldsymbol{A}_2 \odot \boldsymbol{B}_1 \odot \boldsymbol{X}_1 + \boldsymbol{B}_2 \odot \boldsymbol{X}_2] \\ &\quad + \boldsymbol{B}_3 \odot \boldsymbol{X}_3 \\ &= \boldsymbol{A}_3 \odot \boldsymbol{A}_2 \odot \boldsymbol{B}_1 \odot \boldsymbol{X}_1 \\ &\quad + \boldsymbol{A}_3 \odot \boldsymbol{B}_2 \odot \boldsymbol{X}_2 + \boldsymbol{B}_3 \odot \boldsymbol{X}_3. \end{aligned}$$

Hence, in general for any i, we have:

$$\begin{aligned} \boldsymbol{H}_i &= \sum_{j=1}^{i} \left(\bigodot_{k=j+1}^{i} \boldsymbol{A}_k \right) \odot \boldsymbol{B}_j \odot \boldsymbol{X}_j \quad && \in \mathbb{R}^{R\times D}, \\ \boldsymbol{v}_i &= \boldsymbol{H}_i^\top \boldsymbol{c}_i && \in \mathbb{R}^{D}, \end{aligned}$$

where we write $\bigodot$ to indicate an element-wise product over the indices k.

Block-matrix expression. To capture this in matrix form, observe that each coordinate of $\boldsymbol{X}_j$ gets multiplied by a chain of element-wise factors $\boldsymbol{A}_k$ and $\boldsymbol{B}_j$, then finally projected by $\boldsymbol{c}_i$. Aggregating these dimension-wise scalars into a diagonal matrix $\boldsymbol{M}_{i,j} \in \mathbb{R}^{D\times D}$ yields

$$\boldsymbol{M}_{i,j} = \mathrm{Diag}\left(\left[\left(\bigodot_{k=j+1}^{i} \boldsymbol{A}_k \right) \odot \boldsymbol{B}_j \right]^\top \boldsymbol{c}_i \right),$$

for all $j \leq i$, and $\boldsymbol{M}_{i,j} = \mathbf{0}$ otherwise. Stacking these $\boldsymbol{M}_{i,j}$ blocks into a 4D tensor $\boldsymbol{M} \in \mathbb{R}^{N\times N\times D\times D}$ gives us

$$\boldsymbol{\Upsilon} = \boldsymbol{M}\,\boldsymbol{X}, \tag{26}$$

once we interpret $\boldsymbol{M}$ as an $N \times N$ grid of $D \times D$ blocks and flatten $\boldsymbol{X} \in \mathbb{R}^{N\times D}$ to a length-(ND) vector, as explained below in §A.3.

A.2 Mamba-2 Derivation

Mamba-2 uses a similar idea but modifies $\boldsymbol{A}_i$ into a *diagonal matrix* of size $R \times R$, rather than an element-wise parameter array. Formally,

$$\begin{aligned} \boldsymbol{H}_i &= \boldsymbol{A}_i \boldsymbol{H}_{i-1} + \boldsymbol{B}_i \odot \boldsymbol{X}_i \quad && \in \mathbb{R}^{R\times D}, \\ \boldsymbol{v}_i &= \boldsymbol{H}_i^\top \boldsymbol{c}_i && \in \mathbb{R}^{D}, \end{aligned}$$

where $\boldsymbol{A}_i = a_i\, \boldsymbol{I}_{R\times R}$. Unrolling similarly, we get

$$\begin{aligned} \boldsymbol{H}_t &= \sum_{j=1}^{t} \left(\prod_{k=j+1}^{t} \boldsymbol{A}_k \right) \odot \boldsymbol{B}_j \odot \boldsymbol{X}_j \quad && \in \mathbb{R}^{R\times D}, \\ \boldsymbol{v}_t &= \boldsymbol{H}_t^\top \boldsymbol{c}_t && \in \mathbb{R}^{D}. \end{aligned}$$

Since each $\boldsymbol{A}_k$ is a diagonal matrix, the product $\prod_{k=j+1}^{i} \boldsymbol{A}_k$ remains diagonal. Aggregating the resulting dimension-wise multipliers again forms $\boldsymbol{M}_{i,j} \in \mathbb{R}^{D\times D}$, leading to

$$\boldsymbol{M}_{i,j} = \mathrm{Diag}\left(\left[\left(\prod_{k=j+1}^{i} \boldsymbol{A}_k \right) \odot \boldsymbol{B}_j \right]^\top \boldsymbol{c}_i \right),$$

for all $j \leq i$, and $\boldsymbol{M}_{i,j} = \mathbf{0}$ otherwise. The shape-flattening for $\boldsymbol{M}$ and $\boldsymbol{X}$ then follows the same block-matrix logic as in Mamba-1.

