HOW LINEARLY ASSOCIATIVE ARE MEMORIES IN LARGE LANGUAGE MODELS?

Akshat Gupta, Nehal Sindhu, Gopala Anumanchipalli
UC Berkeley
{akshat.gupta, nehalsindhu, gopala}@berkeley.edu

ABSTRACT

Large Language Models (LLMs) exhibit remarkable capacities to store and retrieve factual knowledge, yet the precise mechanisms by which they encode and recall this information remain under debate. Two main frameworks have been proposed to explain memory storage within transformer feed-forward layers: (1) a key-value memory view, and (2) linear associative memories view. In this paper, we investigate the extent to which the second MLP matrix in LLMs behaves as a linear associative memory (LAM). By measuring pairwise angles between input activation vectors that represent key-vectors in the LAM model, we find that the second MLP matrix exhibits relatively higher orthogonality and minimal cross-talk, supporting the LAM interpretation for generic retrieval. However, we also discover that subject-token representations used in factual recall are significantly less orthogonal, indicating greater interference and entanglement. This implies that editing factual "memories" within these matrices may trigger unintended side effects in other related knowledge. Our results highlight both the promise and the pitfalls of viewing feed-forward layers as linear associative memories, underscoring the need for careful strategies when modifying factual representations in LLMs.

1 INTRODUCTION

Transformer language models have shown a remarkable ability to recall factual knowledge (Carlini et al., 2022). Prior work has proposed two main frameworks to explain memory storage and recall mechanisms in large language models: (1) feed-forward layers acting as key-value memories (Geva et al., 2020), where factual information is stored as key-value associations in the multi-layer perceptron (MLP) module of the transformer, and (2) linear associative memories (Bau et al., 2020; Meng et al., 2022a), which model factual recall as the retrieval of stored associations via a linear mapping.[1]

In the key-value memories framework (Geva et al., 2020), the MLP component within the transformer's feed-forward network (FFN) is hypothesized to function as a key-value store. Similar to a database retrieval, the elements of the activation vector act as keys that select and retrieve stored vectors, also known as values. Specifically, in this framework, the second weight matrix of the MLP module which follows the non-linearity, is responsible for storing the value vectors. More concretely, the columns of this matrix correspond to the stored values. This means that for GPT-2 XL (Radford et al., 2019), which has a hidden dimensionality of 1600 and contains 6400 columns in the second MLP matrix, there are a total of 6400 vectors or values present in the vector store. The keys are the individual elements of the vector activations produced by the first MLP matrix. Since the first MLP matrix produces an intermediate vector of dimensionality 6400 for GPT-2 XL, there are 6400 keys, each selecting whether to include a column from the second MLP matrix. Note that in this view, a key is a scalar which retrieves a column of the second MLP matrix, and the final output is a linear combination of columns of the second MLP matrix where the keys act as the scalar coefficients in the combination. When the key is zero, the column vector is not a part of the linear combination, and hence, can be understood as the value vector not being retrieved from the vector

[1]Note that while both these frameworks involve a key-value retrieval, we will refer to the framework proposed in Geva et al. (2020) specifically as key-value memories in this paper.

store. Thus, this view presents a very procedural view of understanding the matrix-vector products happening within the FFN layer.

The linear associative memory view of fact recall (Bau et al., 2020; Meng et al., 2022a) differs significantly from the procedural view of key-value memories. In this view, the second MLP matrix in the FFN layer is hypothesized as a store of a very large number of value vectors which are referred to as values. When a vector is multiplied by this matrix, it is supposed to retrieve exactly one of these very large number of value vectors. The retrieval happens because each of these value vectors is stored using nearly orthogonal key-vectors. Apart from being a store of a large number of vectors which is much greater than the number of columns in the second MLP matrix, another difference between the linear associative view of memories is that the key that retrieves one of these vectors is also a vector.

