Lifelong Knowledge Editing requires Better Regularization

Akshat Gupta[1*], Phudish Preteepamornkul[1,2*], Maochuan Lu[1]
Ahmed Alaa[1], Thomas Hartvigsen[3], Gopala Anumanchipalli[1]

[1]UC Berkeley, [2]SCB DataX, [3]University of Virginia

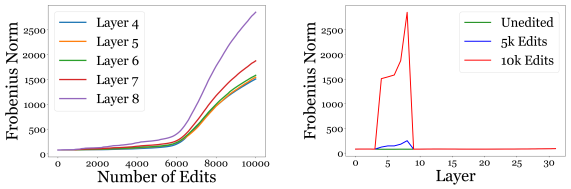
akshat.gupta@berkeley.edu

Abstract

Knowledge editing is a promising way to improve factuality in large language models, but recent studies have shown significant model degradation during sequential editing. In this paper, we formalize the popular locate-then-edit methods as a two-step fine-tuning process, allowing us to precisely identify the root cause of this degradation. We show that model degradation occurs due to (1) over-optimization of internal activations and (2) continuous norm-growth of edited matrices. To mitigate these issues, we introduce two regularization techniques: (1) Most-Probable Early Stopping (MPES) and (2) explicit Frobenius norm-constraint. We demonstrate that applying these simple yet effective regularization techniques at key points in the editing process can substantially mitigate model degradation. Combining these regularization methods enables scaling locate-then-edit methods to 10,000 edits while reducing editing time by 42-61%. These results show that targeted regularization is essential for lifelong knowledge editing.

1 Introduction

Knowledge editing entails editing specific facts that a language model has learned during pre-training in a data and computate efficient manner (De Cao et al., 2021; Yao et al., 2023). The most popular editors follow a "locate-then-edit" approach, where the methods aim to locate and update small subsets of a model's parameters that recall target facts (Meng et al., 2022a,b; Gupta et al., 2024a,c; Ma et al., 2024; Fang et al., 2024). Although these methods have shown strong results when performing singular or small-scale edits, they suffer substantial degradation when applied at scale (Gu et al., 2024a; Gupta et al., 2024b; Thede et al., 2025), leaving the problem of lifelong knowledge editing largely unsolved.



(a) Norm-growth as function of number of edits (b) Layer wise norm post-editing

Figure 1: The continuous growth of norm of edited MLP matrices in MEMIT Llama3-8B during sequential knowledge editing.

In this work, we systematically investigate the factors that cause model degradations and propose appropriate regularization methods to mitigate them, enabling large-scale sequential knowledge editing without sacrificing downstream performance. We begin by first formalizing locate-then-edit methods as a two-step fine-tuning process (section 4), where the first step uses gradient-descent for activation optimization, and the second step performs weight-update via a least-squares update rule. This explicit structure allows us to precisely diagnose the causes of loss of downstream performance and allows us to choose appropriate interventions.

We show that model degradation occurs during continuous sequential editing because of two reasons. Firstly, we show that the gradient-descent step in locate-then-edit methods leads to over-optimization of internal activations, which makes the model predict edited facts with unnaturally high confidence (section 5). We mitigate this by proposing **M**ost-**P**robable **E**arly **S**topping (MPES) - a novel variant of early-stopping where we halt gradient-descent steps when edited facts become most probable across all the different contexts used to calculate the loss. Secondly, we show that sequential editing leads to disproportionate growth of the Frobenius norm of the edited matrix (Figure 1). This allows the outputs produced from the edited layers to have an abnormally larger influence on the

* Equal Contribution

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 22653–22675
November 4-9, 2025

final output of the model, inadvertently causing loss of general ability, which might require information coming from other parts of the model. To address this, we incorporate a Frobenius norm-constraint into the editing objective of locate-then-edit methods (section 6.2). While early stopping and norm-constraints are known regularization methods, their targeted stage-specific application to locate-then-edit methods is non-trivial. With our work, we demonstrate that there exists a lack of proper regularization in existing knowledge editing methods and show that explicit regularization at the appropriate stages of editing is essential and critical for scalable and stable knowledge editing.

Our proposed regularization methods mitigate the respective issues of over-optimization of intermediate activation vectors and disproportionate norm-growth, consequently preserving downstream performance for a larger number of edits. Finally, we show that combining these two regularization methods allows us to perform up to 10,000 sequential edits while maintaining original downstream performance and making editing 41-62% faster.

2 Related Work

Knowledge editing methods can broadly be classified into three types (Yao et al., 2023; Zhang et al., 2024c) - *hypernetwork based editing methods* (De Cao et al., 2021; Mitchell et al., 2021), where an additional model is trained to predict the edited weights of the model, *in-context editing methods* (Mitchell et al., 2022; Hartvigsen et al., 2024; Zhong et al., 2023), where edited information is added in-context, and *locate-then-edit methods* (Meng et al., 2022a,b; Gupta et al., 2024c,a; Fang et al., 2024), where the knowledge source is traced to certain MLP layers (Hase et al., 2024; Yoon et al., 2024) before modifying them using a two-step fine-tuning method (section 4). This work focuses on performing locate-then-edit type of knowledge editing methods.

Overfitting During Knowledge Editing. Some recent works have studied overfitting as a key challenge of knowledge editing. For example, Zhang et al. (2024a) identifies that locate-then-edit methods overfit on edited facts. They propose to mitigate this by introducing an inference-stage regularization approach called "learn to inference" (LTI) that guides edited models to recall new knowledge by adding multi-stage constraints during optimization. With our proposed variant of early-stopping called "most-probable early stopping" (MPES), we present a much simpler way of reducing overfitting. We also show that LTI does not scale to multiple edits and leads to immediate model degradation when performing sequential editing. Meanwhile, MPES leads to significant improvements in downstream performance while countering overfitting (section 5).

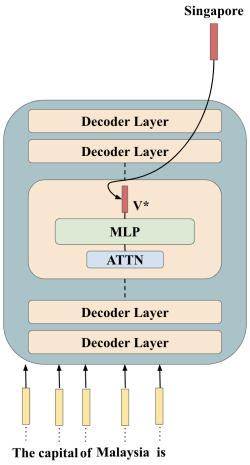
Norm-Growth and Weight Regularization. Prior work has shown that sequential knowledge editing leads to the norm-growth of the updated weight matrices (Gupta et al., 2024b; Yang et al., 2024). Ma et al. (2025) present a similar observation in the form of growth of condition number of the edited matrices as a potential cause of loss of downstream performance. However, none of these works explains why increasing norm of edited matrix is bad for the model. In our work, we analyze the effect of norm-growth on internal activations and show how it connects with model degradations (section 6.1). We verify our conclusions by adding a norm-constraint in the optimization objective of MEMIT. We show that our proposed norm-constraint combines seamlessly with MEMIT's closed-form solution, unlike prior regularization methods like PRUNE (Ma et al., 2025) and RECT (Gu et al., 2024b), which require additional regularization steps post-editing. We also show that our methods significantly outperform PRUNE and RECT in preserving downstream performance over a large number of edits.

3 Methods, Models, and Evaluation

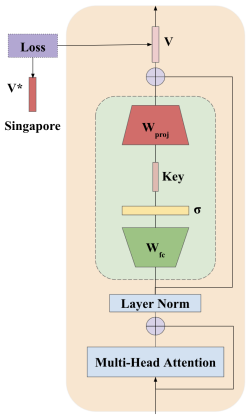
We adopt the experimental setting of AlphaEdit (Fang et al., 2024), a recent locate-then-edit method that is able to perform sequential editing for up to 3,000 facts. Following their setting, we perform sequential edits in batches of 100 facts. This means that 100 facts are edited into the model with each weight update, and multiple such updates are performed sequentially.

We evaluate all algorithms on three representative models - GPT2-XL (Radford et al., 2019), Llama2-7B (Touvron et al., 2023) and Llama3-8B (Yoon et al., 2024). All experiments are performed on the CounterFact (Meng et al., 2022a) and zsRE (Mitchell et al., 2021) datasets, which are standard knowledge editing datasets. We present the results

Code: https://github.com/myanonymousrepo/knowledge-editing-regularization



(a) gradient-descent step which finds the target activations for the MLP matrix.



(b) Target activations are used to update the second MLP matrix (in red).

Figure 2: Presenting locate-then-edit knowledge editing methods as a two-step fine-tuning process.

for Llama2-7B and Llama3-8B on the CounterFact dataset in the main paper and present the remaining results in the appendix due to space constraints.

In this paper, we evaluate the editing algorithms along two dimensions - editing performance and downstream performance. The editing performance evaluates the success of the knowledge editing algorithm in making successful edits, while downstream performance evaluates the extent of model degradation following prior work (Fang et al., 2024; Gupta et al., 2024b; Gu et al., 2024a).

Knowledge Editing Metrics: To evaluate editing performance, we use five standard knowledge editing metrics (Meng et al., 2022a). (i) Efficacy Score (ES), which measures if an edit was successfully made, (ii) Paraphrase Score (PS), which measures if the model is able to recall edited facts in different scenarios, (iii) Neighborhood Score (NS), which measures if edited facts disturb unrelated knowledge, (iv) Overall Score (S), which is the harmonic mean of ES, PS, and NS, and (v) Generation Entropy (GE), which measures the fluency of a model. A detailed explanation of these metrics is given in Appendix A. The editing metrics for each model are calculated after making all 10,000 sequential edits (Fang et al., 2024). This approach ensures that the metrics capture both the success of the edits of the latest batch of facts as well as facts that were edited in the past.

Downstream Performance Metrics: Following prior work (Fang et al., 2024; Gupta et al., 2024b), we measure downstream performance during knowledge editing using 6 tasks - massive multitask language understanding (MMLU) (Hendrycks et al., 2020), natural language inference (NLI, RTE) (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007), sentiment analysis (SST2) (Socher et al., 2013), paraphrase detection (MRPC) (Dolan and Brockett, 2005), and linguistic acceptability classification (CoLA) (Warstadt et al., 2019). Performance is assessed every 1000 edits, following Fang et al. (2024). Additional details are provided in Appendix N.

4 Knowledge Editing as a Two-Step Fine-tuning Process

"Locate-then-edit" family of methods like ROME (Meng et al., 2022a) , MEMIT (Meng et al., 2022b) and AlphaEdit (Fang et al., 2024) update factual knowledge expressed as (subject, relation, object) or (s,r,o). Instead of updating all parameters of a model to incorporate new knowledge, these methods only update certain matrices that are most responsible for fact recall (Meng et al., 2022a). The location of an edit within a model is described by a two-dimensional address - (i) an intermediate layer to be edited and (ii) a token from the list of input tokens used to create the target representation.

The exact layer to be edited is found via causal tracing (Meng et al., 2022a) or an empirical sweep across the model layers (Hase et al., 2024; Yoon et al., 2024). Only the second MLP matrix in the FFN module of the editing layer is updated (Geva et al., 2020; Meng et al., 2022a), forming the first part of the editing address. Meng et al. (2022a) also showed that using the output representation of the last subject token produces the best editing results, providing the second part of the address. The entire editing process is presented in Figure 2.

