LOKI: MEMORY-FREE NULL-SPACE CONSTRAINED LIFELONG KNOWLEDGE EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Lifelong knowledge editing aims to efficiently and sequentially update language models over time, as new knowledge becomes available or when the model makes mistakes, while preserving acceptable performance on past knowledge. One unresolved challenge is that existing methods modify a fixed set of layers for all new knowledge samples, reducing flexibility and increasing catastrophic forgetting. Another is requiring access to previous knowledge and extensive pre-processing to obtain data statistics. To address these challenges, we introduce LOKI, a novel approach that uses dynamic layer selection based on the Hilbert-Schmidt Independence Criterion and projects gradient updates onto the null-space of the model weights, bypassing the requirement for previous knowledge access. We show that LOKI achieves superior performance to existing approaches across a wide variety of experiments, achieving up to a 14% improvement in average accuracy.

1 INTRODUCTION

In an ever-evolving environment, a large language model (LLM) needs to be constantly updated with new facts. However, LLM knowledge is typically limited to information present during pre-training, and retraining LLMs is highly costly and unsustainable for continual updates. To this end, *lifelong knowledge editing* (Fang et al., 2025; Wang et al., 2024) seeks to update LLM knowledge in an inexpensive and sustainable manner over time: ideally, lifelong knowledge editing should be performed without significant computational overhead, access to large amounts of past knowledge or memory, and without adding additional parameters. An additional central challenge is a phenomenon called catastrophic forgetting (McCloskey and Cohen, 1989), where the model's performance degrades significantly on previously learned data when exposed to new training data.

To address these challenges, existing works follow a three-phase structure (Figure 1(a)). First, in the *layer selection* phase, a subset of model layers are chosen to be modified during knowledge editing. This reduces the number of parameters to be edited, as remaining layers are frozen. In the *Past Knowledge Consolidation* phase, measures are taken to efficiently preserve the past knowledge already known by the model. For example, Fang et al. (2025) identify a null space of activations evaluated over past knowledge. Finally, in the *New Knowledge Insertion* phase, selected weights are modified in in directions in this null space, thereby preserving predictive performance over past data.

Nevertheless, existing approaches following this structure suffer from several disadvantages: First, most existing works (Meng et al., 2023; Fang et al., 2025; Wang et al., 2024) opt to modify a fixed set of model layers determined before editing. This limits flexibility, as there is no guarantee that all knowledge is stored in the same location in the model. Second, to consolidate past knowledge, existing methods incur computational and memory costs, such as expensive pre-processing on hundreds of thousands of samples (Meng et al., 2023; Fang et al., 2025), or adding additional model parameters (Wang et al., 2024). Furthermore, these costs are compounded if the selected layer is changed or a new LLM is to be edited, further constraining methods to a fixed-layer/model paradigm.

To this end, we propose **L**ayer-Adaptive **O**rthogonal **K**nowledge **I**nsertion (LOKI), a novel lifelong knowledge editing method. Firstly, we propose a dynamic layer selection algorithm based on an information bottleneck criterion, that uses the Hilbert-Schmidt Independence Criterion (HSIC) (Gretton et al., 2005) to adaptively identify the most important layers per edit. Second, to consolidate previous knowledge, we compute the null-space of the LLM weights themselves, without the need for access to examples of previous knowledge or extensive preprocessing. Finally, to insert the new knowledge

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

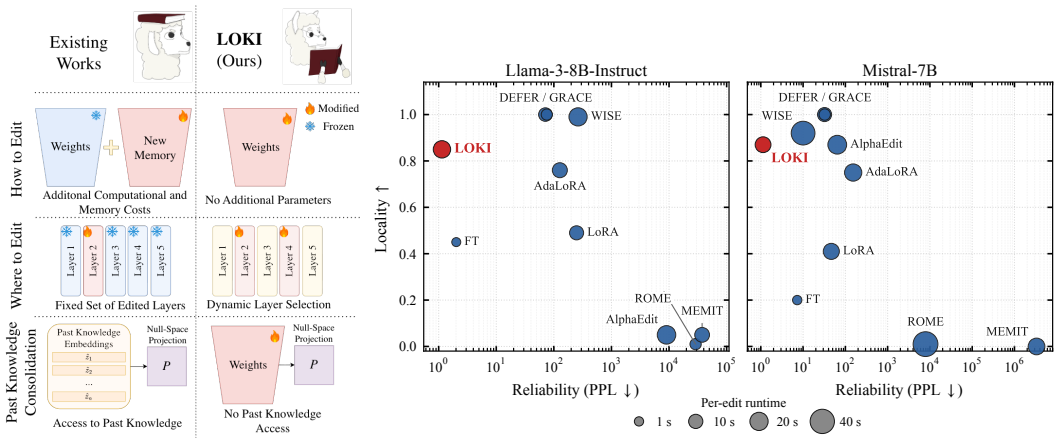


Figure 1: **(a)** Comparison of lifelong knowledge editing between existing works (left) and our method LOKI (right). Existing approaches rely on additional memory/parameters, edit a fixed set of layers, and require access to past knowledge embeddings for null-space projection. In contrast, LOKI edits model weights directly without extra parameters, dynamically selects target layers on a per-sample basis, and consolidates past knowledge via null-space projection without requiring access to previous knowledge samples. **(b)** SelfCheckGPT reliability and locality after $\tau = 600$ sequential edits on Llama-3-8B-Instruct and Mistral-7B for LOKI and 8 competitors. Reliability is target-text perplexity (lower is better) and locality is higher-is-better. Marker area is the mean per-edit runtime, and LOKI is shown in red. LOKI has a significantly lower perplexity compared to other methods, maintaining high-locality on both models.

directly into model weights of the previously selected layers, we perform projected gradient descent while using the null-space projections. Overall, LOKI achieves significantly higher reliability than competitors — up to a 14% increase — with comparable locality and computation time Figure 1(b).

2 Related Work

Knowledge Editing. Existing knowledge editing works face several shortcomings. First, most prior art targets *a fixed set of layers across edits*, presuming a common location for all stored knowledge and limiting flexibility. Meng et al. (2022; 2023) and Fang et al. (2025) rely on a single experiment using interventions on model activations and measuring the resulting change in output, determining the early layers as the target. Hartvigsen et al. (2023); Wang et al. (2024) treat layer selection as a hyperparameter using a validation set. Closer to us, Hu et al. (2024) use a dynamic layer selection scheme for each sample, using the norm of the optimized model activations as criterion, which involves a costly optimization per-layer. In contrast, LOKI uses a dynamic layer selection technique using HSIC which can be computed in a single forward pass for each sample. Secondly, most existing works incur additional memory or computation costs in the past knowledge consolidation phase. Meng et al. (2022; 2023) and Fang et al. (2025) require computing a covariance matrix per edit layer over activations of hundreds of thousands of samples of historical data. Zheng et al. (2023) and Zhong et al. (2023) explicitly store facts or activations in memory. Wang et al. (2024) copy layer weights and only modify the added parameters, determined using activation routing. In all these cases, costs incurred must be repeated if target layers change, further limiting past works to a fixed set of layers across edits. We consolidate past knowledge via a projection to the model weight null-space, thus foregoing the need for access to past data, expensive pre-processing, or added parameters.

Null-space Projection. Null-space projection has emerged as a principled approach to mitigate catastrophic forgetting in continual learning by constraining gradient updates to directions that minimally interfere with previously learned knowledge (Farajtabar et al., 2020). Orthogonal Gradient Descent (OGD) (Farajtabar et al., 2020) projects gradients for new tasks onto the orthogonal complement of gradients from previous tasks, ensuring minimal change to prior predictions. Gradient Projection Memory (GPM) (Saha et al., 2021) extends this idea by computing orthonormal bases of gradient sub-

spaces via Singular Value Decomposition (SVD) on layer activations and storing them in memory for subsequent projection. Fang et al. (2025) use the null-space of model activations for historical data as projection in order to consolidate past knowledge while editing. Our approach differs fundamentally: we project gradients onto the null-space of the *weight matrix* itself, eliminating the need for stored feature statistics or access to previous data while still preserving knowledge learned during training.

Hilbert-Schmidt Independence Criterion. The Hilbert-Schmidt Independence Criterion (HSIC) (Gretton et al., 2005) is a kernel-based measure of statistical dependence, defined as the squared Hilbert-Schmidt norm of the cross-covariance operator in reproducing kernel Hilbert spaces; we provide a formal definition in App. A. HSIC has been applied to train networks without backpropagation via the HSIC Bottleneck (Ma et al., 2020), which optimizes layers by maximizing dependence with outputs while minimizing dependence with inputs, an objective inspired by the Information Bottleneck principle (Tishby and Zaslavsky, 2015). This framework enables efficient compression of learned representations by retaining task-relevant information while discarding redundant input details. We leverage this principle for dynamic layer selection: layers exhibiting high dependence with the output and low dependence with the input are identified as encoding task-critical knowledge and are prioritized. Additionally, we incorporate HSIC as regularization during optimization to encourage information compression and reduce interference with previous knowledge.

3 Background

Notation. Let $f_\theta : \mathcal{X} \to \mathcal{Y}$ be an LLM parameterized by $\theta \in \mathbb{R}^d$, which maps input $X \in \mathbb{R}^{d_1 \times k}$, consisting of $k \in \mathbb{N}$ tokens of dimension d_1, to a probability distribution $y \in \mathbb{R}^{|\mathcal{V}|}$ over the vocabulary of the model with size $|\mathcal{V}|$, denoting the probability of the next token. The LLM consists of L layers, each containing a self-attention module, and a feed-forward network (FFN). As shown in Figure 2(a), the FFN in layer l performs the operation:

$$z^{(l)} = \texttt{FFN}^{(l)}(a^{(l)}) = W^{(l)}_{\texttt{out}}\sigma(W^{(l)}_{\texttt{in}}a^{(l)}), \tag{1}$$

where $a^{(l)} \in \mathbb{R}^{d_1}$ is a token in the output of the attention module, $z^{(l)} \in \mathbb{R}^{d_1}$ is the output, σ is an activation function, and $W^{(l)}_{\texttt{in}} \in \mathbb{R}^{d_2 \times d_1}$, and $W^{(l)}_{\texttt{out}} \in \mathbb{R}^{d_1 \times d_2}$ are up and down projection matrices (the parameters of the FFN). For brevity, we will denote the intermediate activation as $\hat{z}^{(l)} := \sigma(W^{(l)}_{\texttt{in}}a^{(l)})$, meaning $\texttt{FFN}^{(l)}(a^{(l)}) = W^{(l)}_{\texttt{out}}\hat{z}^{(l)}$.

Problem Setting. During lifelong knowledge editing, a pre-trained LLM is to be adapted to sequentially arriving new knowledge. The goal of knowledge editing is to incorporate new knowledge while maintaining its predictive performance over past knowledge. Formally, consider a pre-trained LLM f_θ. A stream of tasks arrives sequentially, each comprising a single edit from the new knowledge dataset $\mathcal{D}_{\texttt{edit}} = \{X_1, X_2, ..., X_\tau\}$: i.e., at task $t \leq \tau$, data point X_t arrives, consisting of tokens partitioned into a subject, relation, and object, i.e., $X_t := (S_t, R_t, O_t)$. Let $\mathcal{L}$ be the negative log-likelihood loss function, and $\mathcal{D}_{\texttt{past}} = \{X_{0,1}, X_{0,2}, ..., X_{0,n}\}$ a reference dataset of past knowledge. The goal of knowledge editing is to adjust weights θ after each task arrival so that (a) loss $\mathcal{L}(O_t|f_\theta(X_t))$ is minimized, while simultaneously (b) $\sum_{j=1}^{n}\mathcal{L}(O_{0,j}|f_\theta(X_{0,j})) + \sum_{i=1}^{t-1}\mathcal{L}(O_i|f_\theta(X_i))$, i.e., the loss values for previous edits and past knowledge, does not increase significantly.