An important aspect of an ideal linear associative memory (LAM) is the hypothesized orthogonality of key-vectors used for retrieval. An ideal LAM has nearly orthogonal key-vectors. This orthogonality property ensures perfect recall of value vectors and isolates each stored memory with the other. If the orthogonality assumption is violated, memory recall becomes approximate and may overlap with other memories. Understanding this crosstalk (lack of orthogonality) between memories is essential for tasks like knowledge editing using methods like ROME (Meng et al., 2022a; Gupta et al., 2024a) or MEMIT (Meng et al., 2022b) which use the linear associative memory model for editing (Gupta et al., 2024c). A large crosstalk means that memories stored inside the MLP matrices have significant overlaps, and editing one fact is likely to have large ripple effects (Cohen et al., 2023). While this has been shown in prior work, in this paper we study the amount of crosstalk within the stored memories under the LAM model and find the reason behind such ripple effects.

Specifically, we calculate the crosstalk between stored memories in large language models and analyze how these memories deviate from ideal linear associative memories. We first show that the second MLP-matrix in the FFN module has a larger amount of orthogonality of stored keys and hence reduced crosstalk, showing that potential correctness of modeling the second MLP matrix in the FFN as a LAM. We do this by finding the angle between activation vectors of random tokens from different contexts. This is done for three representative LLMs - GPT-2 XL (Radford et al., 2019), Pythia 6.9B (Biderman et al., 2023) and Llama2-7B (Touvron et al., 2023). We also show that the second MLP matrices in Llama2-7B are the closest to ideal LAMs with the least amount of crosstalk. Next, we analyze the orthogonality of key-vectors specifically in the case of fact recall, where we find the angle between the activation vectors for the last subject token in a query. We find that the amount of orthogonality goes down and see that there is a much larger crosstalk between memories stored for fact recall. While our results show promise in modeling second MLP matrices as linear associative memories, factual recall exhibits significant crosstalk, and editing facts in these matrices may cause unintended consequences, as shown in prior work (Cohen et al., 2023; Gupta et al., 2024b).

2 Background : Linear Associative Memories

Linear associative memories (Kohonen, 1972) can be formulated as a mapping from an input representation x to an output representation y, governed by a learned weight matrix W, which functions as a memory store:

$$y = Wx \tag{1}$$

where $x \in \mathbb{R}^{d_k}$ represents the input to the matrix and also functions as the key for memory retrieval, $W \in \mathbb{R}^{d_k \times d_v}$ is the learned memory store, and $y \in \mathbb{R}^{d_v}$ is the output or the retrieved memory. $W \in \mathbb{R}^{d_k \times d_v}$ functions as a linear associative memory if it can be approximately represented as the following summation of outer products:

$$W \approx \sum_{i=1}^{N} v_i k_i^T \tag{2}$$

where k_i are the key-vectors, v_i are the values (stored representations), and N is the number of stored associations. Here, $N >> Rank(W)$, which means that the number of associations stored in the matrix is much larger than the rank of the matrix. A crucial part of an ideal linear associative memory is the assumption of approximate orthogonality of the key-vectors. This orthogonality enables retrieval of a specific memory m when the corresponding key is sent to the memory store W. If the key-vectors are approximately orthogonal, a memory v_m can be retrieved perfectly by a simple matrix-vector product with the key k_m as shown:

$$Wk_m \approx \sum_{i=1}^{N} v_i k_i^T k_m = \sum_{i=1}^{N} v_i \delta_{im} = v_m \tag{3}$$

This ideal structure of W enables the retrieval of v_i when x aligns with k_i. However, in a non-ideal case, stored representations may exhibit interference, leading to crosstalk between associations. Crosstalk quantifies the degree of interference among stored representations. Ideally, if factual associations were independent, they would satisfy:

$$k_i^T k_j = 0 \quad \forall\, i \neq j \tag{4}$$

However, due to overlapping memory storage, factual associations are not strictly orthogonal. We quantify this interference by computing the pairwise angle between stored vectors:

$$\theta_{i,j} = \cos^{-1}\left(\frac{k_i^T k_j}{\|k_i\|\|k_j\|}\right) \tag{5}$$

where $\theta_{i,j}$ is the angle between the key-vectors k_i and k_j of the stored representations. High crosstalk occurs when $\theta_{i,j}$ deviates significantly from 90 degrees, indicating a lack of independence between memory units. In this paper, we quantify crosstalk by finding the average angle between pairs of key-vectors.