We explain this with an example. Given a fact to be edited, for example - *"The capital of Malaysia is Singapore"*, the query phrase for the editing process is *"The capital of Malaysia is"* and the target phrase is *"Singapore"*. The first part of the editing address, the exact layer whose second MLP matrix gets edited, is decided before the editing begins. The second part of the editing address is the token index of the last subject token, which in this case would be the last subword-token in *"Malaysia"*. The intermediate hidden representation

of this last subject token is used to make the edit. Once the editing address has been decided, instead of updating the chosen MLP weight matrix directly using gradient-descent, locate-then-edit knowledge editing proceeds in two steps.

1. In the first step (Figure 2(a)), gradient-descent is used to find the appropriate activation vector that acts as a target for the weight matrix to be edited. In the example, the found activation will cause the model to generate "Singapore" in response to the question. The loss function for the gradient-descent step is shown in equation 2. Note that in this step, no weights are updated and just an intermediate activation vector is found.

2. The weight update occurs in the second editing step (Figure 2(b)), where the MLP matrix is updated using the target activation vector found previously. This update uses a least square loss function, which preserves the MLP outputs for unrelated contexts while generating the target activation when the input corresponds to the query phrase.

The loss function used to update the MLP weight matrix is formulated using least-squares in the form of a preservation-memorization objective (Gupta et al., 2024c):

$$\underset{\hat{W}}{\text{argmin}}\ L(\hat{W}) \quad \text{where}$$

$$L(\hat{W}) = \underbrace{\lambda \sum_{i=1}^{P} \left\| \hat{W} k_0^i - W_0 k_0^i \right\|_2^2}_{\text{preservation}} + \underbrace{\sum_{j=1}^{B} \left\| \hat{W} k_e^j - v_e^j \right\|_2^2}_{\text{memorization}} \tag{1}$$

Specifically, W_0 represents the initial weights of the second MLP matrix, which is being edited to $\hat{W}$. k_0 represent input to the MLP matrix for information we want to preserve from the original model, and k_e is input activation vectors representing facts we want to insert into the model. v_e is the target activation vector found in step 1 of editing using gradient-descent. Since the above objective is linear in the argument, we do not need to use gradient-descent for optimization.

Thus, locate-then-edit methods can be seen as a unique type of *2-step fine-tuning method*. Instead of updating the MLP matrix directly using gradient-descent on the desired data, the weight update happens in two steps using two different types of objective functions for each step. The first step uses gradient-descent whereas the second step uses a closed-form solution.

5 Over-Optimization of Target Activations

The gradient-descent step for finding target activations as described in section 4 minimizes average cross-entropy loss for predicting the target fact for the query phrase augmented by 'N' random prefixes. The random prefixes are supposed to represent different contexts in which the edited fact can be recalled, thus aiming to create a more general query representation (Meng et al., 2022a). Let p represent the query-phrase input to the model, o^* represent the target fact to be edited into the model and x_j represent random prefixes added to the query phrase. Then, the gradient-descent step minimizes the following loss function:

$$L(\theta) = \frac{1}{N} \sum_{j=1}^{j=N} -log \mathcal{P}_\theta[o^* | x_j + p], \tag{2}$$

We observe that **when the gradient-descent step stops in these algorithms, the intermediate target representations become over-optimized, assigning unusually high probabilities to edited facts**. More specifically, the average target probability at which the gradient-descent step stops for all locate-then-end algorithms lies between 95-100%. When this target representation is used to update the the edited matrix using equation 1, the resultant edits are predicted with abnormally high probabilities. This can be seen in Table 1, where the edited facts are predicted with a much larger probability compared to the original "unedited" models. More details on the experimental settings can be found in Appendix B. This phenomenon was also observed by Zhang et al. (2024b).

5.1 MPES and Knowledge Editing

As seen in Table 1, the probability with which the unedited Llama2-7B, and Llama3-8B naturally recall a fact are 52% and 49% respectively. However, facts that get edited into the model with algorithms like MEMIT are predicted with an average probability of 78-79%. To overcome this hyper-optimization over a small subset of edited facts, we propose a variant of early stopping called "most-probable early stopping" (MPES).

Conventionally, early stopping is used during training while monitoring validation loss, where training is halted when the validation loss stops improving. In MPES, we stop the gradient-descent step in knowledge editing when the target fact becomes the most probable token for all 'N' query

Model	Prediction Probability				Time / edit (s)		
	Original fact	MEMIT	MEMIT w/ LTI	MEMIT w/ MPES	MEMIT	MEMIT w/ LTI	MEMIT w/ MPES
Llama-2 7B	0.52	0.78	0.30	0.45	4.84	8.67	2.79 (↓ 42 %)
Llama-3 8B	0.49	0.79	0.29	0.41	8.71	9.21	3.31 (↓ 61 %)

Table 1: Comparison between prediction probabilities of edited facts along with editing time when using MPES and LTI (Zhang et al., 2024b) with MEMIT.

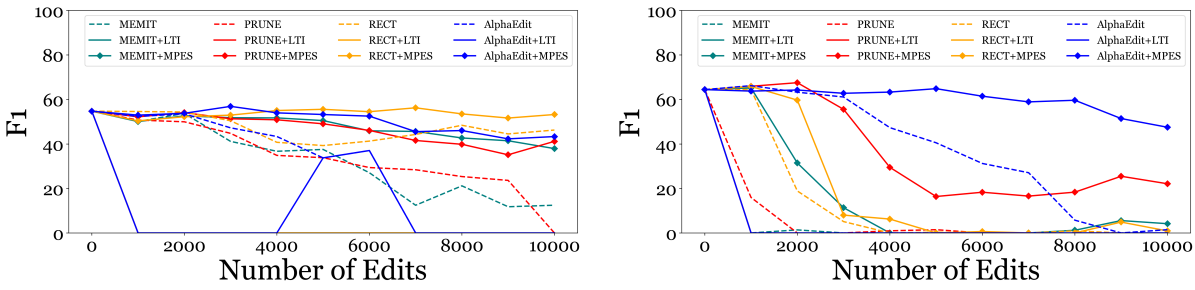
Method	Edit Score		Paraphrase Score		Neighborhood Score		Overall Score		Generation Entropy	
	Llama2-7B	Llama3-8B	Llama2-7B	Llama3-8B	Llama2-7B	Llama3-8B	Llama2-7B	Llama3-8B	Llama2-7B	Llama3-8B
MEMIT	81.04	49.68	64.67	49.29	60.95	**51.31**	67.86	50.08	442.59	373.48
MEMIT + LTI	55.34	51.85	53.57	51.23	49.21	49.21	52.58	50.70	562.07	372.08
MEMIT + MPES	**93.21**	**74.40**	**83.43**	**66.32**	**62.25**	50.78	**77.36**	**62.23**	**569.84**	**466.97**
RECT	82.42	63.17	66.84	56.92	**67.39**	52.89	71.54	57.36	549.35	**588.39**
RECT + LTI	53.87	51.55	52.17	51.08	50.35	50.43	52.09	51.02	498.97	544.41
RECT + MPES	**92.12**	**66.52**	**78.06**	**59.11**	67.16	**59.44**	**77.81**	**61.51**	**596.62**	410.82
PRUNE	70.80	49.38	62.11	49.63	51.86	51.09	60.60	50.02	280.83	340.22
PRUNE + LTI	55.19	49.88	52.98	49.63	50.33	50.19	52.76	49.90	548.23	239.17
PRUNE + MPES	**91.48**	**92.66**	**83.65**	**82.20**	**61.91**	**64.82**	**76.85**	**78.16**	**568.91**	**512.83**
AlphaEdit	61.10	72.67	55.86	63.44	53.75	52.90	56.74	61.95	540.92	465.81
AlphaEdit + LTI	53.82	53.27	53.03	51.70	49.68	49.72	52.11	51.52	524.11	240.07
AlphaEdit + MPES	**84.15**	**88.43**	**74.94**	**82.08**	**62.87**	**56.60**	**72.93**	**72.83**	**583.40**	**565.32**

Table 2: Sequential knowledge-editing performance after 10,000 edits. Scores are shown for nine MEMIT-based and AlphaEdit algorithms under two models. MPES consistently improves editing metrics across all settings.

phrases used for optimization (equation 2). This contrasts with the current stopping criteria in these methods, where gradient-descent is halted after either 20 iterations or upon reaching a loss threshold. These criteria leave open the possibility of over-optimization (and hence overfitting, as observed empirically in Table 1), or underfitting. Apart from preventing the model from becoming overly optimized towards a specific target fact, using MPES for halting gradient-descent also has two other advantages. *Firstly, it simplifies monitoring of the gradient-descent process and provides a principled approach to stopping gradient-descent which is directly tied to the knowledge editing objective, that is, accurately recalling edited facts in a variety of scenarios without overfitting. Secondly, by optimally stopping the gradient-descent process, we also improve the efficiency of locate-then-edit algorithms. MPES reduces the average edit time by 42-61% compared to the standard MEMIT.*

We compare MPES with LTI (Zhang et al., 2024b), which prevents over-optimization during the gradient-descent step using multiple additional loss functions that also increase the average time per edit (LTI takes approximately three times as long as MPES as shown in Table 1). We combine MPES with multiple knowledge editing methods including MEMIT (Meng et al., 2022b), RECT (Gu et al., 2024b), PRUNE (Ma et al., 2025) and AlphaEdit (Fang et al., 2024). We perform 10,000 sequential edits in batches of 100 as described in section 3. The editing and downstream evaluation metrics for sequential editing for both regularization methods can be seen in Table 2 and Figure 3 respectively. We see that MPES provides substantial gain across all key metrics.

Table 2 shows that all editing metrics are enhance significantly for all knowledge editing methods when combined with MPES. Infact, Table 2 shows that LTI harms the editing metrics for sequential editing when compared to even the respective baselines. The downstream performance metrics can be seen in Figure 3. We see that knowledge editing methods augmented with MPES (represented by solid lines with diamond dots) are able to maintain their downstream performance for much longer when compared to the corresponding baselines and LTI. However, despite these improvements, downstream degradation still occurs after a few thousand edits especially in larger models like Llama3-8B. This suggests that MPES alone may not fully address all failure modes, which we explore further in the next section. To summarize, **with MPES we present a principled way of stopping the gradient-descent step during knowledge editing** which results in improved editing performance, delays loss of downstream performance, and makes current knowledge editing meth-



(a) Llama2-7B (b) Llama3-8B

Figure 3: Downstream evaluation when comparing MPES (our method) with LTI for regularizing over-optimization of target activations during knowledge editing.

ods much faster.

6 Norm Growth during Sequential Knowledge Editing

Previous studies show that sequential knowledge editing increases the norm of the edited matrix (Gupta et al., 2024b; Yang et al., 2024; Ma et al., 2025). Figure 1 illustrates this issue, highlighting the extreme norm growth in edited layers of Llama3-8B using MEMIT. Here, the norms of the edited layers increase by more than 10 times, while the unedited layer remains unchanged (Figure 1(b)). Furthermore, norm growth persists continuously during editing, as shown in Figure 1, for not even one edit does the norm remain constant or decrease.

While the anomalous norm-growth was observed in prior work, they do not explain how it affects the general ability of the model. We answer this question by analyzing the residual stream of the model during large-scale knowledge editing.