Knowledge Editing Pipeline. Typically, knowledge editing algorithms follow three phases as depicted in Figure 2(b): (i) layer selection, where the layers to be edited are selected, (ii) knowledge consolidation, where measures are taken to prevent the forgetting of past knowledge, and (iii) knowledge insertion, where the new knowledge is inserted into the model. As a concrete example, we summarize prior art closest to our work w.r.t. each of these three phases below.

MEMIT and AlphaEdit. As MEMIT (Meng et al., 2023) and AlphaEdit (Fang et al., 2025) are particularly relevant to our method, we describe their methodology with greater detail in this section.

(i) Layer Selection Meng et al. (2022), preceding MEMIT, choose the editing layer during preprocessing: model inputs are corrupted by Gaussian noise, and subsequently, each token of each layer is denoised, and the effect is measured on the output. They conclude that knowledge in LLMs must be stored in the early layer FFNs, and hence fix the early layers as the same edited layers for all tasks. More specifically, only the $W^{(l)}_{\texttt{out}}$ matrix of each of these layers is modified. MEMIT and AlphaEdit follow the same assumptions and edit exactly the same layers and restricted parameters.

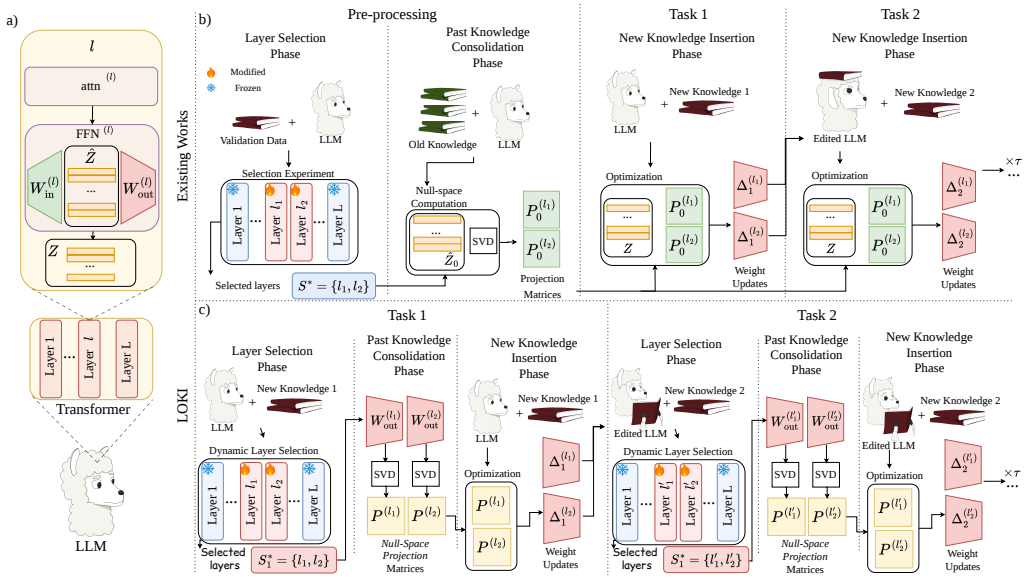


Figure 2: (a) Overview of an LLM. Past methods such as MEMIT/AlphaEdit only edit matrix $W_{\texttt{out}}^{(l)}$ at selected layers. (b) The lifelong knowledge editing pipeline for existing works. Selected layers are fixed across edits, and past knowledge consolidation is an expensive pre-processing step. (c) Our LOKI pipeline. Unlike existing works, LOKI selects important layers on a per-sample basis and does not need access to previous knowledge samples or extensive pre-processing.

(ii) Past Knowledge Consolidation. After choosing layers, both methods consolidate past knowledge via additional pre-processing (see Fig. 2(b)). Concretely, MEMIT summarizes past knowledge by computing the uncentered covariance matrix $C_0^{(l)} = \hat{Z}_0 \hat{Z}_0^\top \in \mathbb{R}^{d_2 \times d_2}$ of the past knowledge embeddings, where $\hat{Z}_0 \in \mathbb{R}^{d_2 \times n}$ are the hidden $\texttt{FFN}^{(l)}$ embeddings corresponding to $\mathcal{D}_{\texttt{past}}$. AlphaEdit takes this a step further by computing the null-space of $\hat{Z}_0$ instead, resulting in a null-space projection matrix $P_0^{(l)}$, as depicted in Figure 2(b):

$$U_0^{(l)} \Sigma_0^{(l)} V_0^{(l)\top} = \texttt{SVD}(\hat{Z}_0 \hat{Z}_0^\top), \qquad \text{and} \qquad P_0^{(l)} = \bar{V}_0^{(l)} \bar{V}_0^{(l)\top}, \tag{2}$$

where $\bar{V}_0^{(l)}$ are the columns of $V_0^{(l)}$ corresponding to zero singular values.[1]

(iii) New Knowledge Insertion. To add new knowledge, MEMIT optimizes an intermediate vector inserted into the model activations. Concretely, they first obtain a desirable vector $z^{(l)*} \in \mathbb{R}^{d_1}$ *for each sample* X_t, by inserting an optimizable vector at the output of a selected layer l's FFN module:

$$z^{(l)*} = \arg\min_z \mathcal{L}(O_t | f_\theta(X_t))_{\texttt{FFN}^{(l)} := z}, \tag{3}$$

where $\texttt{FFN}^{(l)} := z$ denotes replacing the output of $\texttt{FFN}^{(l)}$ with z. The resulting $z^{(l)*}$ vectors are aggregated in $Z^{(l)*} \in \mathbb{R}^{d_1 \times \tau}$ and then used to edit the model weights directly, while using aggregate statistics of past knowledge as a regularizer:

$$W_{\texttt{out}}^{(l)*} = \arg\min_W \|W \hat{Z}^{(l)} - Z^{(l)*}\|_2^2 + \|W \hat{Z}_0^{(l)} - Z_0^{(l)}\|_2^2, \tag{4}$$

where $\hat{Z}^{(l)} \in \mathbb{R}^{d_2 \times \tau}$, $\hat{Z}_0^{(l)} \in \mathbb{R}^{d_2 \times n}$ are the intermediate $\texttt{FFN}^{(l)}$ activations for the new and past knowledge respectively, and $Z_0^{(l)} \in \mathbb{R}^{d_1 \times n}$ is the output of the out projection, i.e., $Z_0^{(l)} = W_{\texttt{out}}^{(l)} \hat{Z}_0^{(l)}$. There is an analytical solution to Equation (4), using the $C_0^{(l)}$ matrix from MEMIT.

AlphaEdit follows the same procedure, but adjusts Equation (4) to include the null-space projection from Equation (2), as well as the features of previous edits:

$$W_{\texttt{out}}^{(l)*} = W_{\texttt{out}}^{(l)} + \arg\min_\Delta \left\{ \|(W_{\texttt{out}}^{(l)} + \Delta P_0^{(l)}) \hat{Z}^{(l)} - Z^{(l)*}\|_2^2 + \|\Delta P_0^{(l)}\|_2^2 + \|\Delta P_0^{(l)} \hat{Z}_p^{(l)}\|_2^2 \right\}, \tag{5}$$

[1]Note that in practice, $\hat{Z}_0 \hat{Z}_0^\top$ may be full rank and have an empty null-space, hence they use singular values below a certain threshold.

where $\hat{Z}_p^{(l)} \in \mathbb{R}^{d_2 \times t-1}$ are the $\texttt{FFN}^{(l)}$ hidden embeddings corresponding to edits of previous tasks $\{1, \ldots, t-1\}$. It is worth noting that the analysis by Meng et al. (2022) with regard to layer selection is limited to a single architecture and dataset. Furthermore, as mentioned in the appendix of Meng et al. (2022), the denoising process is performed on ten consecutive layers at once, yet the effects are attributed to only the middle layer. Finally, Wang et al. (2024) show that, for some architectures, editing the mid-to-late layers is optimal in terms of empirical performance. All of this leads us to hypothesize that static layer selection is limiting the capabilities of knowledge editing algorithms. We propose a dynamic per-sample layer selection algorithm in Section 4. Furthermore, both MEMIT and AlphaEdit require the explicit computation of $\hat{Z}_0$, typically with $n = 100000$. Even though this needs to be done only once, it requires access to past knowledge samples, and it takes significant time and computation: roughly 56 hours of wallclock time to compute these statistics for the Llama3-8B-Instruct model on a single NVIDIA A100 GPU. This process also needs to be repeated for every new LLM. We address these challenges by proposing a substitute for the projection matrix obtained in Equation (2), without requiring accessing or pre-processing previous knowledge (see Section 4).

4 LOKI: Layer-adaptive Orthogonal Knowledge Insertion

We present LOKI (Figure 2(c)), a novel knowledge editing method that seeks to overcome the shortcomings of existing works. We propose a per-sample layer selection algorithm allowing for more flexibility and less interference with past knowledge. Furthermore, we bypass the requirement for access to previous knowledge, extensive pre-processing, additional parameters or memory. We describe our method in detail below.

Layer Selection Phase. We propose to use an HSIC-based information bottleneck criterion to dynamically identify which layers are more important for a given sample. To the best of our knowledge, this is the first attempt at dynamic layer selection for knowledge editing. Let X_t be the input embeddings of an edit sample at time t, and $\hat{Z}^{(l)}$ be the hidden FFN representations of the l-th layer. We define the following information bottleneck criterion $\texttt{HIB}$ for a given layer l and input X_t:

$$\texttt{HIB}(X_t, l) = \lambda_y \texttt{HSIC}(\hat{Z}^{(l)}, Z^{(L)}) - \lambda_x \texttt{HSIC}(X_t, \hat{Z}^{(l)}), \tag{6}$$

where $\texttt{HSIC}$ is the Hilbert-Schmidt Independence Criterion, λ_x, λ_y are scalar hyperparameters, and $Z^{(L)}$ are the FFN output embeddings at the final layer, before the prediction head. $\texttt{HIB}$ measures the flow of information between the input and output embeddings; where if a layer has higher mutual information with the output (i.e., higher $\texttt{HSIC}$) and lower mutual information with the input, intuitively, that means it is holding more important information with regards to that sample, and discarding the noisy information. To identify a subset $S^* \subset \{1, ..., L\}$ of model layers with a fixed size m, we select the top m layers in descending order of $\texttt{HIB}(X_t, l)$. This is equivalent to optimizing the sum of $\texttt{HIB}(X_t, l)$ over subsets of size m, i.e., $S^* = \arg\max_{S \subset [L], |S|=m} \sum_{l \in S} \texttt{HIB}(X_t, l)$. To compute S^*, we evaluate the $\texttt{HIB}$ values for all layers using a single forward pass, and then choose the m layers with the highest values, to be edited for that specific input sample X_t.

Past Knowledge Consolidation Phase. To consolidate previous knowledge, we propose to compute the null-space projection of the selected layers. The intuition behind this choice is that the information about past model knowledge should be ingrained into the weight matrix itself during training. We formalize this intuition in Theorem 4.1, which provides theoretical insight into the relationship of the weight null-space and the null-space of the embeddings. After selecting S^* during the layer selection phase, we can compute the null-space projection of $W_{\texttt{out}}^{(l)}$, where $l \in S^*$, using Singular Value Decomposition (SVD), as follows:

$$U^{(l)} \Sigma^{(l)} V^{(l)\top} = \texttt{SVD}(W_{\texttt{out}}^{(l)}) \qquad \text{and} \qquad P^{(l)} = \bar{V}^{(l)} \bar{V}^{(l)\top}, \qquad \text{for all } l \in S^*, \tag{7}$$

where $P^{(l)} \in \mathbb{R}^{d_2 \times d_2}$ is the projection matrix and $\bar{V}^{(l)} \in \mathbb{R}^{d_2 \times (d_2 - d_1)}$ is the columns of $V^{(l)}$ with zero singular values[2]. In contrast to Equation (2), solving Equation (7) does not require computing $\hat{Z}_0^{(l)}$, and can always be computed by only accessing $W_{\texttt{out}}^{(l)}$, saving on time, computation, and memory.