3 METHODOLOGY

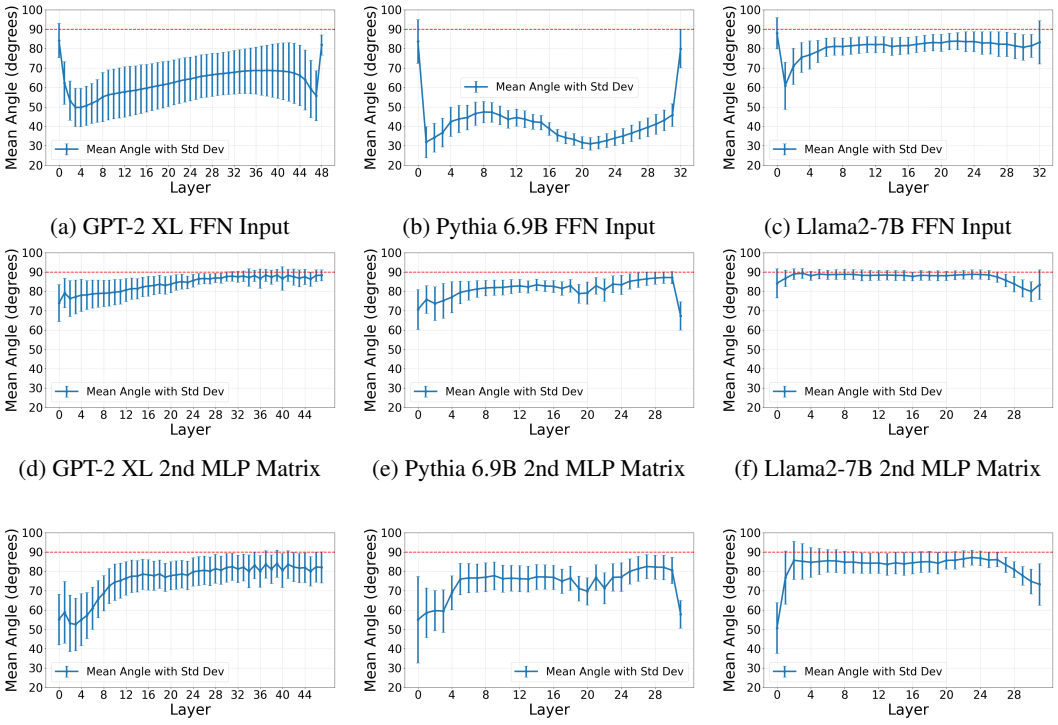
We study the appropriateness of matrices in the FFN modules functioning as ideal linear associative memories. We do this for three different models: GPT-2 XL (1.5B parameters) Radford et al. (2019), Llama2 (7B) Touvron et al. (2023), and the Pythia model family (70M-6.9B) Biderman et al. (2023). We input multiple sentences into the models and extract their activation vectors at the input of the second MLP matrix and at the input of the FFN module[2]. We analyze how well the two MLP matrices within the FFN module represent ideal linear associative memories, that is, have approximately orthogonal key-vectors that are input to the matrices. To do this, we calculate the pairwise angles between intermediate activations at the same layer.

To study the pairwise angles between key-vectors, we use two datasets. First, we use 10,000 paragraphs from the Wikipedia corpus which are on average 150 tokens long and send them as input to the model. We choose one token at random from the input paragraph and store its activations for all layers being analyzed. This gives us 10,000 activation vectors at each layer. We then compute the pairwise angle between 100k activation pairs per layer from the above collection. Second, we use the CounterFact dataset Meng et al. (2022a) to specifically study how retrieval of value vectors happens when factual questions are asked. Sentences in the CounterFact dataset are present in the form of subject-relation-object triplets and we store the activation vector of the last subject token. We then perform pairwise comparisons between subject tokens from different sentences, sample 100k such pairs, and find the angle between them. These subject token representations are used for knowledge editing (Meng et al., 2022a;b) and we measure orthogonality between them to study their effectiveness in recalling individual memories. If these subject tokens recall multiple memories, then knowledge editing is not just editing one memory but multiple memories that have non-zero interference.