6.1 Explaining Loss of Downstream Performance due to Norm-Growth

To understand the impact of this norm growth, we analyze how residual connections work in decoder-only LLMs. The intermediate hidden vector at layer l, represented by h^l, is also sometimes referred to as the *residual stream*. Each decoder-only layer contains one attention and FFN module that feeds directly into the residual stream through residual connections. The exact computations happening within transformer-based decoder-only LLMs can be found in Appendix E. Let the output of the attention module at layer l be represented by a^l, and the output of the FFN module be represented by m^l. As the vectors computed in the attention and MLP modules get added back to the residual stream at each layer, the residual stream represents a summation of an increasing number of vectors as we go deeper into the model. A non-recursive formula for the output of the transformer just before unembedding and final layernorm is shown below:

$$h^L = h^0 + \sum_{i=1}^{i=L} a^i + \sum_{i=1}^{i=L} m^i \tag{3}$$

Here, L represents the total number of layers in a model and h^L represents the residual stream after the final layer. Thus, the output vector at the final layer is a summation of the outputs of individual self-attention and MLP sub-modules.

Now, if the norm of the edited MLP matrix grows as disproportionately as shown in Figure 1, the norm of the vectors that are produced from those edited MLP sub-modules will also grow. This means that the norms of the vectors m^l in the summation corresponding to the edited layers will grow substantially. As the norm of a few vectors in the summation grows, these vectors will begin to dominate the sum. Proof for this is shown in Appendix D.1, where we show that if the norm of a vector in a summation grows, the overall sum effectively tends towards that vector.

We also show this effect empirically. The growing norm of activation vectors produced by edited layers after editing Llama3-8B can be seen in Figure 4. After editing using MEMIT, which edits layers 4-8 for 10,000 sequential edits (edited layers are shown in red color on the x-axis), we see a drastic increase in the norm of activation vectors produced by edited layers. For example, the activation vectors produced by layer 8 account for almost 40% of the total norm, and vectors produced by all edited layers account for 85% of the total. *To emphasize how extreme this is, the residual stream*

for Llama-3-8B is made up of a summation of 65 vectors, and 4 out of the 65 vectors coming from the edited layers account for 85% of the total norm. The norm-contributions for an unedited model can be seen in Figure 9(a) (appendix), where the four edited layers account for less than 4% of the overall norm. Note that this effect cannot be mitigated by LayerNorm or RMSNorm, which just normalize the incoming vectors (Brody et al., 2023), whereas the norm growth of edited layers changes the content of the final hidden representation by making it consist of mostly the output of the edited layers.

This gives us a crucial insight into why the growing norm of edited matrices leads to a loss of downstream performance. **As the norms of the edited matrices increase, and as a consequence, the norm of the activation vectors produced from those matrices also increases, the activation vector at the final layer (h^L) is dominated by the output of edited layers, thus giving the edited layers a larger authority over the final representations. This allows edited layers to override the information produced from other parts of the model**, possibly also helping make successful edits. However, as the norm of the edited matrix continues to grow when we edit it sequentially, suddenly the final representations begin to largely be composed of the outputs of only the edited layers. This makes it impossible for the model to account for the information processed from other sub-modules, which might be important to perform other downstream or unrelated tasks.

6.2 Introducing Norm Constraint

In the above discussion, we show how growing norm of edited matrices is detrimental to the functioning of edited models and hypothesize how it causes loss of downstream performance. To test these conclusions, we propose to add an additional term to the editing objective to control this norm growth. Thus, we augment the MEMIT objective with a Frobenius norm-constraint:

$$L(\hat{W}) = \underbrace{\lambda_p \sum_{i=1}^{P} \left\| \hat{W} k_0^i - W_0 k_0^i \right\|_2^2}_{\text{preservation}} + \underbrace{\sum_{j=1}^{B} \left\| \hat{W} k_e^j - v_e^j \right\|_2^2}_{\text{memorization}} + \underbrace{\lambda_n \left\| \hat{W} - W_0 \right\|_F^2}_{\text{norm-constraint}} \tag{4}$$

Llama-3-8B has 32 layers, each contributing two vectros a^l and m^l to the residual stream along with the input vector.

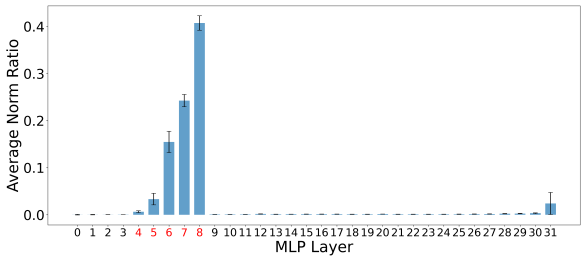


Figure 4: The figure shows the proportion of contribution of activation vectors from each sub-module to the residual stream. Edited layers are shown in red.

The original weight matrix is represented by W_0 and $\hat{W}$ represents the edited matrix. The above objective has a closed-form solution as shown below (proof, Appendix F):

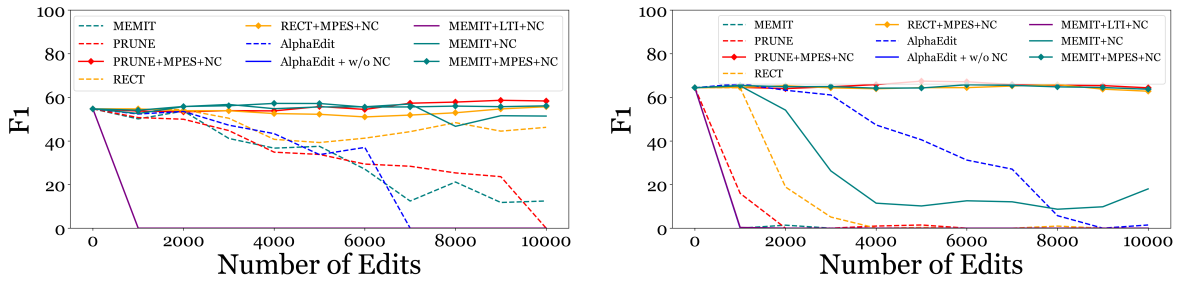
$$\hat{W} = W_0 + \Delta \quad \text{where}$$
$$\Delta = (V_1 - W_0 K_1) K_1^T (\lambda_p K_0 K_0^T + K_1 K_1^T + \lambda_n I)^{-1} \tag{5}$$

Table 3 and Figure 5 show the editing and downstream performance after 10,000 sequential edits using the norm-constraint. Compared to baseline methods such as MEMIT, AlphaEdit, as well as regularization approaches such as PRUNE and RECT, Frobenius norm-constraint version of MEMIT (represented by MEMIT + NC) produces consistent and substantial improvements in editing and downstream performance metrics. Specifically, we see in Table 3 that MEMIT + NC outperforms all previous models on all editing metrics except for the generation entropy metric. This shows that explicitly controlling norm growth is more effective than prior regularization methods like PRUNE and RECT, particularly when editing at scale. When looking at downstream performance in Figure 5, MEMIT + NC also outperforms all prior editing methods when looking for Llama-2-7B, and all methods except AlphaEdit for Llama3-8B. Moreover, the observed gains in downstream performance support our hypothesis that uncontrolled norm growth in edited layers leads to downstream degradation.

Additionally, we want to note that AlphaEdit inherently has an explicit norm-constraint in its optimization objective. To present the importance of norm-constraint during large scale editing, we remove this term from the objective and evaluate the method, as shown in Table 3. To our surprise, we find that AlphaEdit completely fails without the norm constraint, as shown in the row *'AlphaEdit w/o NC'* in Table 3 . Specifically, after 10,000 edits, the model effectively collapes, where Llama2-7B

Method	Edit Score		Paraphrase Score		Neighborhood Score		Overall Score		Generation Entropy	
	Llama2-7B	Llama3-8B	Llama2-7B	Llama3-8B	Llama2-7B	Llama3-8B	Llama2-7B	Llama3-8B	Llama2-7B	Llama3-8B
MEMIT	81.04	49.68	64.67	49.29	60.95	51.31	67.86	50.08	442.59	373.48
PRUNE	70.80	49.38	62.11	49.63	51.86	51.09	60.60	50.02	280.83	340.22
PRUNE + MPES + NC	**91.33**	85.18	**79.12**	75.75	57.72	58.55	73.33	71.39	475.63	538.49
RECT	82.42	63.17	66.84	56.92	**67.39**	52.89	71.54	57.36	**549.35**	**588.39**
RECT + MPES + NC	89.51	82.52	74.13	69.35	60.02	**64.96**	72.60	71.54	480.66	587.79
AlphaEdit	61.10	72.67	55.86	63.44	53.75	52.90	56.74	61.95	540.92	465.81
AlphaEdit w/o NC	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	434.17	560.92
MEMIT + LTI + NC	54.41	50.76	51.87	48.89	52.05	51.17	52.75	50.25	516.24	575.59
MEMIT + NC	87.89	85.72	79.10	77.08	59.72	58.45	73.59	71.86	517.86	367.46
MEMIT + MPES + NC	91.09	**87.54**	78.02	**77.97**	59.78	59.30	**74.03**	**72.97**	545.72	536.43

Table 3: Editing performance of our approach when compared to baseline MEMIT, AlphaEdit and MEMIT regularization method such as PRUNE and RECT.



(a) Llama2-7B (b) Llama3-8B

Figure 5: Average downstream performance for during sequential editing with compared to baseline of MEMIT and addition of MPES and Norm-Constraint (NC).

outputs <unk> and Llama3-8B loops on "!" and all editing scores drop to zero. We also illustrate the unbound growth of norm in Figures 26(a) and 26(b) in Appendix L. These results support our analysis in section 6.1 and re-emphasize that managing norm-growth is essential for stable and successful large-scale editing.

6.3 Combining MPES and Norm Constraint

In this section, we combine MPES, which intervenes in the gradient-descent step, with explicit Frobenius norm constraint, which regularizes the weight-update step in MEMIT. This combined approach, MEMIT + MPES + NC, achieves the strongest results across both editing score and downstream metrics. As shown in Table 3, it consistently improves the consistency of editing scores between models, with the largest gains observed on Llama3-8B. More importantly, Figures 5(a) and 5(b) show that the combination of the two regularization methods beautifully preserves downstream task performance with great effectiveness when compared to other algorithms, even after 10,000 sequential edits.

These findings suggest that MPES and norm-constraint address complementary failure modes during large-scale sequential editing. Their combination provides regularization methods that enable scalable knowledge editing while having minimal loss of downstream task performance. The generalizability of these findings can be seen by the compatibility of MPES with other knowledge editing methods, and the neccesity and importance of managing norm-growth in both MEMIT and AlphaEdit.

7 Conclusion

In this paper, we show that existing knowledge editing methods require appropriate regularization when scaled sequentially to a large number of edits. Specifically, we diagnose and address two core failure modes in large-scale knowledge editing, (1) over-optimization of intermediate activations and (2) uncontrolled norm growth of edited matrices. To mitigate these, we introduce stage-specific regularization techniques in the form of "most-probable earlys stopping" (MPES) and Frobenius Norm-Constraint, which generalize to multiple knowledge editing methods. With the combination of both these regularization methods, we are able to make 10,000 sequential edits while consistently maintaining the downstream performance levels of

the unedited model.