A.3 Block-Matrix Implementation

Define the overall 4D tensor $\boldsymbol{M} \in \mathbb{R}^{N\times N\times D\times D}$ by gathering the blocks $\boldsymbol{M}_{i,j}$ from above. In matrix form, we can treat $\boldsymbol{M}$ as an $N \times N$ grid of $D \times D$ blocks, thus flattening to $\boldsymbol{M} \in \mathbb{R}^{(ND)\times(ND)}$. Simultaneously, reshape $\boldsymbol{X} \in \mathbb{R}^{N\times D}$ into a vector of length (ND) by stacking each token row. Then the usual matrix–vector product recovers the unrolled recurrence:

$$\boldsymbol{\Upsilon} = \boldsymbol{M}\boldsymbol{X} \Leftrightarrow \boldsymbol{v}_i = \sum_{j=1}^{i} \boldsymbol{M}_{i,j}\, \boldsymbol{x}_j.$$

Concretely, the i^{th} block row of $\boldsymbol{M}$ multiplies the token embeddings $\{\boldsymbol{x}_j\}_{j=1}^{N}$, and the result is then reshaped back to produce an $N \times D$ matrix, whose i^{th} row is precisely $\boldsymbol{v}_i^\top$.

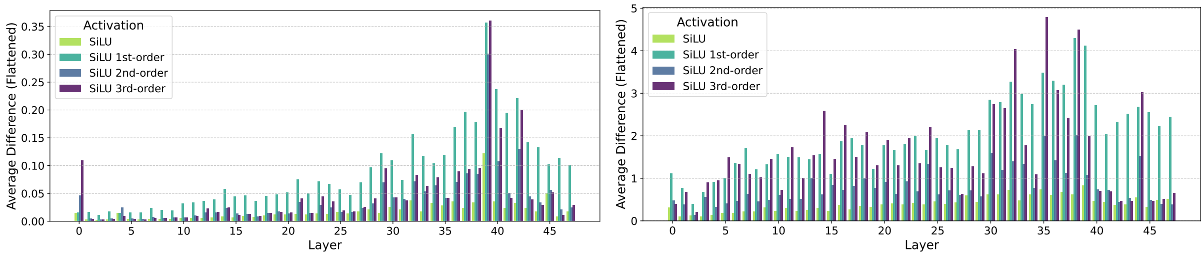

Figure 4: Error amounting to the average difference between the regular Mamba-1 (left) and Mamba-2 (right) layer output and the interpretable version with different approximations f in Equation 19.

B Experimental Details

B.1 Copying

We use 8-layer Mamba 1 and 2 models with 512 as the hidden size and 32 as the vocabulary size, the state dimension is set to 16 and 128 for Mamba 1 and 2, respectively. Only layer 4 is initialized as per Trockman et al. (2024), with their optimal configuration (which differs from Mamba 1 to 2). Optimization: AdamW (Loshchilov and Hutter, 2019) optimizer with the inverse square root (Vaswani et al., 2017) learning rate scheduler (500 warmup steps, 5 000 total steps, 256 samples per batch) and a learning rate of $7e-4$. No dropout or gradient clipping was used. The copying dataset was generated as per (Jelassi et al., 2024) and contains 5 000 training samples and 128 evaluation samples.

B.2 Machine Translation

All model dimensions are coupled to their officially released checkpoints. Optimization: AdamW (Loshchilov and Hutter, 2019) optimizer with a cosine learning rate scheduler (2 000 warmup steps, 18 000 steps, 64 samples per batch) and a learning rate of $7e-4$. Dropout (Srivastava et al., 2014) rate was set to 0.3 and no gradient clipping was used. The IWSLT17 (Cettolo et al., 2017b) dataset contains 232 825 training samples, 890 validation samples and 8 597 samples for both the DE↔EN and FR↔EN versions.