[2]Input to the FFN module is the same as the input to the first MLP matrix within the FFN

4 Results and Discussion

4.1 Comparison of Input Layer and Second MLP Matrix



(a) GPT-2 XL FFN Input (b) Pythia 6.9B FFN Input (c) Llama2-7B FFN Input

(d) GPT-2 XL 2nd MLP Matrix (e) Pythia 6.9B 2nd MLP Matrix (f) Llama2-7B 2nd MLP Matrix

(g) GPT-2 XL 2nd MLP Matrix - CounterFact (h) Pythia 6.9B 2nd MLP Matrix - CounterFact (i) Llama2-7B 2nd MLP Matrix - CounterFact

Figure 1: Comparison of pairwise angles for input layer and second MLP matrix for the Wikipedia Dataset in the first two rows and CounterFact dataset in the third row.

We first study the behavior of the two matrices in the FFN module as ideal linear associative memories. The experiment aims to quantify the average angle between pairs of activation vectors at the input of both matrices in an FFN module. We first perform this analysis for the Wikipedia dataset, where the activations analyzed belong to randomly selected tokens from different contexts. The results for this are shown in Figure 1.

The first row in Figure 1 shows the average angle between pairs for representations at the input of the first matrix in the FFN layers. We see a large deviation from the ideal linear associative memory behavior, where the average angle deviates from 90 degrees by large amounts for all models. The second row of Figure 1 shows the average angle between pairs of activations that are input to the second MLP matrix. We immediately see that the average angle between the activations input to the second MLP-matrix gets closer to 90 degrees compared to the first MLP matrix. This shows that the second MLP matrix is functioning much closer to a linear associative memory in LLMs. However, it is still not an ideal linear associative memory as the angles have a much larger range (80-90 degrees) and hence still have a non-negligible amount of crosstalk. The behavior also depends on the models, where the Llama2-7B models functions the closest to ideal linear associative memories with minimal crosstalk compared to the other models. We also see a trend of increasing amount of orthogonality of key-vectors as we go into deeper layers of all models. This shows that memories are more likely to be independent in later layers and editing might be more localized in those layers.

Note that we select token activations from different contexts to calculate pairwise angles. When activations from the same context or input sentence are selected, we observe high similarity and consequently lower orthogonality. This is natural to observe due to a shared context, which leads to

similar representations as activations of future tokens are a linear combination of prior tokens due to the attention mechanism.

4.2 Analysis for the CounterFact Dataset

In this experiment, we analyze the angle between the subject token activations of different fact recall questions. The research question here is to study whether the key-vectors that query the answers of different factual questions are more or less likely to be orthogonal than activations for two tokens selected at random for two different contexts. If they are orthogonal from each other, this would mean that editing memory corresponding to one key vector will not affect the recall of other memories. If there is a large crosstalk, this means that editing one memory is likely to have a ripple effect and possibly unintended consequences.

The results for this experiment can be seen in the third row of Figure 1. We see that for all the models, the amount of orthogonality goes down comparing to activations of random Wikipedia tokens. This means that the amount of crosstalk for factual recall tasks is larger compared to activations of two tokens selected at random from different contexts. This deviation from orthogonality is largest for GPT-2 XL in the early layers and Pythia. Llama2-7B still maintains the maximum orthogonality, especially between layers 3-24. Although the angle between the representations is still not 90 degrees and likely to cause some crosstalk.

5 Conclusion and Discussion

In this paper, we demonstrate the appropriateness of modeling memory recall in LLMs as linear associative memories. We show that as postulated in prior work, the second MLP matrix in the FFN module can be modelled using ideal linear associative memories. We also show that while these matrices have minimum crosstalk for randomly selected tokens, key-vectors responsible for factual recall have a much larger dependence on each other. This means editing facts are likely to have larger ripple effects compared to editing a random token. In practice, this would mean that editing facts may not be as localized as is hypothesized in prior work Meng et al. (2022a). This may or may not be desirable since in some instances we want specific consequences of a fact to also get edited, while in some cases we don't.

References

David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pp. 351–369. Springer, 2020.

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

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *arXiv preprint arXiv:2307.12976*, 2023.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

Akshat Gupta, Sidharth Baskaran, and Gopala Anumanchipalli. Rebuilding rome: Resolving model collapse during sequential model editing. *arXiv preprint arXiv:2403.07175*, 2024a.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. *arXiv preprint arXiv:2401.07453*, 2024b.

Akshat Gupta, Dev Sajnani, and Gopala Anumanchipalli. A unified framework for model editing. *arXiv preprint arXiv:2403.14236*, 2024c.