8 Limitation

Although our method significantly improves editing speed, scaling beyond 10,000 edits still remains challenging due to compute constraints. Performing 10,000 sequential edits on Llama3-8B already required a full day of GPU time on our hardware. This is because running these edits is combined with a lot of analysis, including downstream performance evaluation and measuring editing accuracy metrics on past edited facts. In contrast, baseline methods such as MEMIT combined with LTI were substantially slower, making larger-scale comparisons infeasible. Future work could explore more efficient optimization techniques or hardware acceleration to push beyond this limit.

Acknowledgements

This work was supported in part by the NVIDIA Academic Grant Program award.

References

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. *TAC*, 7:8.

Shaked Brody, Uri Alon, and Eran Yahav. 2023. On the expressivity role of layernorm in transformers' attention. *Preprint*, arXiv:2305.02582.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Xiang Wang, Xiangnan He, and Tat-seng Chua. 2024. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*.

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

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024a. Model editing can hurt general abilities of large language models. *arXiv preprint arXiv:2401.04700*.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024b. Model editing harms general abilities of large language models: Regularization to the rescue. *Preprint*, arXiv:2401.04700.

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

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

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

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7, pages 785–794.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2024. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2024. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. 2024. Perturbation-restrained sequential model editing. *arXiv preprint arXiv:2405.16821*.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. 2025. Perturbation-restrained sequential model editing. *Preprint*, arXiv:2405.16821.

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

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

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Lukas Thede, Karsten Roth, Matthias Bethge, Zeynep Akata, and Tom Hartvigsen. 2025. Understanding the limits of lifelong knowledge editing in llms. In *International Conference on Machine Learning*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models, 2023. *URL https://arxiv. org/abs/2307.09288*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Wanli Yang, Fei Sun, Xinyu Ma, Xun Liu, Dawei Yin, and Xueqi Cheng. 2024. The butterfly effect of model editing: Few edits can trigger large language models collapse. *arXiv preprint arXiv:2402.09656*.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*.

Junsang Yoon, Akshat Gupta, and Gopala Anumanchipalli. 2024. Is bigger edit batch size always better?–an empirical study on model editing with llama-3. *arXiv preprint arXiv:2405.00664*.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024a. Uncovering overfitting in large language model editing. *Preprint*, arXiv:2410.07819.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024b. Uncovering overfitting in large language model editing. *arXiv preprint arXiv:2410.07819*.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, and 1 others. 2024c. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

A Knowledge Editing Metrics

A more detailed explanation of the knowledge editing metrics used in this paper is below:

1. **Efficacy Score (ES)**: assesses whether an edit has been successful. It is calculated as the percentage of edits where $P(\text{new fact}) > P(\text{old fact})$ when evaluated on paraphrases of the query prompt.

2. **Paraphrase Score (PS)**: measures the model's ability to generalize after an edit. Specifically, it is the percentage of edits where $P(\text{new fact}) > P(\text{old fact})$ for paraphrased versions of the query prompt.

3. **Neighborhood Score (NS)**: evaluates the locality of a model edit by determining whether editing one fact affects other facts stored in the model. It is the percentage of unaffected facts in the neighborhood of the edited fact.

4. **Generation Entropy (GE)**: measures the fluency of the model's text generation post-edit. GE is computed as the weighted average of bi-gram and tri-gram entropies in the text generated by the edited model. A lower GE indicates repetitive text generation, a common failure mode (Meng et al., 2022a; Gupta et al., 2024b).

5. **Score (S)**: introduced by (Meng et al., 2022a), this composite metric combines edit success, generalization, and locality into a single score. It is calculated as the harmonic mean of the Efficacy Score (ES), Paraphrase Score (PS), and Neighborhood Score (NS).

B Experimental Detail on Overfitting During Knowledge Editing

For each method and model, we conducted three different experiments:

1. Unedited fact recall probability - In this case, we calculate the average probability with which a fact is recalled by the unedited/original model. These are the facts that the model learnt through its pre-training. The model is asked questions from the CounterFact dataset, and we average the probability with which the model predicts the fact correctly.

2. Edited fact probability without MPES - In this case, we evaluate the probability with which a model recalls a fact that is edited into the model. This is the standard baseline case without MPES.

3. Edited fact probability with LTI - Here we also evaluate the average probability with which a model recalls an edited fact. In this case we used LTI during editing the fact.

4. Edited fact probability WITH MPES - Here we also evaluate the average probability by which a model recalls an edited fact. In this case, MPES is used during editing the fact.

In each of the experiments we performed, we passed average numbers over 1000 edited facts with a batch size of 1. We use the CounterFact dataset (Meng et al., 2022a) for all of these experiments. We show the result in table 1. As we can see from the table, the unedited fact token probability is pretty low but once we run the edited fact the probability increases to almost 1 for some cases. MPES brings the probability of fact recall for edited facts down to a more natural value, which prevents the overfitting problem that we present in this paper.

C GPT2-XL Result

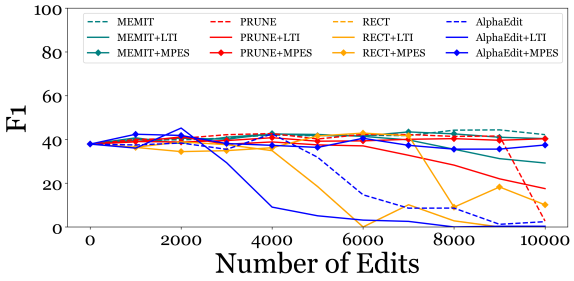
In this section we show the result for GPT2-XL on CounterFact dataset in Tables 4 and 5 and also the downstream average downstream performance in Figure 6. As we can see that the result for GPT2-XL also improves the results as well.

Method	Edit Score	Paraphrase Score	Neighborhood Score	Overall Score	Generation Entropy
MEMIT	**94.04**	**79.91**	57.90	**74.22**	517.37
MEMIT + LTI	82.03	70.51	56.29	67.97	508.35
MEMIT + MPES	92.17	77.01	**60.38**	72.26	**523.57**
PRUNE	61.05	58.05	50.00	55.96	**579.69**
PRUNE + LTI	**80.06**	**69.77**	56.07	**67.18**	542.41
PRUNE + MPES	76.91	59.11	**65.96**	66.54	563.57
RECT	51.40	49.83	52.17	51.12	409.42
RECT + LTI	47.51	47.37	52.74	49.08	185.96
RECT + MPES	**51.99**	**49.93**	**54.29**	**52.01**	**523.63**
AlphaEdit	88.58	70.33	56.04	69.20	580.27
AlphaEdit + LTI	74.20	59.90	54.86	61.98	**596.47**
AlphaEdit + MPES	**95.52**	**82.08**	**60.03**	**76.32**	565.44

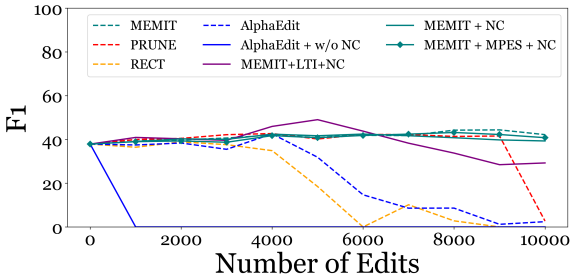
Table 4: Knowledge editing performance for GPT2-XL on the CounterFact dataset for different algorithms in combination with MPES.

Method	Edit Score	Paraphrase Score	Neighborhood Score	Overall Score	Generation Entropy
MEMIT	**94.04**	**79.91**	57.90	74.22	517.37
PRUNE	61.05	58.05	50.00	55.96	579.69
RECT	51.40	49.83	52.17	51.12	409.42
AlphaEdit	88.58	70.33	56.04	69.20	**580.27**
AlphaEdit + w/o NC	47.37	47.37	52.63	49.00	573.40
MEMIT + LTI + NC	84.15	73.12	54.91	68.54	554.47
MEMIT + NC	93.89	80.9	58.00	74.53	504.68
MEMIT + MPES + NC	93.99	79.79	**59.98**	**75.29**	517.44

Table 5: Knowledge editing performance for GPT2-XL on the CounterFact dataset for different algorithms in combination with MPES and NC.



(a) GPT2-XL downstream performance comparison with MPES



(b) GPT2-XL downstream performance comparison with MPES + NC

Figure 6: Average downstream performance for during sequential editing with compared to baseline of MEMIT and addition of MPES and Norm-Constraint (NC).

D Experimental Details on Activation Norm Growth

To assess the impact of the activation vectors generated by the edited layers before and after editing, we conducted an experiment where we edited our model on a total of 10,000 facts using a batch size of 100. Once the model was edited, we evaluated the norm of the activations produced by each layer by passing it through a Wikipedia dataset containing 30,000 examples. For each example, the model performed a one-word prediction task given an input context, and we measured the norm of the activation vectors produced from each layer in the model. We repeated the same process for the unedited model to compare the differences.

For each of the 30,000 examples, we calculated the proportion of the activation norm at each layer. We then plotted the mean and standard deviation of these proportions for both the edited and unedited models in figures 7 - 12. As shown in our results, the proportion of activation norms for the layers that were edited is significantly higher than their neighboring layers. In fact, some of the edited layers show the highest proportions overall. The edited layers are highlighted with *red color* on the x-axis.

D.1 Theoretical Analaysis of Growth of Vector Norm in a Summation

We want to understand the effect of excessive increase in the norm of a vector in a sum of vectors. First, let's start with an easy example where we have a summation of two vectors, $\mathbf{s} = \mathbf{a} + \mathbf{b}$ and then there is excessive increase in the norm of the first vector, that is $\mathbf{s} = k\mathbf{a} + \mathbf{b}$ where k is some positive scalar. To evaluate the effect of this increase, we analyses the tendencies of the sum $\mathbf{s}$ as k increases.

We first want to understand the norm of $||\mathbf{s}||$. We have the following :

$$||\mathbf{s}||^2 = ||k\mathbf{a} + \mathbf{b}||^2 = k^2||\mathbf{a}||^2 + 2k\mathbf{a} \cdot \mathbf{b} + ||\mathbf{b}||^2$$

From this we can clearly see that as k increases the first term quadratic in k will dominate. This means that as $k \to \infty$, $||\mathbf{s}||^2 \to k^2||\mathbf{a}||^2$, or $||\mathbf{s}|| \to k||\mathbf{a}||$, which is the norm of the new vector. Thus, as the norm of one of the vectors in the summation increases, the norm of the summation tends to the norm of that vector with increasing norm.