The following theorem sheds light on our intuition behind choosing the weight null-space, and the relationship it has with the feature null-space as used in Equation (2):

[2]Note that, unlike $\hat{Z}_0^{(l)}$, $W_{\texttt{out}}^{(l)}$ is a low-rank matrix, since $d_2 > d_1$, and hence has a non-empty null-space.

Algorithm 1 LOKI Knowledge Editing

Require: Editing dataset $\mathcal{D}_{\texttt{edit}}$ with τ tasks, LLM f_θ, Num edit layers m, Num steps k, Adjusted Norm constraint η', Learning Rate α

```
for t = 1...τ do
    X_t, O_t ← D_edit[t]                                              {Input arrives}
    Ẑ^(1) ... Ẑ^(L), Z^(L) ← {f_θ(X_t)}                               {Hidden Embeddings}
    scores ← {HIB(X_t, l)  for  l = 1 ... L}
    S* ← top_m(scores)                                                {Selected Layers}
    for l ∈ S* do
        P^(l) ← SVD(W_out^(l))                                        {Equation (7)}
    end for
    for i = 1 ... k do
        L_LOKI ← L(O_t | f_θ(X_t)) − Σ_{l∈S*} HIB(X_t, l)
        for l ∈ S* do
            g ← (∇_{W_out^(l)} L_LOKI) P^(l)                          {Project to Null-Space}
            g ← min(η'/‖g‖_2, 1) · g                                  {Norm Clip}
            W_out^(l) ← W_out^(l) − α · g
        end for
    end for
end for
```

Theorem 4.1. *Let $W_{\texttt{out}}^{(l)} \in \mathbb{R}^{d_1 \times d_2}$ be the down projection matrix for the l-th layer. We assume $\{W_{\texttt{out}}^{(l)}\}_{l=1}^{L}$ have been trained using stochastic gradient descent on samples $\{X_{0,i}\}_{i=1}^{n}$ for n steps, starting from a small initialization $\{W_0^{(l)}\}_{l=1}^{L} \approx \mathbf{0}$. For each $X_{0,i} \in \mathbb{R}^{d_1 \times k}$, arriving at training step i, let $\{\tilde{z}_{0,i}^{(l)}\}_{i=1}^{n} \in \mathbb{R}^{d_2 \times k}$ be the corresponding token hidden embeddings at the input of $W_{\texttt{out}}^{(l)}$ at training step i. Let $\tilde{Z}_0^{(l)} \in \mathbb{R}^{d_2 \times n}$ be the concatenation of all $\{\tilde{z}_{0,i}^{(l)}\}_{i=1}^{n}$. Then $\forall l = 1 \dots L$:* $\texttt{Null}(\tilde{Z}_0^{(l)\top}) \subset \texttt{Null}(W_{\texttt{out}}^{(l)})$.

The proof is in App. B, and follows the fact that the gradient updates to $W_{\texttt{out}}^{(l)}$ are an outer product of the features during training. Note that the above theorem is regarding the model features during training, not after training has completed. Therefore, we must make a simplifying assumption: Let $\hat{z}_{0,i}^{(l)}$ be the token hidden embeddings corresponding to $X_{0,i}$ at the input of $W_{\texttt{out}}^{(l)}$ after training and $\hat{Z}_0^{(l)} \in \mathbb{R}^{d_2 \times n}$ be the concatenation of said embeddings. Assuming $\hat{z}_{0,i}^{(l)} \approx \tilde{z}_{0,i}^{(l)},\ \forall i$, then we have $\texttt{Null}(\hat{Z}_0^{(l)\top}) \subset \texttt{Null}(W_{\texttt{out}}^{(l)})$. The above states that if we assume the model embeddings, and consequently their span, have not changed significantly from training time, the resulting null-spaces are therefore overlapping, and the null-space of the trained model features is also a subset of the weight null-space. While this is a strong assumption, the similarity of model features during and after training is supported by the observations in Neural Tangent Kernel (NTK) literature (Chizat et al., 2019). They observe that during the training of neural networks, the model training enters a lazy regime, and model parameters hardly vary. Furthermore, we empirically validate the effectiveness of projecting the gradients onto the weight null-space in our experiments and ablation studies.

New Knowledge Insertion Phase. Given an input, we propose to optimize the negative log-likelihood of the desired prediction given each input with the added regularization terms, using gradient descent. During optimization, we project the gradients for each layer to the null-space of that layer, using the previously obtained matrices in Equation (7). In addition to the null-space projection, we add the HSIC Information bottleneck introduced in Equation (6) to the optimization objective, described in the following section. Maximizing the HIB for each layer leads to efficient compression of the information learned by the model, reducing interference with past knowledge. Concretely, we add the term $\sum_{l \in S^*} -\texttt{HIB}(X_t, l)$ to the loss function for task t. Furthermore, we restrict the norm of the weight updates for stability, that is, less interference with previous knowledge. Formally, we solve:

$$\min_{\mathbf{\Delta}} \ \mathcal{L}(O_t | f_\theta(X_t)) - \textstyle\sum_{l \in S^*} \texttt{HIB}(X_t, l) \quad \text{s.t. } W_{\texttt{out}}^{(l)} \Delta^{(l)T} = 0, \|\Delta^{(l)}\|_F \leq \eta, \forall \Delta^{(l)} \in \mathbf{\Delta} \tag{8}$$

where $\mathbf{\Delta} = \{\Delta^{(l)} | l \in S^*\}$ is the weight updates for the layers S^*. In contrast to AlphaEdit, we directly optimize the objective, without the need for an intermediate vector (Equation (3)).

As the objective functions are non-convex, we opt to solve Equation (8) using projected gradient descent, where the gradients of the objective with respect to the weights are calculated using back-propagation, then projected using the projection matrices $P^{(l)}$, and finally normalized if needed to

Table 1: Results on the ZsRE dataset (Llama-3-8B-Instruct and Mistral-7B). Metrics include Reliability (Rel. ↑), Generalization (Gen. ↑), and Locality (Loc. ↑) across τ sequential edits, with Avg. representing the mean of the three scores. **Bold** is the best result, and underline is the second-best. LOKI consistently achieves the highest average performance across all edit scales, demonstrating superior scalability and stability compared to existing methods.

Method	$\tau = 1$				$\tau = 10$				$\tau = 100$				$\tau = 1000$			
	Rel. (↑)	Gen. (↑)	Loc. (↑)	Avg. (↑)	Rel. (↑)	Gen. (↑)	Loc. (↑)	Avg. (↑)	Rel. (↑)	Gen. (↑)	Loc. (↑)	Avg. (↑)	Rel. (↑)	Gen. (↑)	Loc. (↑)	Avg. (↑)
Llama-3-8B-Instruct																
FT	**1**	**0.99**	0.95	0.98	0.87	0.83	0.80	0.83	0.84	0.81	0.42	0.69	0.83	0.80	0.11	0.58
LoRA	**1**	**0.99**	0.49	0.83	0.76	0.72	0.32	0.60	0.47	0.45	0.30	0.41	0.36	0.35	0.20	0.31
AdaLoRA	**1**	0.73	0.82	0.85	0.83	0.60	0.57	0.67	0.54	0.49	0.44	0.49	0.36	0.35	0.26	0.32
ROME	0.99	0.96	0.96	0.97	0.61	0.59	0.41	0.54	0.08	0.08	0.02	0.06	0.03	0.03	0.02	0.03
MEMIT	0.99	0.97	0.98	0.98	0.68	0.67	0.73	0.69	0.03	0.03	0.01	0.02	0.00	0.00	0.00	0.00
WiLKE	0.99	0.95	0.95	0.97	0.69	0.66	0.65	0.66	0.02	0.02	0.00	0.01	0.05	0.05	0.01	0.04
DEFER	0.82	0.86	0.68	0.79	0.66	0.67	0.45	0.59	0.33	0.32	**1**	0.55	0.31	0.31	**1**	0.54
GRACE	**1**	0.46	**1**	0.82	**1**	0.42	**1**	0.81	**1**	0.39	**1**	0.8	**1**	0.37	**1**	0.79
SimIE	0.75	0.71	0.79	0.75	0.73	0.69	0.79	0.73	0.72	0.67	0.78	0.72	0.69	0.65	0.75	0.69
WISE	0.99	**0.99**	0.92	0.97	0.77	0.74	**1**	0.84	0.61	0.6	**1**	0.73	0.54	0.53	0.99	0.69
AlphaEdit	0.99	0.88	0.99	0.95	0.92	0.84	0.98	0.92	0.90	0.84	0.94	0.90	0.76	**0.88**	0.65	0.76
LOKI (Ours)	**1**	0.98	0.99	**0.99**	0.94	**0.87**	0.98	**0.93**	**0.93**	**0.86**	0.96	**0.92**	0.91	0.87	0.88	**0.89**
Mistral-7B																
FT	0.99	0.98	0.75	0.91	0.82	0.77	0.22	0.60	0.83	0.77	0.07	0.56	0.69	0.65	0.07	0.47
LoRA	**1**	**1**	0.31	0.77	0.63	0.60	0.21	0.48	0.33	0.32	0.17	0.28	0.15	0.15	0.07	0.12
AdaLoRA	**1**	0.91	0.84	0.92	0.91	0.78	0.62	0.77	0.56	0.51	0.43	0.50	0.50	0.48	0.40	0.46
ROME	0.79	0.77	0.98	0.85	0.58	0.57	0.75	0.63	0.05	0.05	0.02	0.04	0.04	0.04	0.02	0.03
MEMIT	0.81	0.79	0.99	0.86	0.46	0.45	0.61	0.51	0.00	0.00	0.01	0.00	0.04	0.04	0.02	0.03
WiLKE	0.85	0.82	0.98	0.88	0.75	0.85	0.91	0.81	0.05	0.05	0.01	0.04	0.03	0.03	0.01	0.02
DEFER	0.59	0.68	0.79	0.69	0.44	0.43	**1**	0.62	0.40	0.40	**1**	0.60	0.40	0.39	**1**	0.60
GRACE	**1**	0.36	**1**	0.79	**1**	0.15	**1**	0.72	**1**	0.15	**1**	0.72	**1**	0.02	**1**	0.67
SimIE	0.62	0.59	0.94	0.72	0.61	0.58	0.94	0.71	0.60	0.58	0.93	0.70	0.59	0.56	0.92	0.69
WISE	0.96	0.93	**1**	0.96	0.88	0.84	0.99	0.91	0.82	0.76	0.99	0.86	0.65	0.63	0.99	0.76
AlphaEdit	0.86	0.78	0.99	0.88	0.82	0.76	0.99	0.86	0.80	0.75	0.96	0.84	0.80	0.74	0.75	0.76
LOKI (Ours)	**1**	0.96	0.98	**0.98**	0.96	**0.91**	0.97	**0.95**	0.95	**0.88**	0.93	**0.92**	0.92	**0.86**	0.82	**0.87**

preserve the norm constraint. We present a detailed pseudocode of LOKI in Algorithm 1, and refer the reader to App. C for additional implementation details and runtime analysis.

5 Experiments

We provide a detailed description of our experimental setup in App. D; we briefly summarize it here.

Baselines. We compare to SimIE (Guo et al., 2025), AlphaEdit (Fang et al., 2025), ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), and WiLKE (Hu et al., 2024) as baselines that modify model weights directly without the addition of new parameters. Additionally, we compare to WISE (Wang et al., 2024), GRACE (Hartvigsen et al., 2023), and DEFER (Mitchell et al., 2022; Hartvigsen et al., 2023), as baselines which utilize external memory, codebooks, or parameters. Finally, we include Fine-Tuning (FT) following (Zhang et al., 2024), and the parameter-efficient fine-tuning (PEFT) baselines LoRA (Hu et al., 2022) and AdaLoRA (Zhang et al., 2023).