Teuvo Kohonen. Correlation matrix memories. *IEEE transactions on computers*, 100(4):353–359, 1972.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022a.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022b.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models, 2023. *URL https://arxiv. org/abs/2307.09288*, 2023.

A REVIEW: KEY-VALUE MEMORIES

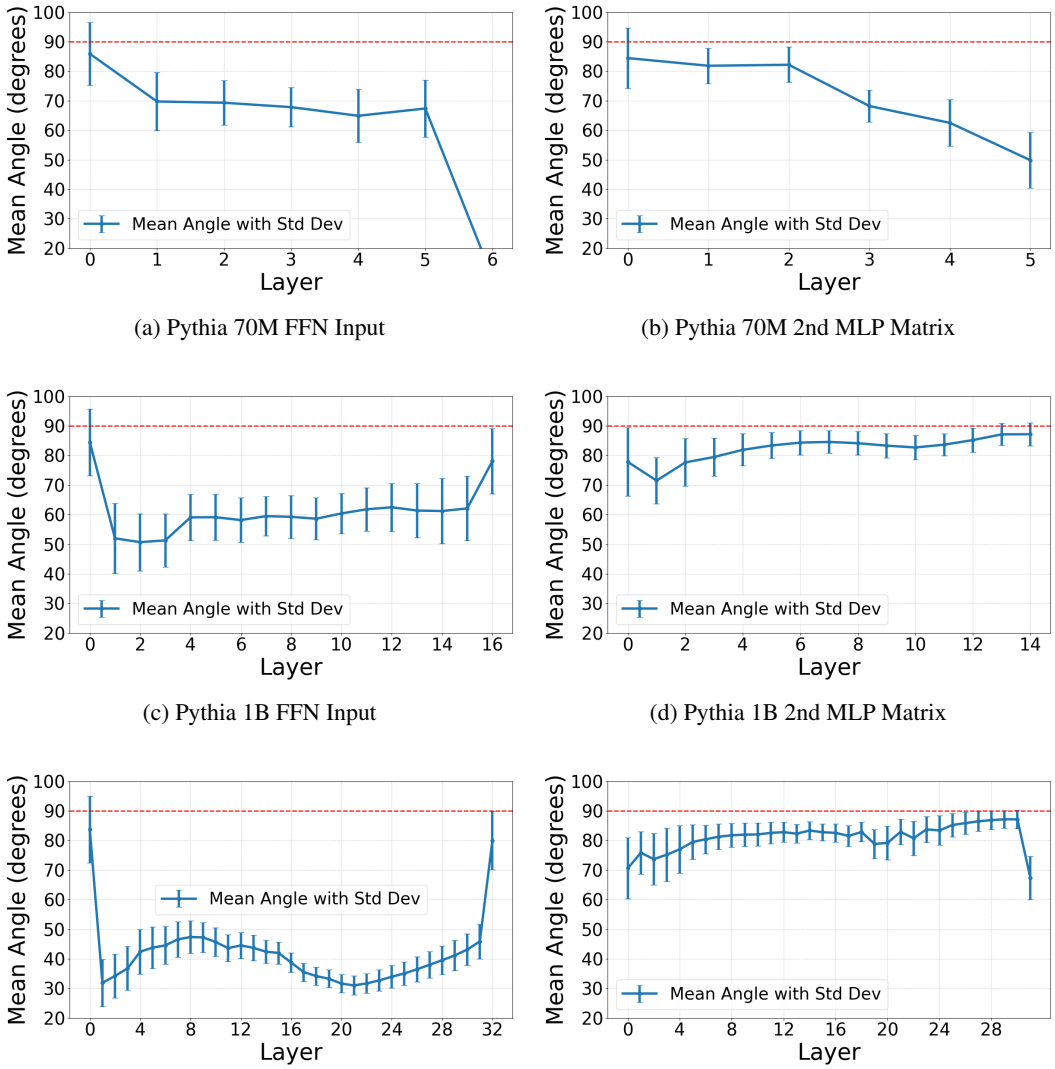
The feed-forward layers in transformer models have been proposed to function as key-value memory stores, where each layer maps input patterns to stored output distributions (Geva et al., 2020). This framework presents a different view of memory storage from linear associative memories by focusing on the role of activation patterns in retrieving stored information.