B.3 Approximation Error

All model dimensions are coupled to their officially released checkpoints when performing continued language pretraining. Optimization: AdamW (Loshchilov and Hutter, 2019) optimizer with a WSD (Hu et al., 2024) learning rate scheduler (2 000 warmup steps, 27 900 stable steps, 3 100 decay steps, 32k tokens per batch) and a learning rate of $5e-5$. We used gradient clipping set to 5.0 and no dropout. Moreover, we employed an α parameter in order to smoothly interpolate between the old (SiLU) and the new activations (ReLU or identity). The value of α followed a power law during training: $\min(1, \texttt{current\_step}/(\texttt{total\_steps} - \texttt{decay\_steps}))^2$. Note that the learning rate decay period coincides with the phase where the model relies only on the new activation.

B.4 Computational Details

All experiments involving LATIM were carried on Nvidia RTX A6000 GPUs with 48GB VRAM.

C Extended Approximation Error

Following §4.4, we include additional data which details how the decomposition error changes with different SiLU approximations f on each layer. This experiment has been conducted over the GoldAlign (Vilar et al., 2006) dataset. The results can be seen in Figure 4. Overall, casting f as SiLU leads to the lowest approximation errors across all models and layers.

D Additional Experiments

D.1 Copying

In addition to the Mamba-2 visualizations in §4.1, we further include Mamba-1-based versions in Figure 5. These include a comparison with MambaLRP which performs especially poorly for this experiment as previously observed in Table 2. Moreover, in Figure 6 we show a filtered version of these plots with just the source→copy interactions (left-bottom block). We highlight how models learn a pattern centered around the diagonal. As per this argument, Table 2 relies on the three main diagonals as its gold label.

D.2 Machine Translation

In addition to the Mamba-1 visualizations in §4.2, we further include Mamba-2-based versions in Figure 7.

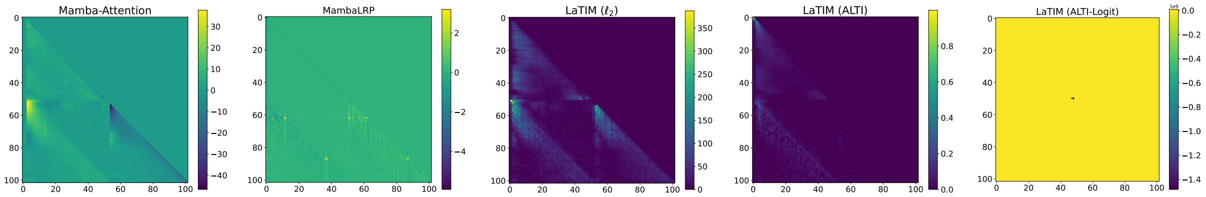

Figure 5: Heatmaps produced by different interpretable approaches for Mamba-1. The interaction between source and copied tokens (along the diagonal line) becomes clearer with LaTIM.