Architectures, Datasets & Metrics. Following previous works (Wang et al., 2024), we utilize three open-source models: Llama-3-8B-Instruct (Grattafiori et al., 2024), Mistral-7B (Jiang et al., 2023), and GPT-J-6B (Wang and Komatsuzaki, 2021). Following Wang et al. (2024), we evaluate LOKI on the *question answering* dataset ZsRE (Levy et al., 2017), the hallucination correction dataset SelfCheckGPT (Manakul et al., 2023), and finally, to measure out of distribution edit effectiveness, the Temporal dataset (Hewitt et al., 2024). We measure the effectiveness of editing algorithms using three main metrics: Reliability, Generalization, and Locality. The concrete score reported for each metric is benchmark-specific. Full definitions for datasets and metrics are given in App. D.

Main Editing Results. We present results for ZsRE, SelfCheckGPT, and Temporal in Tables 1, 2, and 3, respectively. We report deviations across independent runs for the same settings in App. E. Overall, LOKI outperforms existing methods in most scenarios. In Table 1, LOKI achieves top average performance across all scenarios, with a particularly stark performance gap at $\tau = 1000$ on both models, achieving up to 14% improvement in average performance over the runner-up. While some methods perform well in only one or two metrics for each experiment, LOKI is able to balance all three consistently. In Table 2, LOKI is able to consistently achieve the best reliability score (up to 84% improvement over the runner-up for $\tau = 600$) and remain close to the best locality scores (less than 15% degradation). It is worth noting that methods with higher locality scores suffer from a minimum 47% degradation in perplexity for multiple edits. Similarly, in Table 3, LOKI achieves the best or second-best Reliability and Generalization score, while remaining competitive in Locality (less than 20% degradation), attaining the second-best average score.

Table 2: Performance comparison on the SelfCheckGPT hallucination correction task. Metrics include Reliability (Rel. ↓), as target-text perplexity (PPL) and Locality (Loc. ↑) across τ sequential edits; we omit generalization because paraphrase prompts are unavailable for this task. **Bold** is the best result, and underline is the second-best. LOKI demonstrates superior Rel. and competitive Loc. throughout the editing process.

Method	Llama-3-8B-Instruct								Mistral-7B							
	$\tau = 1$		$\tau = 10$		$\tau = 100$		$\tau = 600$		$\tau = 1$		$\tau = 10$		$\tau = 100$		$\tau = 600$	
	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)
FT	1.01	0.95	1.12	0.84	1.44	0.65	2.02	0.45	1.02	0.77	1.41	0.44	1.94	0.22	7.26	0.20
LoRA	**1.00**	0.82	3.52	0.69	6.55e1	0.55	2.48e2	0.49	**1.00**	0.70	2.61	0.61	1.58e1	0.54	4.65e1	0.41
AdaLoRA	1.05	0.98	5.03	0.93	3.41e2	0.85	1.27e2	0.76	1.01	0.97	2.36	0.92	3.19e1	0.84	1.53e2	0.75
ROME	1.39	0.97	4.74e1	0.61	5.00e4	0.02	2.89e4	0.01	1.98	0.99	2.58	0.91	6.77e3	0.03	7.87e3	0.01
MEMIT	1.32	0.99	1.84e3	0.93	6.25e10	0.01	3.75e4	0.05	1.59	**1**	3.13e2	0.93	9.03e4	0.00	3.33e6	0.00
DEFER	2.34e1	0.80	6.86e1	**1**	7.12e1	**1**	7.12e1	**1**	1.76e1	0.92	3.22e1	0.99	3.22e1	**1**	3.22e1	**1**
GRACE	1.40	**1**	7.10e1	**1**	7.17e1	**1**	7.54e1	**1**	1.92	**1**	3.21e1	**1**	3.22e1	**1**	3.25e1	**1**
WISE	1.44	0.97	3.47	0.97	3.54	0.99	2.63e2	0.99	1.08	**1**	1.43e1	0.98	1.74	0.96	10.0	0.92
AlphaEdit	1.29	**1**	1.49	0.99	5.04	0.96	9.04e3	0.05	1.71	**1**	1.88	0.99	2.96	0.98	6.43e1	0.87
LOKI (Ours)	1.01	0.98	**1.01**	0.98	**1.03**	0.94	**1.14**	0.85	1.01	0.99	**1.01**	0.99	**1.04**	0.94	**1.13**	0.87

Method	$\tau = 10$				$\tau = 75$			
	Rel. (↑)	Gen. (↑)	Loc. (↑)	Avg. (↑)	Rel. (↑)	Gen. (↑)	Loc. (↑)	Avg. (↑)
w/o Editing	0.56	0.21	—	—	0.56	0.21	—	—
FT	0.96	0.35	0.69	0.67	0.93	0.32	0.62	0.62
LoRA	0.13	0.00	0.07	0.07	0.10	0.00	0.04	0.05
AdaLoRA	0.38	0.17	0.24	0.26	0.34	0.20	0.31	0.28
ROME	0.10	0.00	0.06	0.05	0.02	0.00	0.02	0.01
MEMIT	0.82	0.28	0.99	0.70	0.43	0.14	0.49	0.35
DEFER	0.66	0.29	0.81	0.59	0.55	0.21	0.95	0.57
GRACE	0.59	0.22	**1**	0.60	0.56	0.22	**1**	0.59
WISE	0.99	**0.36**	0.93	**0.76**	**0.98**	**0.38**	0.99	**0.78**
AlphaEdit	0.90	0.27	0.99	0.72	0.87	0.28	0.97	0.71
LOKI (Ours)	**1**	**0.36**	0.82	0.73	**0.98**	0.36	0.81	0.72

Table 3: OOD performance on the Temporal dataset (GPT-J-6B). Metrics include Reliability (Rel. ↑), OOD Generalization (Gen. ↑), and Locality (Loc. ↑) across τ sequential edits. **Bold** is the best result, and underline is the second-best. LOKI maintains high Rel., and Gen., and competitive Loc. across the editing sequence.

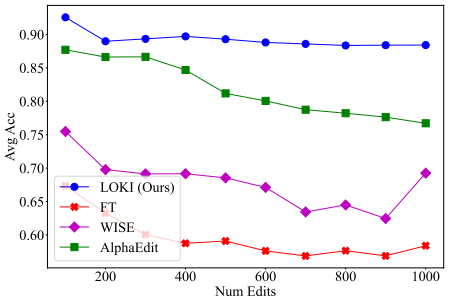


Figure 3: Average accuracy of various methods every 100 edits for 1000 sequential edits on the Llama-3-8B-Instruct and ZsRE dataset. LOKI suffers the least performance degradation over time across competitors.

Runtime Efficiency. We report per-edit runtime in Table 6 (App. C), for every dataset and backbone. Averaged over the five settings and excluding preprocessing, LOKI (15.0s) is faster than ROME (17.3s), SimIE (27.6s), and the two strongest baselines across our benchmarks, AlphaEdit (16.1s) and WISE (23.5s). Accounting for preprocessing makes AlphaEdit 1.9× slower than LOKI over 600 edits on SelfCheckGPT with Llama-3-8B-Instruct.

Evolution over Tasks and Layer Selection Frequency. It is also of interest to measure performance across tasks in regular intervals. In Figure 3, we visualize the average performance of LOKI, WISE, AlphaEdit and FT every 100 edits across 1000 sequential edits, on the ZsRE dataset and the Llama-3 model. LOKI maintains superior accuracy over time and suffers less deterioration compared to competitors. We also study the frequency with which different layers are selected using LOKI during editing. Figure 5 shows layer choice frequencies for Llama-3 across 1000 sequential edits on the ZsRE dataset, where each layer's selection rate is shown as a percentage; layers used by AlphaEdit/MEMIT are depicted in red for reference. We observe that LOKI prefers to edit the later layers quite heavily and sometimes chooses the mid-to-late layers.

Null-space Overlap and Feature Preservation. We empirically study the overlap between the weight null-space and the feature null-space as in AlphaEdit (Fang et al., 2025). We measure this overlap using the Grassmannian distance between subspaces, and comparing the overlap between the weight null-space and the feature null-space, to that of a random subspace and the feature null-space. In Table 4, we observe that the weight null-space significantly overlaps with the feature null-space, validating our claims in Theorem 4.1. We also conduct a qualitative study into the model features resulting from LOKI. In Figure 4, we visualize the last token of the Llama-3 model embeddings for the ZsRE dataset before and after 1000 sequential edits using TSNE. We see that similar to AlphaEdit, LOKI is able to avoid the heavy shift in model features observed using FT, without having to compute

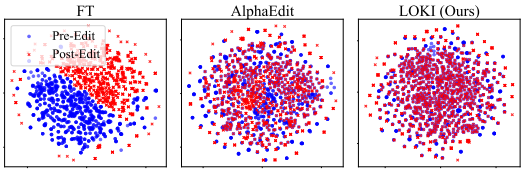


Figure 4: TSNE visualization of Llama-3-8B-Instruct final layer embeddings before and after 1000 sequential edits on the ZsRE dataset. LOKI manages to avoid a significant shift in model embeddings without access to previous knowledge.

Layer		4	5	6	7	8
Llama-3	W-Null (Ours)	**2.42**	**3.19**	**2.56**	**2.40**	**2.42**
	Random	4.28	5.01	4.18	3.69	4.82
Mistral-7B	W-Null (Ours)	**2.88**	**2.85**	**2.56**	**2.41**	**2.79**
	Random	4.68	4.43	4.10	3.59	4.33

Table 4: Grassmannian distance between the null-space of the weights (our proposed method) and the feature null-space (as used by AlphaEdit) compared to a random subspace of the same dimension. The weight null-space has clearly lower distance to the feature null-space.

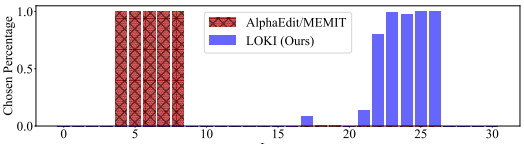


Figure 5: Visualization of the layer selection frequency for Llama-3 during 1000 sequential edits. The horizontal axis represents layers, and the vertical axis shows what percentage of edits chose that layer. Unlike AlphaEdit/MEMIT, LOKI chooses editing layers per-sample, and prefers the mid-to-late layers.

Reg.	Layer	Null	Rel. (↑)	Gen. (↑)	Loc. (↑)	Avg. (↑)
			0.66	0.62	0.2	0.5
		✓	0.78	0.63	0.85	0.76
	✓		0.82	0.79	0.41	0.68
	✓	✓	**0.91**	**0.87**	0.87	0.88
✓			0.71	0.67	0.26	0.55
✓		✓	0.81	0.65	0.86	0.77
✓	✓		0.83	0.82	0.49	0.71
✓	✓	✓	**0.91**	**0.87**	**0.88**	**0.89**

Table 5: Ablation study of LOKI components on the ZsRE dataset ($\tau = 1,000$, Llama). Metrics include Reliability (Rel. ↑), Generalization (Gen. ↑), Locality (Loc. ↑), and their mean (Avg. ↑). **Bold** is the best result. The full configuration achieves the highest scores.

the feature null-space (Equation (2)) as AlphaEdit does. This further validates our intuition about the effectiveness of weight null-space projection for preserving past knowledge.

Ablation Study. We perform an ablation study of all major components of LOKI: the weight null-space projection (Null), the dynamic layer selection (Layer), and the added HSIC bottleneck regularization loss term (Reg.). When null-space projection is absent, we don't project the loss gradients during optimization. When the layer selection is absent, we always optimize the same layers as AlphaEdit, and when the regularization term is absent, we do not add it to the optimization loss function. Table 5 shows the results on the ZsRE dataset for 1000 sequential edits on the Llama-3-8B-Instruct architecture. We observe that all three components have an overall positive effect on average performance when added, with the effect of the weight null-space projection being especially high, followed by the layer selection, and finally a marginal improvement with the HSIC regularization term. This further validates our intuition about the weight null-space projection, as well as the design of our layer selection criterion. We experiment further on the effects of layer selection in App. F.