Next, we look at the tendencies of the orientation of the summation as the norm of one vector increases. Let θ be the angle between $\mathbf{s}$ and $k\mathbf{a}$. Then, the cosine of the angle between the summation and the new vector $k\mathbf{a}$ is as follows (note that angle between $\mathbf{s}$ and $k\mathbf{a}$ is same as the angle between $\mathbf{s}$ and $\mathbf{a}$):

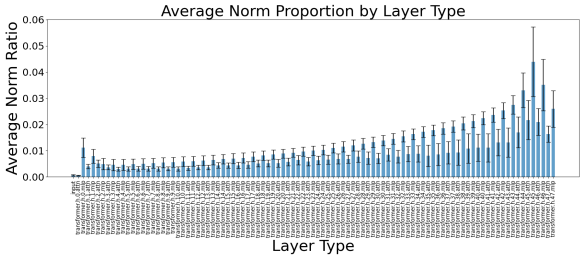
$$\cos\theta = \frac{\mathbf{s} \cdot \mathbf{a}}{||\mathbf{s}||||\mathbf{a}||} = \frac{(k\mathbf{a} + \mathbf{b}) \cdot \mathbf{a}}{||k\mathbf{a} + \mathbf{b}||||\mathbf{a}||}$$

In the limit of $k \to \infty$, $||\mathbf{s}|| \to k||\mathbf{a}||$ as shown above. Thus, the cosine expression in the limit can be written as:

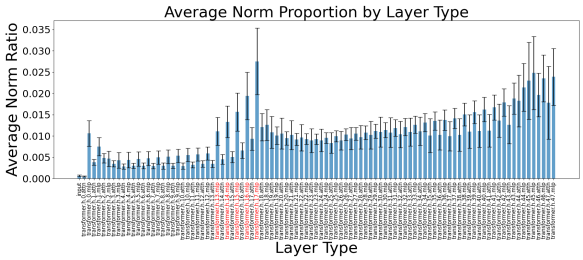
$$\begin{aligned}\lim_{k\to\infty} \cos\theta = \frac{(k\mathbf{a} + \mathbf{b}) \cdot \mathbf{a}}{k||\mathbf{a}||^2} &= \frac{k\mathbf{a} \cdot \mathbf{a}}{k||\mathbf{a}||^2} + \frac{\mathbf{b} \cdot \mathbf{a}}{k||\mathbf{a}||^2} \\ &= 1 + \frac{\mathbf{b} \cdot \mathbf{a}}{k||\mathbf{a}||^2}\end{aligned}$$

Thus, as $k \to \infty$, the cosine of angle between the sum and the vector tends to 1, or the angle between the summation and the vector tends to zero. This shows that as the norm of a vector in the summation continues to increase, the both the norm and the orientation of the summation tends towards the vector with increasing norm.

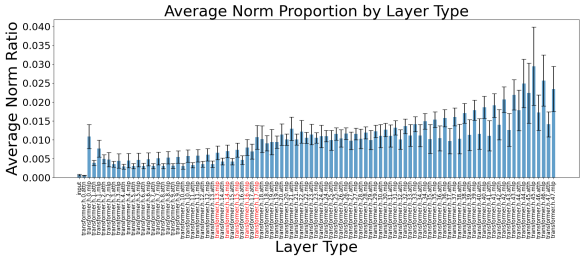
Finally, the same proof can be generalied to a summation of multiple vectors, where $\mathbf{b}$ represents the sum of the remaining vectors.



(a) Average Norm Proportion For Unedited GPT2-XL

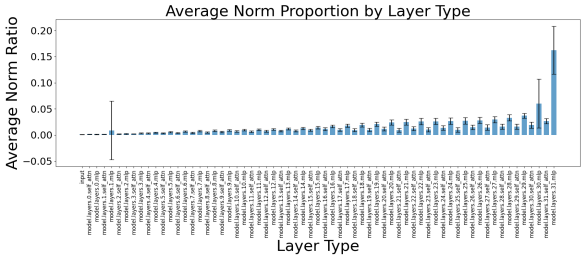


(b) Average Norm Proportion for GPT2-XL using Alphaedit

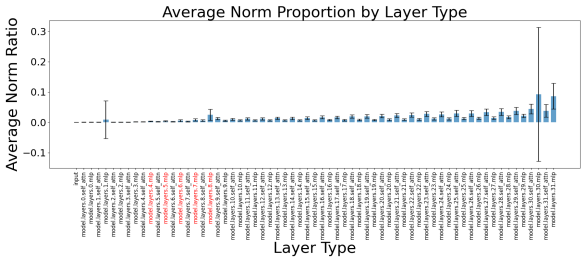


(c) Average Norm Proportion for GPT2-XL using MEMIT

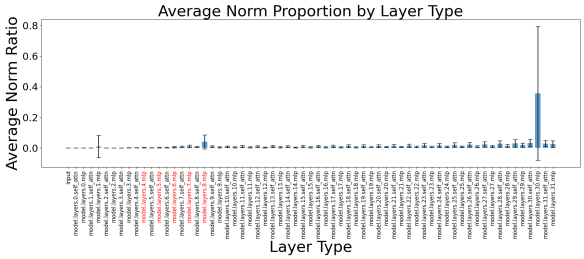
Figure 7: Activation norm growth for GPT2-XL.



(a) Average Norm Proportion For Unedited Llama2-7B

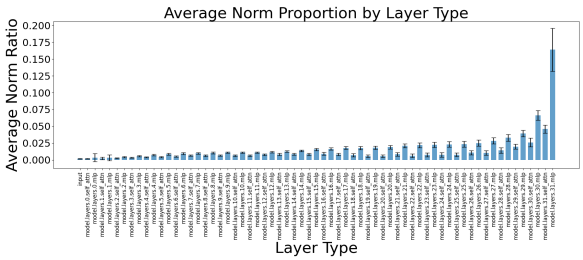


(b) Average Norm Proportion for Llama2-7B using Alphaedit

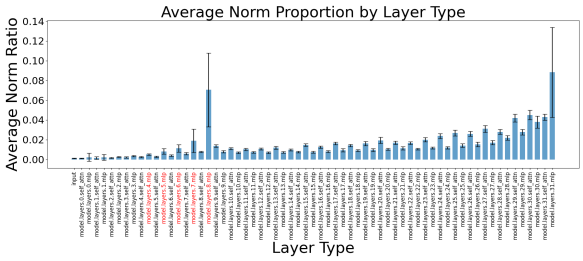


(c) Average Norm Proportion for Llama2-7B using MEMIT

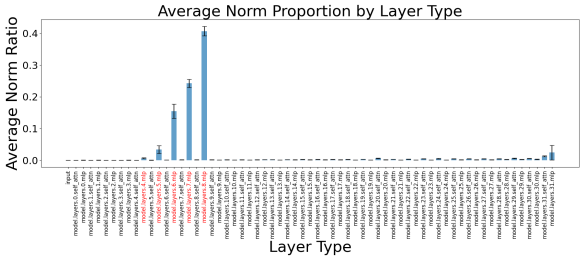
Figure 8: Activation norm growth for Llama2-7B.



(a) Average Norm Proportion For Unedited Llama3-8B

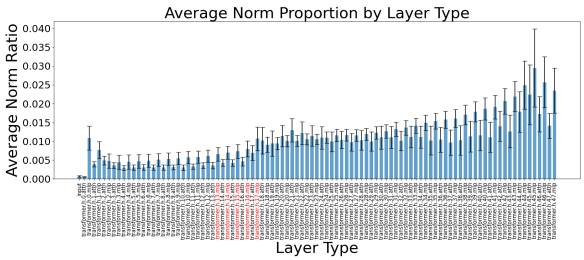


(b) Average Norm Proportion for Llama3-8B using Alphaedit

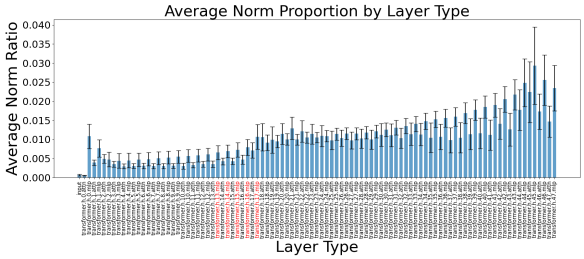


(c) Average Norm Proportion for Llama3-8B using MEMIT

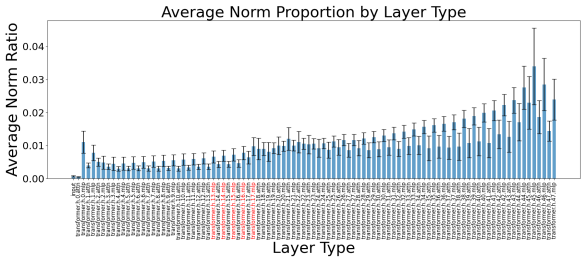
Figure 9: Activation norm growth for Llama3-8B.



(a) Average Norm Proportion For GPT2-XL using MEMIT

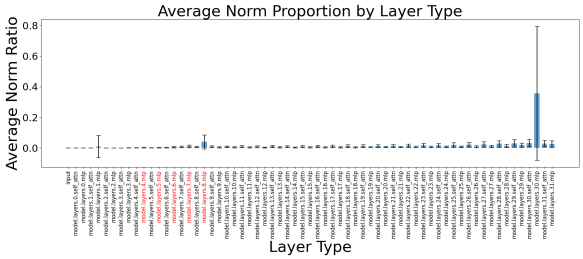


(b) Average Norm Proportion for GPT2-XL using NC

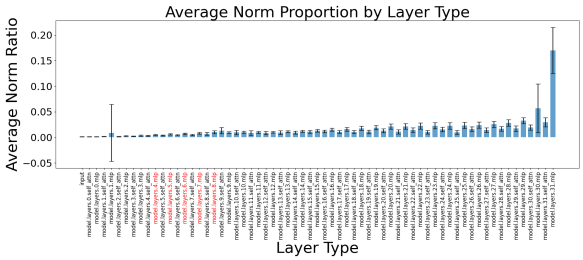


(c) Average Norm Proportion for GPT2-XL using MPES + NC

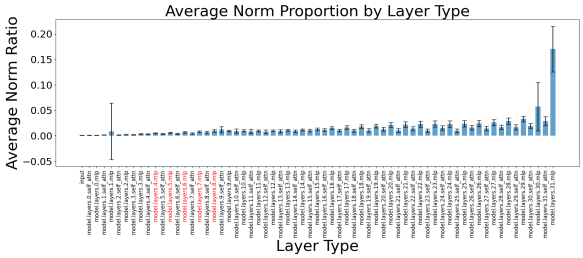
Figure 10: Activation norm growth for GPT2-XL using NC and MPES + NC.



(a) Average Norm Proportion For Llama2-7B using MEMIT

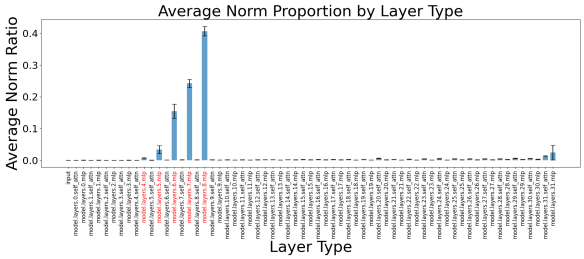


(b) Average Norm Proportion for Llama2-7B using NC

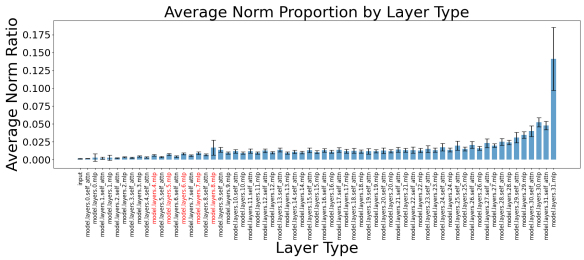


(c) Average Norm Proportion for Llama2-7B using MPES + NC

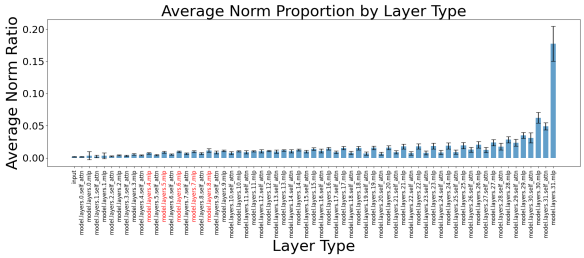
Figure 11: Activation norm growth for Llama2-7B using NC and MPES + NC.