Each feed-forward network (FFN) layer in a transformer consists of two learned weight matrices: K (keys) and V (values). Given an input representation $\mathbf{x} \in \mathbb{R}^d$, the feed-forward transformation is defined as:

$$\text{FFN}(x) = f(x \cdot K^T) \cdot V \tag{6}$$

where $K \in \mathbb{R}^{m \times d}$ represents the key matrix, $V \in \mathbb{R}^{m \times d}$ is the value matrix, and f is a non-linearity (typically ReLU or GELU). The term $x \cdot K^T$ computes activation scores that determine which values contribute to the final output.

B Impact of Model Scale



(a) Pythia 70M FFN Input

(b) Pythia 70M 2nd MLP Matrix

(c) Pythia 1B FFN Input

(d) Pythia 1B 2nd MLP Matrix

(e) Pythia 6.9B FFN Input

(f) Pythia 6.9B 2nd MLP Matrix

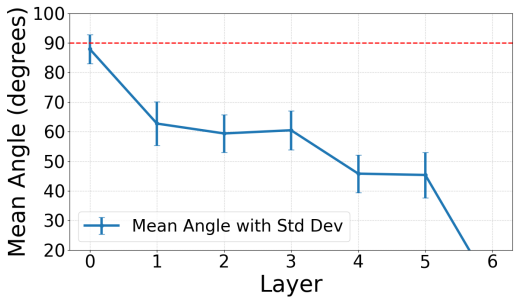
Figure 2: Comparison of pairwise angles for input layer and second MLP matrix across different scales of Pythia models.

The Pythia model family, ranging from 70M to 6.9B parameters, shows clear differences in how activation patterns change with model size. As seen in Figure 2a, smaller models have more variability and less stable memory representations, which affects how well they can recall information.

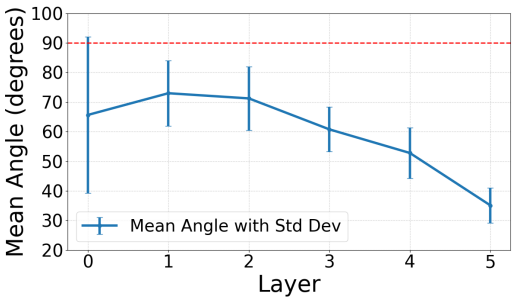
In the input layer, the 70M model quickly loses structure within the first six layers (Figure 2a). As models grow, this pattern changes: the 1B model shows more stability in the middle layers (Figure 2c), and the 6.9B model maintains more consistent patterns across layers (Figure 2e). Larger models appear better at separating different pieces of information.

The second MLP matrix also changes with model size. In the 70M model, interference increases as the layers deepen (Figure 2b). The 1B model shows less interference and more stable patterns (Figure 2d), while the 6.9B model keeps patterns distinct across layers (Figure 2f).

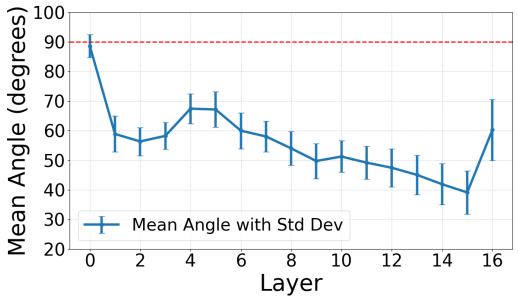
These results indicate that larger models handle memory better by maintaining more distinct patterns. The second MLP matrix becomes more reliable as model size increases, reducing interference and making it easier for the model to retrieve facts without confusion.



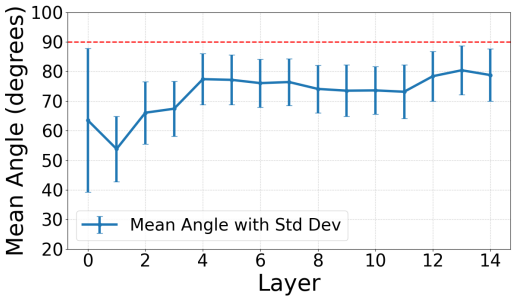
(a) Pythia 70M FFN Input Layer - CounterFact