6 Conclusion

The major challenge in lifelong knowledge editing is to edit knowledge efficiently and sequentially, without relying on external memory, additional parameters, or extensive preprocessing. We attempt to address this challenge by proposing LOKI, a novel knowledge editing method that directly modifies model weights without auxiliary parameters. LOKI performs per-sample layer selection using an information bottleneck criterion and utilizes the model weight null-space to avoid requiring access to previous knowledge samples and features. We empirically validate the performance of LOKI in a variety of scenarios, observing up to a 14% increase in performance compared to existing methods. Additionally, we perform exploratory experiments and ablation studies, validating our intuitions behind our design choices. Finally, we discuss limitations and future work in App. G.

AI USE STATEMENT

AI assistants were used in the paper to assist with writing style and code development. They were not used in any significant way to generate ideas, methodology or design experiments. Illustrations in the figures are hand drawn.

ETHICS STATEMENT

In this work, we present LOKI, a novel knowledge editing method that aims to facilitate the incorporation of new knowledge into LLMs. As LLMs see widespread use in society, knowledge editing in LLMs can be beneficial in a variety of aspects: fixing model mistakes (Balachandran et al., 2022), correcting hallucinations (Ji et al., 2023), or removing bias (Ferrara, 2023). However, as with any algorithm capable of modifying LLM behaviors, knowledge editing methods can be misused. For example, models can be edited to encourage toxic behavior, generate misinformation, or use inappropriate language. These risks are not exclusive to LOKI, and we believe the behavior and usage of LLMs should be sufficiently regulated to prevent misuse, including with knowledge editing. Overall, we believe that the benefits of LOKI are substantial enough to outweigh potential risks and that LOKI does not raise any new or exclusive ethical concerns.

REPRODUCABILITY STATEMENT

We describe our algorithm, datasets, experiments and hyperparameters used in detail in Section 4, Section 5, and Section D. We base our implementation on the EasyEdit framework (Wang et al., 2023), available at `https://github.com/zjunlp/EasyEdit`, the code for our method is available in the supplementary material under the LOKI folder(s) in the `easyeditor/models`, and `hparams` directories, and the code to run our experiments can be found in `run_zsre_llama2.py` and `run_wise_editing.py` files in the `examples` folder.

REFERENCES

Vidhisha Balachandran, Hannaneh Hajishirzi, William Cohen, and Yulia Tsvetkov. Correcting diverse factual errors in abstractive summarization via post-editing and language model infilling. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9818–9830, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained model editing for language models. In *ICLR*, 2025.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.

Emilio Ferrara. Should chatgpt be biased? challenges and risks of bias in large language models. *arXiv preprint arXiv:2304.03738*, 2023.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In *International Conference on Algorithmic Learning Theory*, pages 63–77. Springer, 2005.

Yaming Guo, Siyang Guo, Hengshu Zhu, and Ying Sun. Towards lifelong model editing via simulating ideal editor. In *Forty-second International Conference on Machine Learning*, 2025.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36:47934–47959, 2023.

John Hewitt, Sarah Chen, Lanruo Lora Xie, Edward Adams, Percy Liang, and Christopher D Manning. Model editing with canonical examples. *arXiv preprint arXiv:2402.06155*, 2024.

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3476–3503, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, 2017.

Wan-Duo Kurt Ma, J. P. Lewis, and W. Bastiaan Kleijn. The HSIC bottleneck: Deep learning without back-propagation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5085–5092, 2020.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, 2023.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

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, 2022.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

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

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop*, pages 1–5. IEEE, 2015.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, et al. Easyedit: An easy-to-use knowledge editing framework for large language models. *arXiv preprint arXiv:2308.07269*, 2023.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797, 2024.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, 2023.

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

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? *arXiv preprint arXiv:2305.12740*, 2023.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=0hTPJBnncc.

A HSIC DEFINITION

Let $\mathcal{F}$ and $\mathcal{G}$ be reproducing kernel Hilbert spaces (RKHSs) on domains $\mathcal{X}$ and $\mathcal{Y}$ with associated kernels $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. The Hilbert-Schmidt Independence Criterion (HSIC) between random variables X and Y with joint distribution p_{XY} is defined as the squared Hilbert-Schmidt norm of the cross-covariance operator C_{XY}:

$$\texttt{HSIC}(X, Y) := \|C_{XY}\|_{HS}^2 \tag{9}$$

Given m i.i.d. samples $\{(x_i, y_i)\}_{i=1}^m$ from p_{XY}, the empirical estimator is:

$$\widehat{\texttt{HSIC}}(X, Y) = \frac{1}{(m-1)^2}\text{tr}(KHLH) \tag{10}$$

where $K_{ij} = k(x_i, x_j)$, $L_{ij} = l(y_i, y_j)$ are the Gram matrices, and $H = I - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$ is the centering matrix. With universal kernels (e.g., Gaussian), $\texttt{HSIC}(X, Y) = 0$ if and only if X and Y are statistically independent (Gretton et al., 2005).

B PROOFS

Theorem 4.1. *Let $W_{\texttt{out}}^{(l)} \in \mathbb{R}^{d_1 \times d_2}$ be the down projection matrix for the l-th layer. We assume $\{W_{\texttt{out}}^{(l)}\}_{l=1}^L$ have been trained using stochastic gradient descent on samples $\{X_{0,i}\}_{i=1}^n$ for n steps, starting from a small initialization $\{W_0^{(l)}\}_{l=1}^L \approx \mathbf{0}$. For each $X_{0,i} \in \mathbb{R}^{d_1 \times k}$, arriving at training step i, let $\{\tilde{z}_{0,i}^{(l)}\}_{i=1}^n \in \mathbb{R}^{d_2 \times k}$ be the corresponding token hidden embeddings at the input of $W_{\texttt{out}}^{(l)}$ at training step i. Let $\tilde{Z}_0^{(l)} \in \mathbb{R}^{d_2 \times n}$ be the concatenation of all $\{\tilde{z}_{0,i}^{(l)}\}_{i=1}^n$. Then $\forall l = 1 \dots L$: $\texttt{Null}(\tilde{Z}_0^{(l)\top}) \subset \texttt{Null}(W_{\texttt{out}}^{(l)})$.*

Proof. For the sake of brevity, we will omit the l superscript and 0 subscript. Let $\hat{z}_{i,1}, ..., \hat{z}_{i,k}$ be the vectors corresponding to the i-th sample. Let $z_{i,1}, ..., z_{i,k}$ be the corresponding output tokens of $W_{\texttt{out}}^{(l)}$. Assuming loss function $\mathcal{L}_i$, and learning rate α_i, the gradient update to $W_{\texttt{out}}$ is as follows due to the chain rule, with a slight abuse of notation:

$$\frac{\partial \mathcal{L}_i}{\partial W_{\texttt{out}}} = \sum_{j=1}^{k_i} \frac{\partial \mathcal{L}_i}{\partial z_{i,j}} \cdot \frac{\partial z_{i,j}}{\partial W_{\texttt{out}}} = \sum_{j=1}^{k_i} \frac{\partial \mathcal{L}_i}{\partial z_{i,j}} \cdot \frac{\partial W_{\texttt{out}}^{(l)} \hat{z}_{i,j}}{\partial W_{\texttt{out}}^{(l)}} = \sum_{j=1}^{k_i} \frac{\partial \mathcal{L}_i}{\partial z_{i,j}} \hat{z}_{i,j}^\top \tag{11}$$

Therefore, we can write $W_{\texttt{out}}$ as the culmination of all the gradient updates $1, ..., n$, as

$$W_{\texttt{out}} = W_0 + \sum_{i=1}^{n} \alpha_i \cdot \sum_{j=1}^{k_i} \frac{\partial \mathcal{L}_i}{\partial z_{i,j}} \hat{z}_{i,j}^\top = W_0 + \sum_{i=1}^{n} \sum_{j=1}^{k_i} \alpha_i \cdot \frac{\partial \mathcal{L}_i}{\partial z_{i,j}} \hat{z}_{i,j}^\top \tag{12}$$

Where W_0 is the initialization. We can take the RHS to be a single $n \times k$ sum, and write it in matrix form:

$$W_{\texttt{out}} = W_0 + (\partial Z) A \hat{Z}^\top \tag{13}$$

where $A \in \mathbb{R}^{s \times s}$ is a diagonal matrix, with elements α_i repeated k_i times each with $s := \sum k_i$, $\partial Z \in \mathbb{R}^{d_1 \times s}$ is a matrix with $\frac{\partial \mathcal{L}_i}{\partial z_{i,j}}$ as its columns, and $\hat{Z} \in \mathbb{R}^{d_2 \times s}$ is a matrix with $\hat{z}_{i,j}$ as its columns. Now assuming the initialization W_0 is small or zero, i.e., $W_0 \approx 0$, we can write:

$$W_{\texttt{out}} = (\partial Z) A \hat{Z}^\top$$

which completes the proof, since $\hat{Z}^\top$ is the rightmost term of the product, and any vector $\mathbf{v}$ such that $\hat{Z}^\top \mathbf{v} = \mathbf{0}$ (i.e., $\mathbf{v} \in \texttt{Null}(\hat{Z}^\top)$) must also satisfy $W_{\texttt{out}}\mathbf{v} = (\partial Z) A (\hat{Z}^\top \mathbf{v}) = \mathbf{0}$. Therefore, the null space of $\hat{Z}^\top$ is necessarily contained within the null space of $W_{\texttt{out}}$, satisfying $\texttt{Null}(\hat{Z}^\top) \subset \texttt{Null}(W_{\texttt{out}})$. Note that the small initialization assumption, $W_0 \approx 0$, is standard practice when training deep neural networks (Glorot and Bengio, 2010).

□

C Implementation Details

We base our implementation on the EasyEdit framework (Wang et al., 2023), available at `https://github.com/zjunlp/EasyEdit`, the code for our method is available in the supplementary material under the LOKI folder(s) in the `easyeditor/models`, and `hparams` directories, and the code to run our experiments can be found in `run_zsre_llama2.py` and `run_wise_editing.py` files in the `examples` folder. We will make this code public upon acceptance. All experiments are executed on NVIDIA H200 GPUs using three model architectures: Llama-3-8B-Instruct, Mistral-7B-v0.1, and GPT-J-6B. For all experiments, we perform edits sequentially with a batch size of 1, ensuring the model is instantly updated and corrected after each input following the lifelong model editing setting (Hartvigsen et al., 2023; Wang et al., 2024). We conduct an additional experiment where we increase batch size in App. E.

To mitigate the variance introduced by individual edit samples in experiments involving a small number of sequential edits, we report results averaged over multiple independent runs. In Table 1, the total number of sequential edits τ increases from 1 to 1,000 for the ZsRE dataset, and from 1 to 600 for the SelfCheckGPT dataset in Table 2. For experiments involving a small number of edits (typically from $\tau = 1$ up to $\tau = 100$), we report results averaged over multiple runs using distinct edit samples. For example, results for $\tau = 1$ on the ZsRE dataset are averaged over 1,000 independent experiments, each using a different edit sample. Similarly, for $\tau = 10$ on the SelfCheckGPT dataset, we average results over 60 runs with distinct edit samples. For the Temporal dataset (Tables 3 and 14), results for $\tau = 10$ are averaged over 7 independent experiments. These averaging procedures are consistent across the main results and the extended results in App. E.

C.1 Edit time comparison

Table 6 reports the execution time of each editing algorithm across all models and datasets. Averaged over the five dataset/backbone settings, LOKI takes 15.0 seconds per edit, outperforming AlphaEdit (16.1s), ROME (17.3s), WISE (23.5s) and SimIE (27.6s) in speed, but slower than the remaining baselines, all of which perform worse in our benchmarks. Per-edit time is driven by the number of optimization steps, the number of edited layers, and the input sequence length rather than by the dataset itself, so every method slows down on datasets with longer inputs (the Wikipedia-style prompts of SelfCheckGPT, the long targets of Temporal). The relative cost bands are broadly preserved, though the exact ranking shifts with input length. The runtimes reported for MEMIT and AlphaEdit **do not** include the amortized one-time cost of gathering past-knowledge covariance statistics (and, for AlphaEdit, of building null-space projections P on top of them), so fair comparison would shift both methods much slower once preprocessing is counted.