(a) Average Norm Proportion For Llama3-8B using MEMIT



(b) Average Norm Proportion for Llama3-8B using NC



(c) Average Norm Proportion for Llama3-8B using MPES + NC

Figure 12: Activation norm growth for Llama3-8B using NC and MPES and NC.

E Internal Computations within Decoder-Only LLMs

The internal computations happening within a transformer-based decoder-only LLM is shown below:

$$f^l = \texttt{LN1}(h^{l-1}) \tag{6}$$
$$a^l = \texttt{Att}(f^l) \tag{7}$$
$$g^l = \texttt{LN2}(h^{l-1} + a^l) \tag{8}$$
$$m^l = W^l_{proj}\sigma(W^l_{fc}g^l + b^l_{fc}) + b_{proj} \tag{9}$$
$$h^l = h^{l-1} + a^l + m^l \tag{10}$$

The intermediate hidden vector between each layer, h^l, is also sometimes referred to as the *residual stream*. LN1 represents the first LayerNorm (or equivalently RMSNorm for Llama models) that acts just before the attention module and LN2 represents the second LayerNorm just before the MLP module. Att represents the self-attention module in an LLMs whereas the action of a traditional MLP module is written in terms of the individual weight matrices (W_{fc}, W_{proj}). As the vectors computed in the attention and MLP modules get added back to the residual stream at each layer, the residual stream represents a summation of an increasing number of vectors as we go deeper into the model.

F Proof for MEMIT + MPES + NC Objective

First we have that we can write the equation 4 in term of matrix form where we can stack the k^i_0. Specifically, we define $K_0 = [k^1_0 \,|k^2_0\, | \cdots \,|k^P_0]$, $K_1 = [k^1_e \,|k^2_e\, | \cdots \,|k^B_e]$ and $V_1 = [v^1_e \,|v^2_e\, | \cdots \,|v^B_e]$

column wise and instead the L2 norm will become the frobenius norm and we have that

$$\lambda_p \left\|\hat{W}K_0 - W_0K_0\right\|^2_F + \left\|\hat{W}K_1 - V_1\right\|^2_F + \lambda_n \left\|\hat{W} - W_0\right\|^2_F$$

We can differentiate this expression with respect to $\hat{W}$ and let it equal to 0, we get the following

$$\lambda_p\hat{W}K_0K_0^T - \lambda_pW_0K_0K_0^T + \hat{W}K_1K_1^T - V_1K_1^T + \lambda_n\hat{W} - \lambda_nW_0 = 0$$

$$\lambda_p\hat{W}K_0K_0^T + \hat{W}K_1K_1^T + \lambda_n\hat{W} = \lambda_pW_0K_0K_0^T + V_1K_1^T + \lambda_nW_0$$

Since $\hat{W} = W_0 + \Delta$ we have the following

$$\lambda_p(W_0+\Delta)K_0K_0^T + (W_0+\Delta)K_1K_1^T + \lambda_n(W_0+\Delta) = \lambda_pW_0K_0K_0^T + V_1K_1^T + \lambda_nW_0$$

$$\lambda_pW_0K_0K_0^T + \lambda_p\Delta K_0K_0^T + W_0K_1K_1^T + \Delta K_1K_1^T + \lambda_n(W_0+\Delta) = \lambda_pW_0K_0K_0^T + V_1K_1^T + \lambda_nW_0$$

$$\Delta(\lambda_pK_0K_0^T + K_1K_1^T + \lambda_nI) = (V_1 - W_0K_1)K_1^T$$
$$\Delta = (V_1 - W_0K_1)K_1^T(\lambda_pK_0K_0^T + K_1K_1^T + \lambda_nI)^{-1}$$

G Editing Hyperparameters and Hardware Details

Here we present the hyperparameters that we find and used for the MPES, Norm constraint, and MPES + Norm constraint. Tables 6, 7 and 8 show the hyperparameters for CounterFact dataset and tables 9,10 and 11 show the hyperparameters for the zsRE dataset. Additionally, there is one more hyperparameter that requires tuning, as stopping immediately when the target token becomes the most probable may not always be optimal. We define this hyperparameter as the probability cutoff, which determines how many additional steps we take before stopping. Specifically, a cutoff of $+n$ means that we continue for n more steps after the target token first becomes the most probable.

All experiments in this paper are done on NVIDIA A6000, including experiments where editing speeds of different methods are timed.

Method	Model	λ_p	Probability Cut Off
AlphaEdit + MPES	GPT2-XL	20,000	+1
	Llama2-7B	15,000	+0
	Llama3-8B	15,000	+0
MEMIT + MPES	GPT2-XL	20,000	+2
	Llama2-7B	15,000	+1
	Llama3-8B	15,000	+2

Table 6: Hyperparameters for different algorithms with MPES on CouterFact dataset

Method	Model	λ_p	λ_n
Norm Constraint	GPT2-XL	20,000	10
	Llama2-7B	15,000	10
	Llama3-8B	15,000	20

Table 7: Hyperparameters for Norm Constraint on CouterFact dataset

Method	Model	λ_p	λ_n	Probability Cut Off
MPES + NC	GPT2-XL	20,000	10	+3
	Llama2-7B	15,000	10	+2
	Llama3-8B	15,000	20	+1

Table 8: Hyperparameters for MPES + Norm Constraint on CouterFact dataset

H Norm growth Result

In this section we present the observation of norm growth during editing for different methods (including Norm Constraint and MPES + Norm Constraint) with different models. Figures 13 and 14 show the result for different methods for GPT2-XL. Figures 15 and 16 show the result for Llama2-7B. Lastly figures 17 and 18 show the result for Llama3-8B.

METHOD	MODEL	λ_p	λ_n
NORM CONSTRAINT	GPT2-XL	20,000	40
	LLAMA2-7B	15,000	10
	LLAMA3-8B	15,000	20

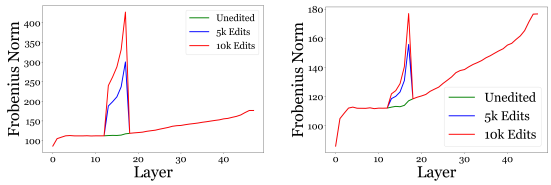
Table 9: Hyperparameters for Norm Constraint on zsRE dataset

METHOD	MODEL	λ_p	PROBABILITY CUT OFF
ALPHAEDIT + MPES	GPT2-XL	20,000	+1
	LLAMA2-7B	15,000	+1
	LLAMA3-8B	15,000	+3
MEMIT + MPES	GPT2-XL	20,000	+5
	LLAMA2-7B	15,000	+1
	LLAMA3-8B	15,000	+4

Table 10: Hyperparameters for different algorithms with MPES on zsRE dataset

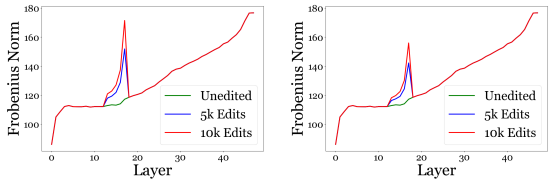
METHOD	MODEL	λ_p	λ_n	PROBABILITY CUT OFF
MPES + NC	GPT2-XL	20,000	40	+4
	LLAMA2-7B	15,000	10	+2
	LLAMA3-8B	15,000	10	+3

Table 11: Hyperparameters for MEMIT+ Norm Constraint on zsRE dataset



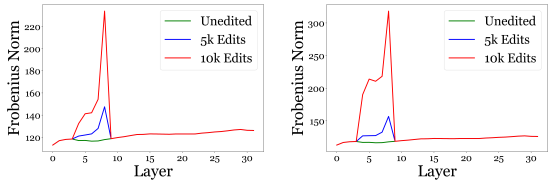
(a) Norm growth of GPT2-XL using Alphaedit (b) Norm growth of GPT2-XL using MEMIT

Figure 13: Norm growth of GPT2-XL using AlphaEdit and MEMIT



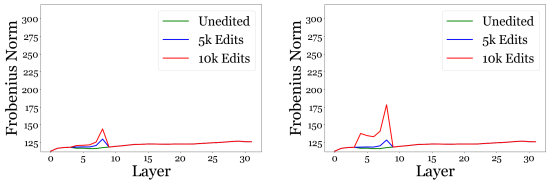
(a) Norm growth of GPT2-XL using NC (b) Norm growth of GPT2-XL using MPES + NC

Figure 14: Norm growth of GPT2-XL using NC and MPES + NC



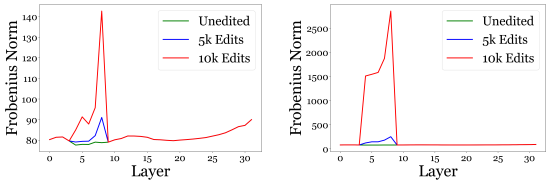
(a) Norm growth of Llama2-7B using Alphaedit (b) Norm growth of Llama2-7B using MEMIT

Figure 15: Norm growth of Llama2-7B using AlphaEdit and MEMIT



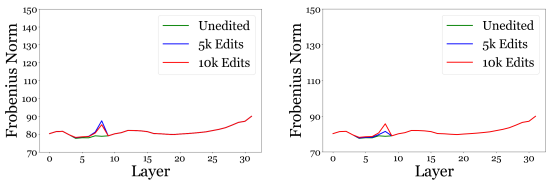
(a) Norm growth of Llama2-7B using NC (b) Norm growth of Llama2-7B using MPES + NC

Figure 16: Norm growth of Llama2-7B using NC and MPES + NC



(a) Norm growth of Llama3-8B using Alphaedit (b) Norm growth of Llama3-8B using MEMIT

Figure 17: Norm growth of Llama3-8B across different methods



(a) Norm growth of Llama3-8B using NC (b) Norm growth of Llama3-8B using MPES + NC

Figure 18: Norm growth of Llama3-8B using NC and MPES + NC

I Editing Performance on zsRE dataset

Tables 12 - 14 show the editing scores for the sequential editing experiments on zsRE.