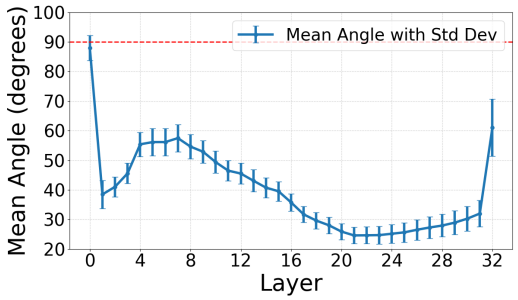
(b) Pythia 70M 2nd MLP Matrix - CounterFact



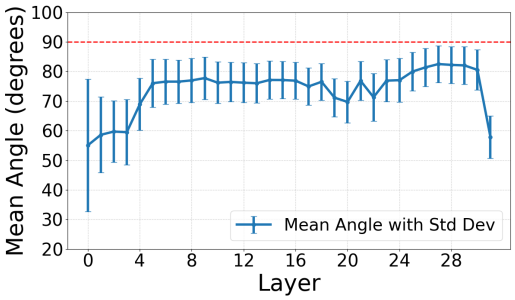
(c) Pythia 1B FFN Input - CounterFact



(d) Pythia 1B 2nd MLP Matrix - CounterFact



(e) Pythia 6.9B FFN Input - CounterFact



(f) Pythia 6.9B 2nd MLP Matrix - CounterFact

Figure 3: Comparison of pairwise angles for input layer and second MLP matrix across different scales of Pythia models for CounterFact dataset.

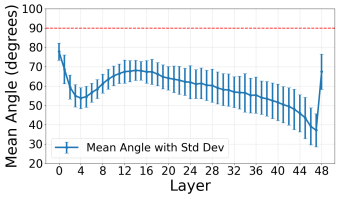
The CounterFact dataset reveals similar trends in how memory representations change with model scale. As shown in Figure 3a, smaller models exhibit more variability and less stable patterns. The 70M model, for instance, shows a sharp decline in structure across layers, indicating challenges in maintaining distinct memory representations.

In the input layer, the 70M model quickly loses structure within the first six layers (Figure 3a). The 1B model shows more stability in the middle layers (Figure 3c), while the 6.9B model maintains more consistent patterns across layers (Figure 3e). This suggests that larger models are better at preserving the distinctions needed for accurate factual recall.

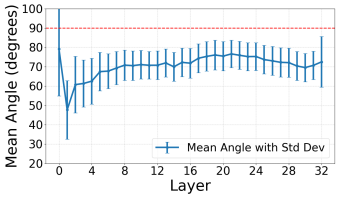
The second MLP matrix follows a similar pattern. The 70M model shows significant interference as layers deepen (Figure 3b), while the 1B model exhibits more stable patterns (Figure 3d). The 6.9B model demonstrates the most consistent behavior across layers (Figure 3f). This consistency

in larger models supports more reliable memory retrieval with less interference from unrelated information.

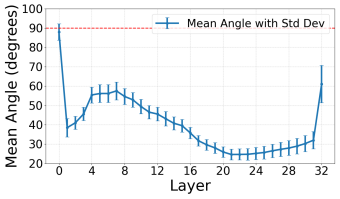
These findings indicate that larger models perform better on factual recall tasks, with more stable and distinct memory representations. The patterns observed align with the behavior seen in the Wikipedia-based experiments, suggesting that model scale helps maintain clear distinctions between memories across different contexts.



(a) GPT-2 XL FFN Input - CounterFact



(b) Llama FFN Input - CounterFact



(c) Pythia-6.9B FFN Input - CounterFact

Figure 4: Comparison of pairwise angles for input layers across GPT-2 XL, Llama, and Pythia-6.9B models using the CounterFact dataset.

The CounterFact dataset shows clear differences in how activation patterns change across GPT-2 XL, Llama, and Pythia-6.9B models. As seen in Figure 4a, GPT-2 XL exhibits a steady decline in structure across layers, with less stable memory patterns over time. Llama maintains more consistent patterns throughout, suggesting better memory organization across layers. Pythia-6.9B shows a sharp drop initially, followed by more stable patterns in later layers.

These results indicate that while all three models show a decrease in structure early on, the patterns of stabilization vary. Llama's consistent patterns suggest more reliable memory retention, while Pythia-6.9B shows more gradual changes across layers. GPT-2 XL maintains less stable patterns, with more variation as layers deepen.