Table 7 gives estimated wall-clock for that preprocessing alone—Wikipedia, 100k samples, Llama-3-8B-Instruct—on H200 and A100: across all 32 layers the EasyEdit (Wang et al., 2023) pipeline we use totals about $\sim$25 h (H200) and $\sim$56 h (A100), with a layer-wise breakdown in the table. There, statistics are collected *separately for each target layer* (the corpus is iterated per layer), each pass performing a *partial forward* from input embeddings through the transformer only as far as the rewrite module for that layer; the forward ends right after that module so the network tail is not executed, yet deeper rewrite modules still require more compute per batch than earlier ones. AlphaEdit repeats this covariance accumulation and adds the per-layer projection P, whose incremental cost is noted in the table caption. Overall, LOKI avoids this costly preprocessing: it does not iterate a large corpus to build layer-wise covariance or feature null-spaces from past activations, and instead obtains its projections directly from the model weights at edit time. When this preprocessing is amortized across 600 edits on SelfCheckGPT with Llama-3-8B-Instruct, both methods remain substantially slower than LOKI: AlphaEdit is $1.93\times$/$9.73\times$ slower on H200 (only 5 edited layers/all 32) and $2.95\times$/$20.53\times$ slower on A100, while MEMIT is $1.37\times$/$9.08\times$ slower on H200 and $2.38\times$/$19.77\times$ slower on A100.

C.2 Implementation of LOKI

LOKI performs knowledge editing through a multi-stage process. First, it dynamically selects critical layers using Hilbert-Schmidt Independence Criterion (HSIC) to identify layers with maximal information flow between input and output activations. The method then computes null-space projection matrices for each selected layer directly from the weight matrices via singular value

Table 6: Per-edit runtime by dataset and backbone (seconds per edit, mean $\pm$ std over N per-edit measurements, with the model restored to its base state between edits; $N = 300$ for ZsRE, $N = 600$ for SelfCheckGPT, and $N = 75$ for Temporal, its full edit split). Avg. is the unweighted mean of the five settings. Editing-only: data loading, evaluation, and the one-time preprocessing required by MEMIT and AlphaEdit (Table 7) are excluded. All cells are measured on a single NVIDIA H200 with batch size 1.

	ZsRE		**SelfCheckGPT**		**Temporal**	
Method	Llama	Mistral	Llama	Mistral	GPT-J	**Avg.**
FT	$0.23_{\pm 0.21}$	$0.46_{\pm 0.26}$	$0.59_{\pm 0.28}$	$1.10_{\pm 0.61}$	$0.84_{\pm 0.23}$	0.64
ROME	$1.51_{\pm 0.74}$	$7.75_{\pm 2.93}$	$4.14_{\pm 1.60}$	$41.76_{\pm 14.60}$	$31.27_{\pm 4.28}$	17.29
GRACE	$2.64_{\pm 0.07}$	$2.77_{\pm 0.07}$	$4.06_{\pm 0.85}$	$4.36_{\pm 0.81}$	$18.23_{\pm 3.11}$	6.41
DEFER	$6.18_{\pm 0.13}$	$6.36_{\pm 0.13}$	$8.49_{\pm 1.69}$	$9.63_{\pm 1.71}$	$9.66_{\pm 1.56}$	8.06
MEMIT	$5.68_{\pm 1.71}$	$6.39_{\pm 1.72}$	$11.16_{\pm 4.40}$	$15.83_{\pm 5.29}$	$17.90_{\pm 2.75}$	11.39
WiLKE	$27.01_{\pm 1.02}$	$25.76_{\pm 0.93}$	-	-	-	26.38
AlphaEdit	$6.93_{\pm 0.62}$	$9.67_{\pm 0.95}$	$20.74_{\pm 5.39}$	$21.53_{\pm 5.45}$	$21.51_{\pm 2.50}$	16.08
WISE	$6.37_{\pm 1.48}$	$5.29_{\pm 3.10}$	$19.90_{\pm 5.68}$	$37.75_{\pm 16.11}$	$48.42_{\pm 6.48}$	23.55
LoRA	$6.61_{\pm 0.40}$	$9.83_{\pm 0.32}$	$9.65_{\pm 1.61}$	$14.29_{\pm 2.24}$	$10.80_{\pm 1.30}$	10.24
AdaLoRA	$8.92_{\pm 0.26}$	$12.55_{\pm 0.19}$	$11.69_{\pm 1.62}$	$17.33_{\pm 2.28}$	$12.68_{\pm 1.30}$	12.63
SimIE	$10.31_{\pm 2.08}$	$11.47_{\pm 1.77}$	$39.28_{\pm 13.59}$	$44.24_{\pm 14.53}$	$32.68_{\pm 6.66}$	27.60
LOKI (Ours)	$10.82_{\pm 1.75}$	$8.15_{\pm 1.19}$	$17.64_{\pm 3.22}$	$13.63_{\pm 2.28}$	$24.56_{\pm 1.66}$	14.96

Table 7: Covariance matrix build times on H200 vs. A100 (Llama-3-8B-Instruct, Wikipedia, 100k samples). MEMIT and AlphaEdit both require these covariance statistics; AlphaEdit additionally builds a null-space projection matrix P per layer, adding $\sim$30–37 s/layer on H200 and $\sim$60–72 s/layer on A100 ($\sim$16–20 min and $\sim$34–38 min total for all 32 layers, respectively).

Layer	**Est. full (min) H200**	**Est. full (min) A100**
0	13.3	34.2
4	21.8	52.4
9	32.5	75.1
13	41.2	93.6
18	51.9	116.6
22	60.5	135.2
27	71.3	159.5
31	80.2	177.8
All 32	**$\sim$1,490 ($\sim$25 h)**	**$\sim$3,375 ($\sim$56 h)**

decomposition, ensuring new edits are projected onto subspaces orthogonal to the weight matrix structure. Weight updates are optimized via Adam with a combined loss function that includes: (1) cross-entropy loss on target tokens, (2) KL divergence regularization to preserve model behavior on unrelated inputs, and (3) HSIC-based bottleneck constraints to maintain information flow through intermediate layers.

Below are several implementation details regarding the different components of LOKI:

- As mentioned above, we use an additional KL divergence loss term similar to that of Meng et al. (2023) to preserve the KL divergence of model responses to the prompt "subject is a {insert subject}", in order to regularize the model further. The kl_factor hyperparameter is the coefficient of this loss term, added to the objective in Equation (8).
- Additionally, similar to Meng et al. (2022) once again, we prepend each input X_t with several (typically 5) prompts generated by the model from scratch, to improve generalization. The resulting inputs are then treated as a batch and used in the loss terms instead of X_t by itself. That is $\bar{X}_t = [[G_1, X_t], [G_2, X_t], ...]$, where G_is are generated by the LLM from scratch.

Table 8: LOKI hyperparameters for each benchmark and backbone reported in the main results. The ZsRE column is shared by Llama-3-8B-Instruct and Mistral-7B-v0.1; SelfCheckGPT lists separate settings for each model; Temporal lists GPT-J-6B.

	ZsRE	**SelfCheckGPT**		**Temporal**
Hyperparameter	Llama / Mistral	Llama-3-8B-Instruct	Mistral-7B	GPT-J-6B
Number of updated Layers	3	7	5	3
Optimization steps (num_steps)	25	25	25	30
Weight learning rate (w_lr)	1×10^{-4}	1.34×10^{-4}	2.58×10^{-4}	5.13×10^{-5}
Null-space dimension ($null_dim$)	1000	1500	3000	1000
HSIC coefficient λ_x	0.001	6.2×10^{-4}	4.46×10^{-4}	3.19×10^{-4}
HSIC coefficient λ_y	0.001	7.05×10^{-5}	1.78×10^{-4}	6.62×10^{-3}
HSIC bandwidth ($hsigma$)	1.0	2.28	2.66	10.86
KL regularization (kl_factor)	0.02	0.01	0.01	0.019
Norm constraint ($norm_constraint$)	0.05	1.62×10^{-1}	1.37×10^{-2}	1.47×10^{-3}

Table 9: Sensitivity of LOKI to the null-space dimension $null_dim$ (ZsRE, Llama-3-8B-Instruct, $\tau = 1{,}000$ sequential edits, all other hyperparameters fixed at their Table 8 ZsRE values). Avg. is the mean of Rel., Gen., and Loc.; "full" keeps the entire null space with no truncation. Performance is flat over $[500, 2000]$ and degrades at both extremes.

$null_dim$	100	250	500	1000	1500	2000	3000	5000	full
Avg. (↑)	0.820	0.874	0.887	0.888	**0.890**	0.886	0.880	0.869	0.846

- The above generalization also serves as samples for the HSIC criterion. As HSIC measures the independence between distributions, it requires multiple samples from each distribution. The embeddings of each input prepended by a prompt are averaged over the dimension of the tokens and each used as one sample for the HSIC criterion in Equation (6).
- We empirically find that the null-space matrix in Equation (7) tends to be more effective at reducing interference if it is low-rank. This is not surprising since it aligns with the fact that the desired subspace is a subset of the weight null-space, and that low-rank updates in general tend to interfere less with past knowledge, as is exploited in LoRA fine-tuning. Therefore, we retain the first $null_dim$ bases of the zero singular value bases of the $V^{(l)}$ matrix for calculating $P^{(l)}$, that is: $P^{(l)} = \bar{V}^{(l)}[:, : null_dim]\bar{V}^{(l)}[:, : null_dim]^\top$, where $null_dim$ is task-specific (Table 8). To assess sensitivity to this truncation, Table 9 sweeps $null_dim$ on ZsRE with Llama-3-8B-Instruct over $\tau = 1{,}000$ sequential edits, holding all other hyperparameters fixed. Performance is flat for $null_dim \in [500, 2000]$ (0.886–0.890) and degrades at both extremes: aggressive truncation discards too much of the null space (0.820 at $null_dim = 100$), while keeping the null space in full leaves the largest overlap with past edits (0.846). The task-specific values in Table 8 (1,000–3,000) all lie within 0.010 of the optimum, so the exact choice is not critical.
- The HSIC regularization term added to the objective in Equation (8) is slightly modified to use the output of a linear transform Z rather than previously $\hat{Z}$ so that both terms are differentiable w.r.t. to the modified weights. (Note that the layer selection term is exactly as described previously, and only the regularization term is changed slightly.)

We determine LOKI hyperparameters in three stages. For ZsRE (Tables 1 and **??**), we search over candidate settings on Llama-3-8B-Instruct and deploy the best-performing configuration for both Llama and Mistral without additional per-model tuning. For SelfCheckGPT (Table 2 and 13)) and Temporal (Table 3 and 14)), we use the standard training and evaluation splits. Specifically, we run a bounded search on the training split, select the configuration that optimizes the in-split objective for each backbone, and evaluate that single configuration on the held-out split used in the tables. Table 8 collects every numerical setting that enters the reported LOKI runs.

C.3 COMPETITORS HYPERPARAMETERS

FT. (Zhang et al., 2024) For Llama-3-8B-Instruct and Mistral-7B-v0.1, we edit layer 21 with 50 training steps using Adam optimizer with learning rate 5×10^{-4}. For GPT-J-6B, we use layer 21 with 25 steps and the same learning rate. All models use a maximum sequence length 40.

LoRA. (Hu et al., 2022) We fine-tune Low-Rank Adaptation (LoRA) modules on `q_proj` and `v_proj`, following the default method in EasyEdit (Wang et al., 2023). We use rank r=8, scaling α=32, dropout 0.1, Adam learning rate 1×10^{-4}, 50 optimizer steps per edit, and no weight decay.