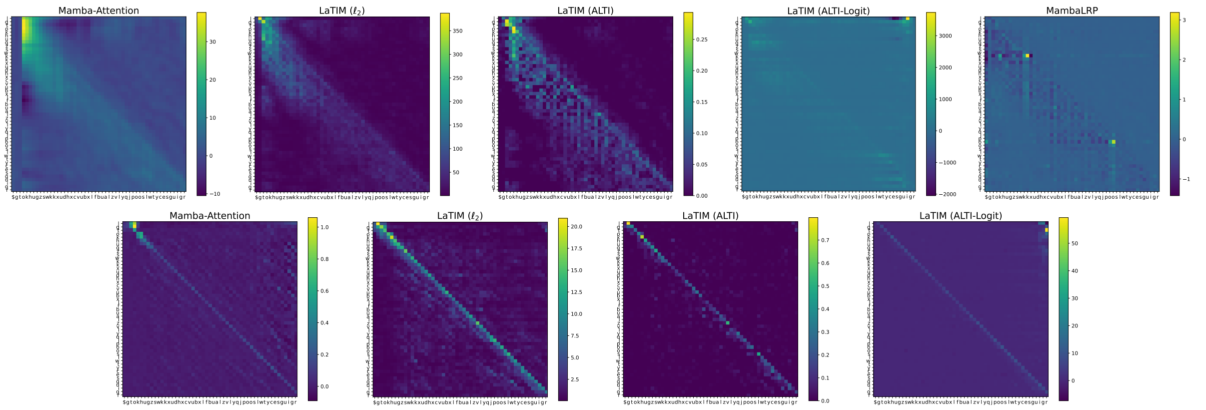

Figure 6: Heatmaps produced by the different interpretability methods for Mamba-1 (top) and Mamba-2 (bottom) on a copying sample focusing only on the source→copy interaction. Note how both models learned to focus over a off-diagonal pattern instead of a direct token-copy map.

Size	**1024**			**2048**		
	$k=1$	$k=2$	$k=4$	$k=1$	$k=2$	$k=4$
130M	99.8	58.2	28.1	99.7	57.3	30.8
370M	100.0	57.6	33.1	98.0	55.1	34.1
780M	99.8	68.1	60.2	84.5	59.3	51.4
1.4B	99.3	63.2	39.6	99.7	60.8	38.9

Table 6: Mamba-2 accuracy (%) in the Passkey Retrieval task at recovering the correct output. We vary the model size, sequence length (1024 and 2048) and the number of keys $k \in \{1, 2, 4\}$. Computed over 1000 samples.

D.3 Retrieval-based Generation

Passkey Retrieval. We compute accuracy statistics in the passkey retrieval task for Mamba-2 370M for each variation (1, 2 and 4 passkeys). We observe that the model has a heavy bias towards the first passkey that appears in context as the average accuracy decreases as more keys get introduced (Table 6). To strengthen our argument, accuracy heavily depends on whether the desired passkey is the first that appears (Table 4).

Frequent Word Extraction. In Figure 9 (left) we plot Mamba-2's focus over the context tokens in the Frequent Word Extraction task when we only consider the "and uqbc" (underlined) tokens. As we can see, only some words get attended to, making it difficult for the model to track word frequencies. To strengthen this effect, the average attention per word instance decreases heavily. For example, the word "fdcvcu" occurs 68 times and its first few occurrences have an average attention score across layers substantially higher than the remainder.

E AI Assistants

We used Cursor during development, and ChatGPT during paper writing for grammar correction.

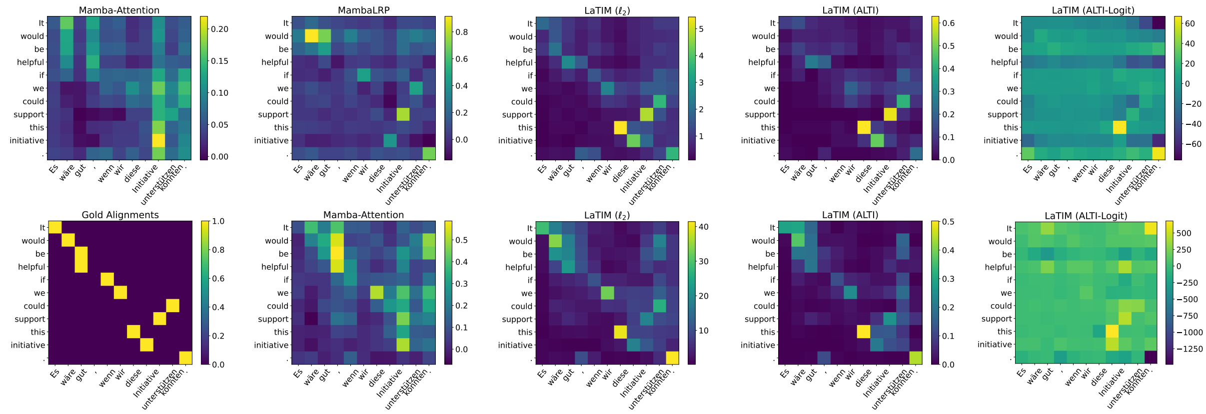

Figure 7: Heatmaps produced by the different interpretability methods for Mamba-1 (top) and Mamba-2 (bottom) fine-tuned on DE→EN data.