Model	Method	Edit Score	Paraphrase Score	Neighborhood Score
GPT2-XL	AlphaEdit	42.1	33.61	14.61
	AlphaEdit + MPES	54.99	43.18	18.40
	MEMIT	74.60	61.77	22.40
	MEMIT + MPES	**75.09**	61.58	23.37
	MEMIT + NC	74.51	**61.90**	23.39
	MEMIT + MPES + NC	74.46	61.79	**23.41**

Table 12: Editing performance for GPT2-XL on zsre dataset

Model	Method	Edit Score	Paraphrase Score	Neighborhood Score
Llama2-7B	AlphaEdit	83.77	77.12	41.96
	AlphaEdit + MPES	83.80	77.64	41.97
	MEMIT	79.49	74.29	41.80
	MEMIT + MPES	83.01	77.45	44.64
	MEMIT + NC	88.73	84.05	47.98
	MEMIT + MPES + NC	**89.10**	**84.28**	**48.51**

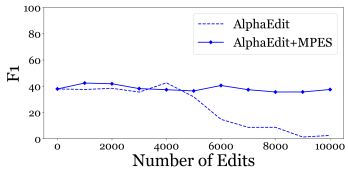
Table 13: Editing performance for Llama2-7B on zsre dataset

Model	Method	Edit Score	Paraphrase Score	Neighborhood Score
Llama3-8B	AlphaEdit	89.27	82.19	45.23
	AlphaEdit + MPES	93.54	85.93	47.32
	MEMIT	96.45	90.30	48.91
	MEMIT + MPES	**96.85**	**90.76**	47.34
	MEMIT + NC	90.40	84.58	49.09
	MEMIT + MPES + NC	93.15	86.19	**49.81**

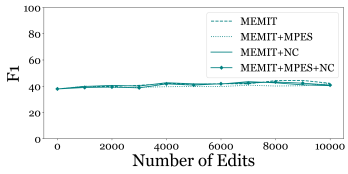
Table 14: Editing performance for Llama3-8B on zsre dataset

J Downstream Performance

Figures 19(a) - 20(b) show the result for the downstream performance for GPT2-XL on both CounterFact and zsRE datasets. Figures 21(a) - 22(b) show the result for Llama2-7B on both datasets. Lastly, figures 23(a) - 24(b) show the result for Llama3-8B on both datasets.

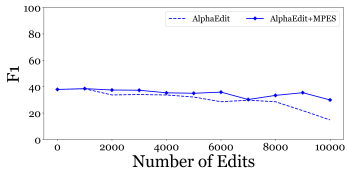


(a) AlphaEdit and MPES

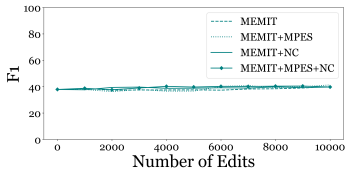


(b) MEMIT, MPES, NC, MPES+NC

Figure 19: Downstream Performance for GPT2-XL using different editing methods with CounterFact dataset

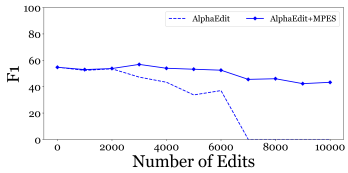


(a) AlphaEdit and MPES

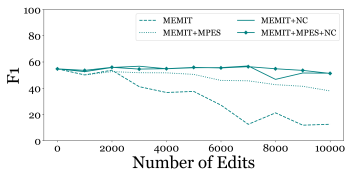


(b) MEMIT, MPES, NC, MPES+NC

Figure 20: Downstream Performance for GPT2-XL using different editing methods with zsRE dataset

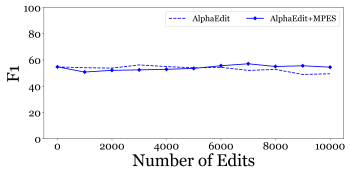


(a) AlphaEdit and MPES

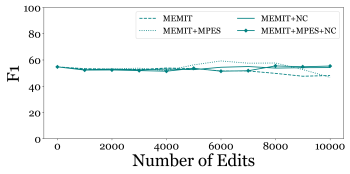


(b) MEMIT, MPES, NC, MPES+NC

Figure 21: Downstream Performance for Llama2-7B using different editing methods with CounterFact dataset

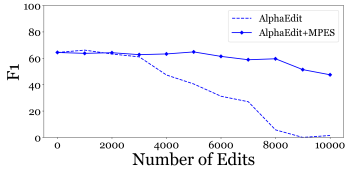


(a) AlphaEdit and MPES

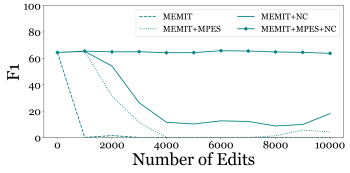


(b) MEMIT, MPES, NC, MPES+NC

Figure 22: Downstream Performance for Llama2-7B using different editing methods with zsRE dataset

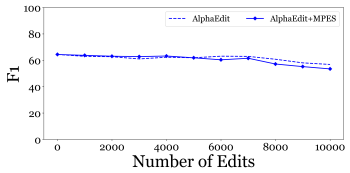


(a) AlphaEdit and MPES

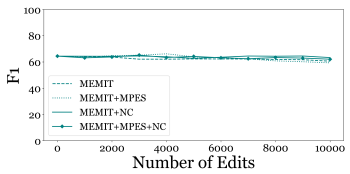


(b) MEMIT, MPES, NC, MPES+NC

Figure 23: Downstream Performance for Llama3-8B using different editing methods with CounterFact dataset



(a) AlphaEdit and MPES



(b) MEMIT, MPES, NC, MPES+NC

Figure 24: Downstream Performance for Llama3-8B using different editing methods with zsRE dataset

K Norm Decrease From MPES

In this section we want to highlight one observation that we have is that while MPES does reduce the norm of the edited layer shown in Figure 25 by quite a lot compared to just MEMIT alone in Figure 1(a). But we still note that the norm growth is still 3 times higher which is the reason why we think that explicit norm control is still needed.

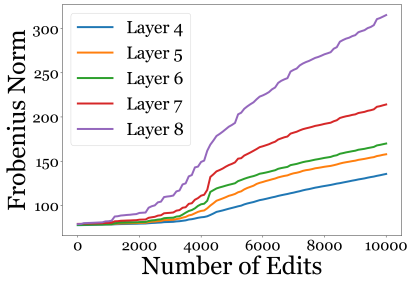
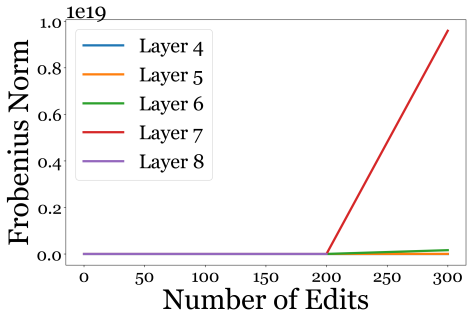


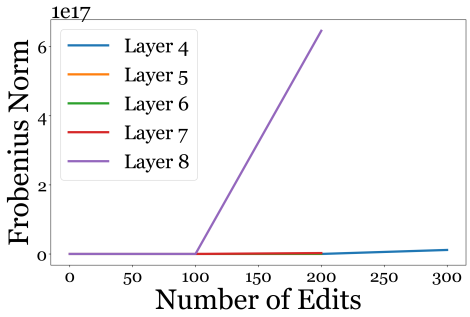
Figure 25: The figure shows the Norm-growth as function of number edits in MEMIT Llama3-8B with MPES.

L Unbound growth of norm in AlphaEdit

Here we provide the result if we remove the norm constraint from the objective function of AlphaEdit. As we can see, once we remove the norm constraint, the norm growth of AlphaEdit just becomes unbound. Note that the reason why the x-axis only goes up to 300 edits is that after that the norm value becomes overflow in Python and it therefore it cannot be plot.



(a) AlphaEdit with Llama2-7B



(b) AlphaEdit with Llama3-8B

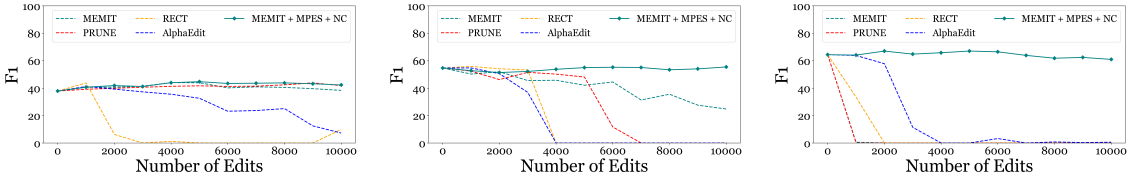
Figure 26: The figures show the unbound Norm-growth as function of number edits in AlphaEdit

M Batch Size Effect

For this section we show the result for different batch size we see that our method MPES + NC is still effective

Method	Edit Score			Paraphrase Score			Neighborhood Score			Overall Score			Generation Entropy		
	GPT2-XL	Llama2-7B	Llama3-8B	GPT2-XL	Llama2-7B	Llama3-8B	GPT2-XL	Llama2-7B	Llama3-8B	GPT2-XL	Llama2-7B	Llama3-8B	GPT2-XL	Llama2-7B	Llama3-8B
MEMIT	91.97	81.73	50.32	77.67	65.58	49.56	58.00	**66.22**	50.59	73.19	70.45	50.15	503.98	522.72	281.88
PRUNE	**92.76**	51.80	48.50	**78.43**	50.15	49.21	57.25	50.17	51.15	73.17	50.70	49.60	505.02	247.88	297.80
RECT	51.01	51.85	48.94	48.92	50.47	49.98	52.57	49.34	50.14	50.79	50.53	49.68	539.58	323.46	192.70
AlphaEdit	91.24	56.92	57.80	73.71	51.38	56.55	56.39	53.73	49.70	70.99	53.92	54.44	**586.77**	494.64	430.37
MEMIT + MPES + NC	92.57	**90.79**	**89.71**	78.19	**79.76**	**80.48**	**60.36**	59.70	**57.63**	**74.70**	**74.44**	**73.31**	510.81	**555.18**	**539.91**

Table 15: Editing performance of our approach when compared to baseline MEMIT, AlphaEdit and MEMIT regularization method such as PRUNE and RECT using batchsize 10.



(a) Downstream Performance for GPT2-XL with batch size 10

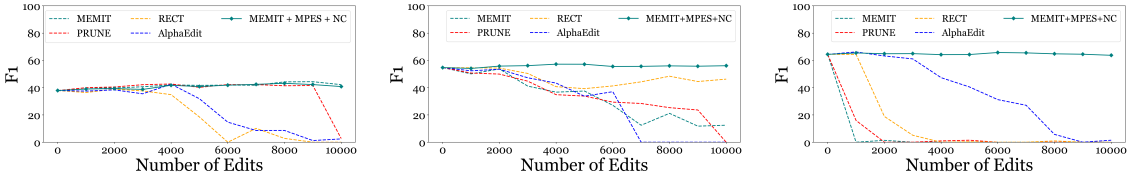
(b) Downstream Performance for Llama2-7B with batch size 10

(c) Downstream Performance for Llama3-8B with batch size 10

Figure 27: Different Downstream Performance for different models with batch size 10

Method	Edit Score			Paraphrase Score			Neighborhood Score			Overall Score			Generation Entropy		
	GPT2-XL	Llama2-7B	Llama3-8B	GPT2-XL	Llama2-7B	Llama3-8B	GPT2-XL	Llama2-7B	Llama3-8B	GPT2-XL	Llama2-7B	Llama3-8B	GPT2-XL	Llama2-7B	Llama3-8B
MEMIT	**94.04**	81.04	49.68	**79.91**	64.67	49.29	57.90	60.95	51.31	74.22	67.86	50.08	517.37	442.59	373.48
PRUNE	61.05	70.80	49.38	58.05	62.11	49.63	50.00	51.86	51.09	55.96	60.60	50.02	579.69	280.83	340.22
RECT	51.40	82.42	63.17	49.83	66.84	56.92	52.17	**67.39**	52.89	51.12	71.54	57.36	409.42	549.35	**588.39**
AlphaEdit	88.58	61.10	72.67	70.33	55.86	63.44	56.04	53.75	52.90	69.20	56.74	61.95	**580.27**	540.92	465.81
MEMIT + MPES + NC	93.35	**92.57**	**88.77**	78.66	**82.64**	**78.19**	**59.84**	60.43	**60.07**	**74.75**	**76.04**	**73.71**	523.49	**560.16**	523.61

Table 16: Editing performance of our approach when compared to baseline MEMIT, AlphaEdit and MEMIT regularization method such as PRUNE and RECT using batchsize 100.