AdaLoRA. (Zhang et al., 2023) We fine-tune Adaptive Low-Rank Adaptation (AdaLoRA) modules on `q_proj` and `v_proj`, following the default method in EasyEdit (Wang et al., 2023). Similar to LoRA, we use maximum rank r=8, scaling α=32, dropout 0.1, Adam learning rate 1×10^{-4}, 50 optimizer steps per edit, and no weight decay.

ROME. (Meng et al., 2022) For Llama-3-8B-Instruct and Mistral-7B-v0.1, we edit layer 5 following the default settings. For GPT-J-6B, we also edit layer 5 following their implementation. All models use 25 gradient steps (20 for GPT-J-6B) with learning rate 5×10^{-1} to compute hidden representations, and compute covariance statistics from 100,000 Wikipedia samples in float32 precision.

MEMIT. (Meng et al., 2023) For Llama-3-8B-Instruct and Mistral-7B-v0.1, we update layers [4, 5, 6, 7, 8] with covariance adjustment factor $\lambda = 15,000$. For GPT-J-6B, we target layers [3, 4, 5, 6, 7, 8] with the same λ value. All models use 25 gradient steps with learning rate 5×10^{-1}, and 100,000 Wikipedia samples for covariance computation.

WiLKE. (Hu et al., 2024) We could not find the original hyperparameters used by the authors other than the defaults in the published code. The computation cost of WiLKE increases linearly with number of available layers, so we set the available layers to [4,5,6,7,8] to match MEMIT. All other hyperparameters were kept as is from the code provided by the authors.

DEFER. (Hartvigsen et al., 2023) We follow the hyperparameter settings from Wang et al. (2024), using learning rate 7×10^{-5}, 100 training steps per edit, and 0.5 threshold. For Llama-3-8B-Instruct and Mistral-7B-v0.1, we edit layer 27; and for GPT-J-6B, layer 25, following the implementation in Wang et al. (2023).

GRACE. (Hartvigsen et al., 2023) Consistent with the original work, we set the learning rate to 1.0 and use the replace_last strategy, modifying only the final token's activations in autoregressive settings. The deferral radius ϵ is set to 1.0 for all models. We use 50 training iterations for Llama-3-8B-Instruct and Mistral-7B-v0.1, and 200 for GPT-J-6B. The edited layers are 29 for Llama-3-8B-Instruct, 29 for Mistral-7B-v0.1, and 25 for GPT-J-6B.

WISE. (Wang et al., 2024) We adopt the hyperparameter settings from Wang et al. (2024). For Llama-3-8B-Instruct, we use SGD optimizer with learning rate $\eta = 1.0$, mask ratio $\rho = 0.2$, activation margins $\alpha = 5.0$, $\beta = 20.0$, $\gamma = 10.0$, activation ratio 0.5, merge weight 0.5, and 30 training steps. For Mistral-7B-v0.1, we use the same configuration except 70 training steps, activation ratio 0.88, and merge weight 0.5. For GPT-J-6B, we use 70 steps, activation margins $\alpha = 15.0$, $\beta = 40.0$, $\gamma = 20.0$, activation ratio 0.88, and merge weight 1.0. All models use 2 knowledge shards and edit layer 29 (Llama-3), 27 (Mistral), or 21 (GPT-J-6B).

AlphaEdit. (Fang et al., 2025) For Llama-3-8B-Instruct and Mistral-7B-v0.1, we edit critical layers [4, 5, 6, 7, 8] following Meng et al. (2023). To compute hidden representations at each critical layer, we perform 25 optimization steps with learning rate 1×10^{-1}. For GPT-J-6B, we target layers [3, 4, 5, 6, 7, 8] with 25 optimization steps per layer and learning rate 5×10^{-1}. As in MEMIT, we use $\lambda = 15,000$ and 100,000 Wikipedia samples. The null-space threshold is 2×10^{-2} for all models, and L2 regularization is set to 1 for Llama-3 and Mistral, and 10 for GPT-J-6B.

SimIE. (Guo et al., 2025) For Llama-3-8B-Instruct and Mistral-7B-v0.1 we use ROME as the base method with $\lambda = 10$ for the best performance as detailed in the original paper. The hyperparameters of ROME are as before.

D EXPERIMENTAL DETAILS

D.1 DATASETS

ZsRE. Zero-shot Relation Extraction (ZsRE) (Levy et al., 2017) is a context-free question answering benchmark designed to evaluate factual knowledge storage and retrieval in language models. Each instance contains an edit prompt x_e (e.g., *"What was the date of Air France Flight 447?"*), a target answer y_e (e.g., *"1 June 2009"*), a semantically equivalent rephrase x'_e (e.g., *"What's the date of Air France Flight 447?"*) for generalization evaluation, and an unrelated locality prompt x_{loc} (e.g., *"where is the 7th game of the world series played"*) with its corresponding answer to assess whether edits affect irrelevant knowledge. We use the ZsRE splits provided by EasyEdit: 163,196 training samples, 1,037 edit samples, and 19,086 evaluation samples. For all ZsRE results reported in Table 1, we randomly sample 1,000 instances from the edit split.

SelfCheckGPT. We use the hallucination correction benchmark introduced by Hartvigsen et al. (2023), constructed from the SelfCheckGPT dataset (Manakul et al., 2023). The original SelfCheckGPT work prompted GPT-3 (Brown et al., 2020) to generate 238 Wikipedia-style biographies and annotated each sentence for factual accuracy. For model editing, hallucinated sentences are replaced with corresponding factually correct sentences from actual Wikipedia entries. Each instance contains a prompt x_e consisting of GPT-3-generated text up to the hallucination (e.g., *"This is a Wikipedia passage about ron meagher."*) and a target y_e containing the correct Wikipedia sentence (e.g., *"Ron Meagher (born October 2, 1941, Oakland, California, USA) is best known as the bassist of the American rock band The Beau Brummels."*). The dataset comprises 1,392 potential edits with substantially longer token sequences than standard QA benchmarks. Locality is evaluated using unrelated text continuations. Following Wang et al. (2024), we use 600 test samples after filtering excessively long sequences.

Temporal. The Temporal dataset (Hewitt et al., 2024) evaluates out-of-distribution (OOD) generalization for emerging entities. Each instance contains a prefix x_e extracted from an entity's Wikipedia page (e.g., *"Luka Dončić"*), a GPT-4 generated description y_e containing potentially noisy but informative content about the entity, and an actual Wikipedia suffix y_{ood} for OOD evaluation. The dataset tests whether edits performed on formulaic examples generalize to complex natural text, where the distributional shift involves linguistic complexity rather than domain change. Locality samples are drawn from the Pile (Gao et al., 2020), the pretraining corpus of GPT-J.

D.2 METRICS

We evaluate editing methods using three standard metrics (Yao et al., 2023; Zhang et al., 2024):

Reliability (Rel.). Measures whether the edited model correctly produces the target output for edit prompts. For QA tasks, this is computed as the accuracy on edit samples:

$$\text{Rel.} = \frac{1}{\tau}\sum_{t=0}^{\tau} \mathbb{1}\big(f_{\theta_t}(x_e^\top) = y_e^\top\big) \ . \tag{14}$$

Higher accuracy is better (↑). For hallucination correction (SelfCheckGPT), we report perplexity (PPL) on the target text following Wang et al. (2024); lower PPL is better (↓).

Generalization (Gen.). Assesses whether edits transfer to semantically equivalent rephrasings of the edit prompt. When paraphrases x'_e are available,

$$\text{Gen.} = \frac{1}{\tau}\sum_{t=0}^{\tau} \mathbb{1}\big(f_{\theta_t}(x_{e'}^\top) = y_e^\top\big) \ . \tag{15}$$

Higher accuracy is better (↑). This formulation applies where paraphrase prompts are supplied (ZsRE). For OOD evaluation on Temporal, we use a probability threshold where success is defined as $\mathcal{L}_{\theta_t}(x_e, y_{\text{ood}}) < -\log(0.8)$; higher success rate is better (↑). For SelfCheckGPT we do not report Gen. because the dataset does not include paraphrase examples.

Locality (Loc.). Evaluates whether edits preserve the model's behavior on unrelated inputs:

$$\text{Loc.} = \frac{1}{\tau}\sum_{t=0}^{\tau} \mathbb{1}\big(f_{\theta_t}(x_{\text{loc}}^\top) = f_{\theta_0}(x_{\text{loc}}^\top)\big) \ . \tag{16}$$

Higher agreement with the pretrained model's predictions on locality prompts is better (↑). This metric is shared across all tasks.

D.3 COMPETITORS

FT. (Zhang et al., 2024) Fine-tunes a single MLP layer using the autoregressive loss while freezing all other parameters. It utilizes a cross-entropy loss on the target answer while masking the original text.

LoRA. (Hu et al., 2022) Freezes the base model and injects trainable low-rank update matrices into the target projection modules. Training minimizes next-token cross-entropy on the new answer while masking the prompt region.

AdaLoRA. (Zhang et al., 2023) Freezes the base model and fits low-rank adapters with an adaptive rank budget across target modules. Training minimizes next-token cross-entropy on the new answer while masking the prompt region.

ROME. (Meng et al., 2022) Rank-One Model Editing locates factual associations in specific MLP layers via causal tracing, then modifies parameters through least-squares approximation to inject new knowledge.

MEMIT. (Meng et al., 2023) Mass-Editing Memory in a Transformer extends ROME to multiple layers simultaneously, enabling batch editing of thousands of facts in a single operation.

WiLKE. (Hu et al., 2024) An extension of ROME/MEMIT, which proposes to choose dynamically which layer to edit based on the norm of the optimized intervention vector used in ROME. This is the only other method we are aware of that uses dynamic layer selection for knowledge editing. Furthermore, the selection process is incredibly costly, as it requires a full optimization of the intervention vector for each layer that can be chosen. This means that effectively, if WiLKE is able to choose from even a small number of layers, for example 5, it would require roughly 5 times the computation time compared to MEMIT/ROME.

DEFER. (Hartvigsen et al., 2023) Re-implementation of SERAC (Mitchell et al., 2022) for lifelong editing. Introduces a scope classifier g that routes inputs to either the original model or an auxiliary one-layer counterfactual network o. Both networks are jointly fine-tuned during editing.

GRACE. (Hartvigsen et al., 2023) General Retrieval Adaptors for Continual Editing maintains a discrete key-value codebook without modifying model weights. During inference, input activations are matched against cached keys; if within a learned deferral radius ϵ, the corresponding value replaces the layer output.

WISE. (Wang et al., 2024) Introduces a dual-memory architecture with a frozen main memory (original weights) and a trainable side memory for edits. A routing mechanism directs queries to the appropriate memory based on activation patterns. Knowledge sharding distributes edits across orthogonal parameter subspaces, later merged via TIES-Merge to prevent conflicts.

AlphaEdit. (Fang et al., 2025) Extends locate-then-edit methods by projecting weight perturbations onto the null space of previously edited knowledge, ensuring new edits minimally interfere with prior updates.

SimIE. (Guo et al., 2025) Formulates the ideal parameter shift as the minimum-norm solution to a linear system, constructed using the Moore-Penrose inverse, and subsequently enables recursive updates by truncating the limiting expression of the Moore-Penrose inverse under two mild assumptions.

E EXTENDED RESULTS

As discussed in App. C, results for smaller sequential edit scales ($\tau \leq 100$ for ZsRE, SelfCheckGPT and $\tau = 10$ for Temporal) are averaged over several runs with distinct edit samples to mitigate variance. In this section, we extend the results to include the standard deviation across multiple independent runs to provide a comprehensive view of the stability and reliability of each editing method. The following tables present these detailed metrics all three datasets ZsRE in **??**, SelfCheckGPT in Table 13, and Temporal in Table 14.

Table 10: Performance comparison on the ZsRE dataset on Llama3-8B batch editing with a total of $\tau = 2000$ samples across sequential batches of 100.

Method	Rel	Gen	Loc	Avg
FT	0.90	0.79	0.39	0.70
AlphaEdit	0.90	0.69	0.68	0.76
LOKI	**0.92**	**0.78**	**0.83**	**0.84**

Table 11: Performance comparison on the ZsRE dataset on Mistral-7B batch editing with a total of $\tau = 2000$ samples across sequential batches of 100.