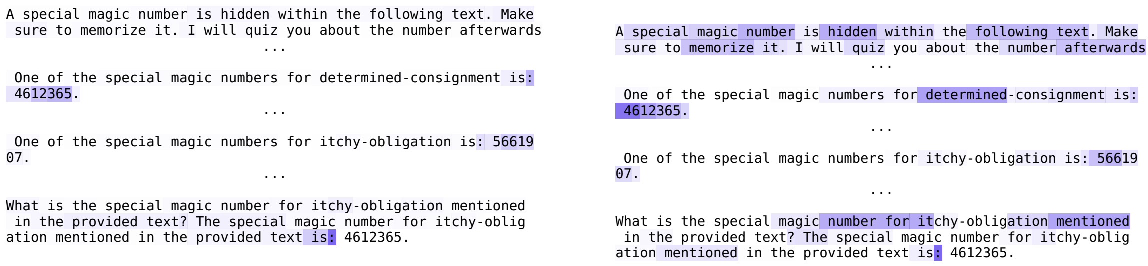

Figure 8: Attention plots obtained by LATIM (ℓ_2) (left) and MambaAttention (right) on a Passkey Retrieval sample, showing that MambaAttention focuses on several misleading tokens, such as "determined", "number for", and "mentioned". In contrast, LATIM (ℓ_2) focuses only on meaningful strings, like "4612365" (the predicted key) and "5661907" (the correct key).

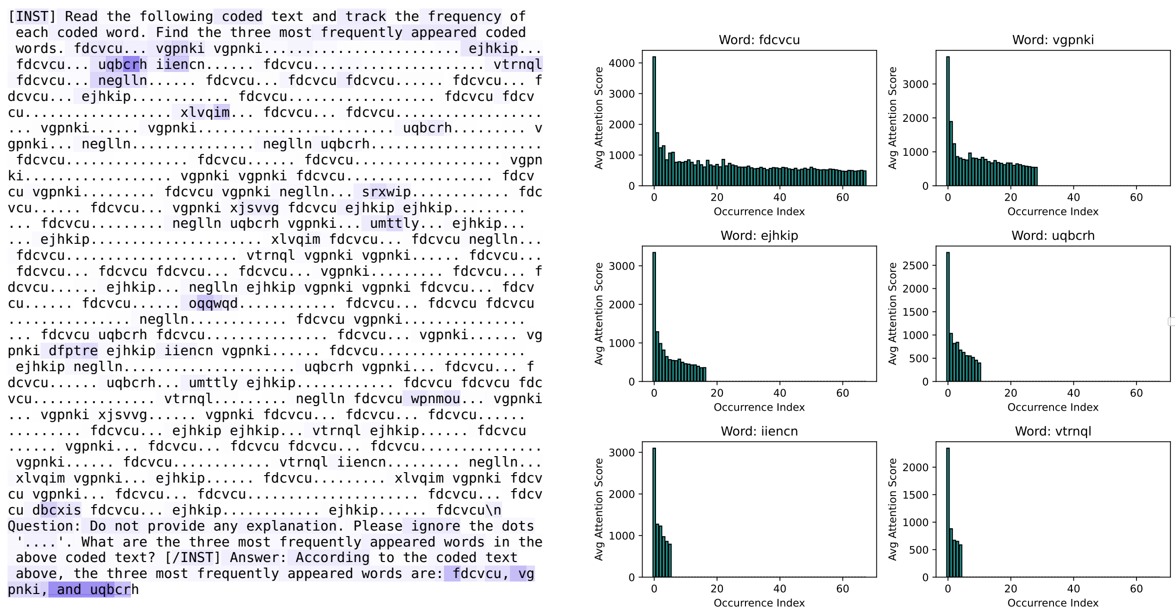

Figure 9: Left: Attention map obtained by LATIM (ℓ_2) on the Frequent Word Extraction task, showing that the model is focusing on the incorrectly generated "and uqbcr" token range (Mamba-2 370M layer 23). Right: Average attention score per word instance, showing that the model's focus reduces heavily after the first few word occurrences.