(a) Downstream Performance for GPT2-XL with batch size 100

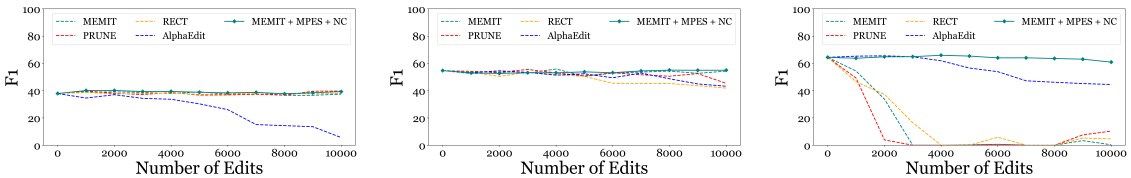
(b) Downstream Performance for Llama2-7B with batch size 100

(c) Downstream Performance for Llama3-8B with batch size 100

Figure 28: Different Downstream Performance for different models with batch size 100

Method	Edit Score			Paraphrase Score			Neighborhood Score			Overall Score			Generation Entropy		
	GPT2-XL	Llama2-7B	Llama3-8B	GPT2-XL	Llama2-7B	Llama3-8B	GPT2-XL	Llama2-7B	Llama3-8B	GPT2-XL	Llama2-7B	Llama3-8B	GPT2-XL	Llama2-7B	Llama3-8B
MEMIT	**93.55**	93.94	74.45	**81.31**	76.83	61.25	59.66	67.64	57.33	75.47	78.03	63.56	558.93	577.69	457.97
PRUNE	92.42	88.58	62.83	81.27	70.43	54.11	58.79	64.64	52.29	74.75	73.25	56.05	548.27	527.05	461.71
RECT	87.49	87.35	57.36	71.94	60.86	52.31	64.9	**73.67**	64.34	73.64	72.37	57.59	**603.91**	566.67	215.44
AlphaEdit	92.61	**96.11**	**91.29**	76.78	**87.27**	**76.88**	56.09	61.63	68.95	72.03	**78.76**	**78.00**	587.19	588.09	593.35
MEMIT + MPES + NC	92.57	93.75	84.29	80.01	80.45	70.51	**61.26**	65.71	**69.48**	**75.71**	78.30	74.19	566.94	**593.12**	**598.86**

Table 17: Editing performance of our approach when compared to baseline MEMIT, AlphaEdit and MEMIT regularization method such as PRUNE and RECT using batchsize 1000.



(a) Downstream Performance for GPT2-XL with batch size 1000

(b) Downstream Performance for Llama2-7B with batch size 1000

(c) Downstream Performance for Llama3-8B with batch size 1000

Figure 29: Different Downstream Performance for different models with batch size 1000

N Evaluation of Downstream Performance

In this paper, we assess model degradation by measuring downstream performance at regular intervals of edits. Our evaluation suite is wide-ranging and consists of the following 6 tasks – sentiment analysis (SST2) (Socher et al., 2013), paraphrase detection (MRPC) (Dolan and Brockett, 2005), natural language inference (NLI, RTE) (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), linguistic acceptability classification (CoLA) (Warstadt et al., 2019), and massive multitask language understanding (MMLU) (Hendrycks et al., 2020).

For each task, we created a subset of 100 examples balanced across all multiple-choice options. The models were evaluated on the tasks above, and the accuracy score was measured every 1000 edits. In order to improve the models' initial performance and achieve meaningful signals, we provided few-shot examples. The few-shot prompt templates used for each task are shown in Figures 30-35.

```
Review : an exhilarating futuristic thriller-noir , minority report twists the best of technology around a gripping story , delivering a riveting , pulse intensifying escapist adventure of the first order
Sentiment : positive

Review : try as i may , i ca n't think of a single good reason to see this movie , even though everyone in my group extemporaneously shouted , ' thank you ! '
Sentiment : negative

Review : the film 's performances are thrilling .
Sentiment : positive

Review : vera 's technical prowess ends up selling his film short ; he smoothes over hard truths even as he uncovers them .
Sentiment : negative

Review : [input]
Sentiment :
```

Figure 30: Few shot prompt template used for SST-2

```
Question: Which expression is equivalent to 4 x 9?
(A) (4x 4) + (4x5)
(B) (4+4) x (4+5)
(C) (4+4)+(4+5)
(D) (4x 4) x (4x5)
Answer: A

Question: A marketing researcher is conducting a survey in a large selling area by contacting a small group of people that is representative of all people in that area. The small, representative group is known as the
(A) population
(B) sample
(C) stratification
(D) universe
Answer: B

Question: A research participant eats half a bowl of M&M candies, and then stops eating. How would a motivation researcher using drive reduction theory explain this participant's behavior?
(A) Humans are instinctively driven to eat sugar and fat when presented to them.
(B) The Yerkes-Dodson law explains that people will eat food when presented to them, but usually in moderate amounts in order to avoid being perceived as selfish.
(C) The primary drive of hunger motivated the person to eat, and then stop when she/he regained homeostasis.
(D) The research participant was satisfying the second step on the hierarchy of needs: Food needs.
Answer: C

Question: In a deductively valid argument
(A) If all the premises are true, the conclusion must be true
(B) The conclusion has already been stated in its premises
(C) If all the premises are true, the conclusion may still be false
(D) Both A and B
Answer: D

Question: [input]
Answer:
```

Figure 31: Few shot prompt template used for MMLU

Are the sentences paraphrases of each other.
Sentence 1: Federal regulators have turned from sour to sweet on a proposed $2.8 billion merger of ice cream giants Nestle Holdings Inc. and Dreyer 's Grand Ice Cream Inc .
Sentence 2: Federal regulators have changed their minds on a proposed $2.8 billion merger of ice cream giants Nestle Holdings and Dreyer 's Grand Ice Cream .
Answer: Yes

Are the sentences paraphrases of each other.
Sentence 1: In the year-ago quarter , the steelmaker recorded a profit of $16.2 million , or 15 cents per share , on sales of $1.14 billion .
Sentence 2: In the second quarter last year , AK Steel reported a profit of $16.2 million , or 15 cents a share .
Answer: No

Are the sentences paraphrases of each other.
Sentence 1: He added : "I 've never heard of more reprehensiblebehaviour by a doctor .
Sentence 2: The Harrisons ' lawyer Paul LiCalsi said : " I ' ve never heard of more reprehensible behaviour by a doctor .
Answer: Yes

Are the sentences paraphrases of each other.
Sentence 1: While dioxin levels in the environment were up last year , they have dropped by 75 percent since the 1970s , said Caswell .
Sentence 2: The Institute said dioxin levels in the environment have fallen by as much as 76 percent since the 1970s .
Answer: No

Are the sentences paraphrases of each other.
Sentence 1: [input 1]
Sentence 2: [input 2]
Answer:

Figure 32: Few shot prompt template used for MRPC

Is this sentence linguistically acceptable?
Sentence: The evidence assembled by the prosecution convinced the jury.
Answer: Yes

Is this sentence linguistically acceptable?
Sentence: I live at the place where Route 150 crosses the Hudson River and my dad lives at it too.
Answer: No

Is this sentence linguistically acceptable?
Sentence: The government's imposition of a fine.
Answer: Yes

Is this sentence linguistically acceptable?
Sentence: Sam gave the ball out of the basket.
Answer: No

Is this sentence linguistically acceptable?
Sentence: [input]
Answer:

Figure 33: Few shot prompt template used for RTE

The town is also home to the Dalai Lama and to more than 10,000 Tibetans living in exile.
Question: The Dalai Lama has been living in exile since 10,000. True or False?
Answer: True

P. Prayong, who like Kevala belongs to the Theravada sect of Buddhism, chose India over other Buddhist majority nations as it is the birthplace of Gautama Buddha.
Question: P. Prayong is a member of Theravada. True or False?
Answer: False

The medical student accused of murdering an erotic masseuse he met on Craigslist is drowning in more than $100,000 in student loan debt and is so broke he can't afford to pay an attorney, according to court papers. Philip Markoff, a 23-year-old suspended Boston University medical school student, owes $130,000 in student loans and does not get money from his parents, leaving him to lean on a taxpayer-funded attorney for his defense, according to a court document in Boston Municipal Court that labels him indigent. Markoff graduated from the State University of New York-Albany and was a second-year medical student at BU.
Question: The medical student Philip Markoff was engaged. True or False?
Answer: True

Traditionally, the Brahui of the Raisani tribe are in charge of the law and order situation through the Pass area. This tribe is still living in present day Balochistan in Pakistan.
Question: The Raisani tribe resides in Pakistan. True or False?
Answer: False

The latest attacks targeted the U-S embassy and a top prosecutor's office in the Uzbek capital.
Question: [input]. True or False?
Answer:

Figure 34: Few shot prompt template used for CoLA

Turkey is unlikely to become involved in, or allow U.S. forces to use Turkish territory in a Middle East war that does not threaten her territory directly. entails the U.S. to use Turkish military bases.
True or False?
Answer: False

Brooklyn Borough Hall featured a Who's Who in New York's literary community during the second annual Brooklyn Book Festival. According to Brooklyn Borough President Marty Markowitz, the borough's zip code 11215 boasts more authors than anywhere else in the country. It appeared to be the case on Sunday. More than 100 authors were featured at the day-long event, including The Basketball Diaries writer Jim Carroll, former M*A*S*H star Mike Farrell, author and illustrator Mo Willems, Jack Kerouac's sometime lover and National Book Critics Circle Award recipient Joyce Johnson and PEN American Center President Francine Prose. entails the The Brooklyn Book Festival is held in Brooklyn Borough every year.
True or False?
Answer: True

NASA's Saturn exploration spacecraft, Cassini , has discovered an atmosphere about the moon Enceladus . This is the first such discovery by Cassini, other than Titan , of the presence of an atmosphere around a Saturn moon. entails the Titan is the fifteenth of Saturn's known satellites.
True or False?
Answer: False

Dr. Eric Goosby, a pioneer in the fight against AIDS, is President Obama's choice to run the American effort to combat the disease globally, the White House announced Monday. The President's Emergency Plan For AIDS Relief, known as Pepfar, was championed by President George W. Bush. It is expected to spend $48 billion over the next five years and is credited with markedly reducing the disease's death rate. Its prevention policy has been controversial because of its emphasis on socially conservative methods. With a new administration and a Democratic majority in the House, organizations seeking prevention choices beyond abstinence and fidelity — including a renewed commitment to distributing condoms — are eager to try to rewrite the guidelines. entails the Pepfar is committed to fighting AIDS.
True or False?
Answer: True

[input]
True or False?
Answer:

Figure 35: Few shot prompt template used for NLI