Method	Rel	Gen	Loc	Avg
FT	0.80	0.71	0.16	0.56
AlphaEdit	0.82	0.64	**0.72**	0.73
LOKI	**0.92**	**0.90**	**0.72**	**0.85**

Table 12: Sequential model editing results on the ZsRE dataset (Llama-3-8B-Instruct and Mistral-7B). Metrics include Reliability (Rel. ↑), Generalization (Gen. ↑), and Locality (Loc. ↑) across τ sequential edits, with Avg. representing the mean of the three scores. **Bold** indicates the best result and underlining indicates the second-best result for each metric and number of edits (τ). LOKI consistently achieves the highest average performance across all edit scales, demonstrating superior scalability and stability compared to existing methods as τ increases up to 1,000.

τ	Metric	FT	LoRA	AdaLoRA	ROME	MEMIT	DEFER	GRACE	SimIE	WISE	AlphaEdit	**LOKI** (Ours)
Llama-3-8B-Instruct												
1	Rel. (↑)	**1**$_{\pm 0.00}$	**1**$_{\pm 0.00}$	**1**$_{\pm 0.00}$	0.99$_{\pm 0.03}$	0.99$_{\pm 0.02}$	0.82$_{\pm 0.24}$	**1**$_{\pm 0.00}$	0.75$_{\pm 0.28}$	0.99$_{\pm 0.08}$	0.99$_{\pm 0.05}$	**1**$_{\pm 0.00}$
	Gen. (↑)	**0.99**$_{\pm 0.05}$	**0.99**$_{\pm 0.07}$	0.73$_{\pm 0.30}$	0.96$_{\pm 0.02}$	0.97$_{\pm 0.02}$	0.86$_{\pm 0.22}$	0.46$_{\pm 0.02}$	0.71$_{\pm 0.29}$	**0.99**$_{\pm 0.08}$	0.88$_{\pm 0.25}$	0.98$_{\pm 0.08}$
	Loc. (↑)	0.95$_{\pm 0.14}$	0.49$_{\pm 0.33}$	0.82$_{\pm 0.24}$	0.96$_{\pm 0.01}$	0.98$_{\pm 0.05}$	0.68$_{\pm 0.31}$	**1**$_{\pm 0.00}$	0.79$_{\pm 0.26}$	0.92$_{\pm 0.24}$	0.99$_{\pm 0.04}$	0.99$_{\pm 0.03}$
	Avg. (↑)	0.98$_{\pm 0.09}$	0.83$_{\pm 0.11}$	0.85$_{\pm 0.13}$	0.97$_{\pm 0.02}$	0.98$_{\pm 0.04}$	0.79$_{\pm 0.15}$	0.82$_{\pm 0.00}$	0.75$_{\pm 0.16}$	0.97$_{\pm 0.09}$	0.95$_{\pm 0.09}$	**0.99**$_{\pm 0.06}$
10	Rel. (↑)	0.87$_{\pm 0.08}$	0.76$_{\pm 0.15}$	0.83$_{\pm 0.08}$	0.61$_{\pm 0.11}$	0.68$_{\pm 0.11}$	0.66$_{\pm 0.13}$	**1**$_{\pm 0.00}$	0.73$_{\pm 0.09}$	0.77$_{\pm 0.11}$	0.92$_{\pm 0.07}$	0.94$_{\pm 0.04}$
	Gen. (↑)	0.83$_{\pm 0.09}$	0.72$_{\pm 0.14}$	0.60$_{\pm 0.11}$	0.59$_{\pm 0.11}$	0.67$_{\pm 0.11}$	0.67$_{\pm 0.13}$	0.42$_{\pm 0.15}$	0.69$_{\pm 0.10}$	0.74$_{\pm 0.11}$	0.84$_{\pm 0.10}$	**0.87**$_{\pm 0.08}$
	Loc. (↑)	0.80$_{\pm 0.10}$	0.32$_{\pm 0.15}$	0.57$_{\pm 0.14}$	0.41$_{\pm 0.20}$	0.73$_{\pm 0.09}$	0.45$_{\pm 0.17}$	**1**$_{\pm 0.00}$	0.79$_{\pm 0.09}$	**1**$_{\pm 0.00}$	0.98$_{\pm 0.04}$	0.98$_{\pm 0.02}$
	Avg. (↑)	0.83$_{\pm 0.05}$	0.60$_{\pm 0.12}$	0.67$_{\pm 0.07}$	0.54$_{\pm 0.08}$	0.69$_{\pm 0.06}$	0.59$_{\pm 0.08}$	0.81$_{\pm 0.03}$	0.73$_{\pm 0.05}$	0.84$_{\pm 0.05}$	0.92$_{\pm 0.09}$	**0.93**$_{\pm 0.03}$
100	Rel. (↑)	0.84$_{\pm 0.04}$	0.47$_{\pm 0.07}$	0.54$_{\pm 0.04}$	0.08$_{\pm 0.07}$	0.03$_{\pm 0.02}$	0.33$_{\pm 0.02}$	**1**$_{\pm 0.00}$	0.72$_{\pm 0.02}$	0.61$_{\pm 0.05}$	0.90$_{\pm 0.02}$	**0.93**$_{\pm 0.02}$
	Gen. (↑)	0.81$_{\pm 0.05}$	0.45$_{\pm 0.07}$	0.49$_{\pm 0.04}$	0.08$_{\pm 0.07}$	0.03$_{\pm 0.01}$	0.32$_{\pm 0.02}$	0.39$_{\pm 0.07}$	0.67$_{\pm 0.02}$	0.6$_{\pm 0.05}$	0.84$_{\pm 0.03}$	**0.86**$_{\pm 0.03}$
	Loc. (↑)	0.42$_{\pm 0.06}$	0.30$_{\pm 0.08}$	0.44$_{\pm 0.07}$	0.02$_{\pm 0.01}$	0.01$_{\pm 0.01}$	**1**$_{\pm 0.00}$	**1**$_{\pm 0.00}$	0.78$_{\pm 0.04}$	**1**$_{\pm 0.00}$	0.94$_{\pm 0.04}$	0.96$_{\pm 0.02}$
	Avg. (↑)	0.69$_{\pm 0.03}$	0.41$_{\pm 0.06}$	0.49$_{\pm 0.04}$	0.06$_{\pm 0.03}$	0.02$_{\pm 0.03}$	0.55$_{\pm 0.01}$	0.8$_{\pm 0.02}$	0.72$_{\pm 0.02}$	0.73$_{\pm 0.02}$	0.90$_{\pm 0.02}$	**0.92**$_{\pm 0.01}$
1000	Rel. (↑)	0.83	0.36	0.36	0.03	0.00	0.31	**1**	0.69	0.54	0.76	0.91
	Gen. (↑)	0.80	0.35	0.35	0.03	0.00	0.31	0.37	0.65	0.53	**0.88**	0.87
	Loc. (↑)	0.11	0.20	0.26	0.02	0.00	**1**	**1**	0.75	0.99	0.65	0.88
	Avg. (↑)	0.58	0.31	0.32	0.03	0.00	0.54	0.79	0.69	0.69	0.76	**0.89**
Mistral-7B												
1	Rel. (↑)	0.99$_{\pm 0.03}$	**1**$_{\pm 0.00}$	**1**$_{\pm 0.00}$	0.79$_{\pm 0.23}$	0.81$_{\pm 0.23}$	0.59$_{\pm 0.27}$	**1**$_{\pm 0.00}$	0.62$_{\pm 0.29}$	0.96$_{\pm 0.16}$	0.86$_{\pm 0.17}$	**1**$_{\pm 0.00}$
	Gen. (↑)	0.98$_{\pm 0.06}$	**1**$_{\pm 0.00}$	0.91$_{\pm 0.18}$	0.77$_{\pm 0.23}$	0.79$_{\pm 0.23}$	0.68$_{\pm 0.29}$	0.36$_{\pm 0.02}$	0.59$_{\pm 0.29}$	0.93$_{\pm 0.21}$	0.78$_{\pm 0.22}$	0.96$_{\pm 0.11}$
	Loc. (↑)	0.75$_{\pm 0.29}$	0.31$_{\pm 0.30}$	0.84$_{\pm 0.22}$	0.98$_{\pm 0.05}$	0.99$_{\pm 0.04}$	0.79$_{\pm 0.34}$	**1**$_{\pm 0.00}$	0.94$_{\pm 0.13}$	**1**$_{\pm 0.00}$	0.99$_{\pm 0.01}$	0.98$_{\pm 0.05}$
	Avg. (↑)	0.91$_{\pm 0.10}$	0.77$_{\pm 0.10}$	0.92$_{\pm 0.09}$	0.85$_{\pm 0.11}$	0.86$_{\pm 0.11}$	0.69$_{\pm 0.17}$	0.79$_{\pm 0.01}$	0.72$_{\pm 0.14}$	0.96$_{\pm 0.09}$	0.88$_{\pm 0.09}$	**0.98**$_{\pm 0.06}$
10	Rel. (↑)	0.82$_{\pm 0.12}$	0.63$_{\pm 0.18}$	0.91$_{\pm 0.10}$	0.58$_{\pm 0.12}$	0.46$_{\pm 0.13}$	0.44$_{\pm 0.08}$	**1**$_{\pm 0.00}$	0.61$_{\pm 0.09}$	0.88$_{\pm 0.07}$	0.82$_{\pm 0.07}$	0.96$_{\pm 0.04}$
	Gen. (↑)	0.77$_{\pm 0.12}$	0.60$_{\pm 0.17}$	0.78$_{\pm 0.11}$	0.57$_{\pm 0.12}$	0.45$_{\pm 0.13}$	0.43$_{\pm 0.07}$	0.15$_{\pm 0.04}$	0.58$_{\pm 0.09}$	0.84$_{\pm 0.09}$	0.76$_{\pm 0.08}$	**0.91**$_{\pm 0.08}$
	Loc. (↑)	0.22$_{\pm 0.18}$	0.21$_{\pm 0.16}$	0.62$_{\pm 0.09}$	0.75$_{\pm 0.19}$	0.61$_{\pm 0.26}$	**1**$_{\pm 0.00}$	**1**$_{\pm 0.00}$	0.94$_{\pm 0.04}$	0.99$_{\pm 0.00}$	0.99$_{\pm 0.01}$	0.97$_{\pm 0.02}$
	Avg. (↑)	0.60$_{\pm 0.08}$	0.48$_{\pm 0.15}$	0.77$_{\pm 0.08}$	0.63$_{\pm 0.08}$	0.51$_{\pm 0.10}$	0.62$_{\pm 0.04}$	0.72$_{\pm 0.02}$	0.71$_{\pm 0.05}$	0.91$_{\pm 0.04}$	0.86$_{\pm 0.04}$	**0.95**$_{\pm 0.03}$
100	Rel. (↑)	0.83$_{\pm 0.03}$	0.33$_{\pm 0.09}$	0.56$_{\pm 0.09}$	0.05$_{\pm 0.03}$	0.00$_{\pm 0.00}$	0.4$_{\pm 0.02}$	**1**$_{\pm 0.00}$	0.60$_{\pm 0.02}$	0.82$_{\pm 0.06}$	0.80$_{\pm 0.03}$	0.95$_{\pm 0.00}$
	Gen. (↑)	0.77$_{\pm 0.04}$	0.32$_{\pm 0.07}$	0.51$_{\pm 0.07}$	0.05$_{\pm 0.03}$	0$_{\pm 0.00}$	0.4$_{\pm 0.02}$	0.15$_{\pm 0.01}$	0.58$_{\pm 0.02}$	0.76$_{\pm 0.05}$	0.75$_{\pm 0.04}$	**0.88**$_{\pm 0.03}$
	Loc. (↑)	0.07$_{\pm 0.02}$	0.17$_{\pm 0.11}$	0.43$_{\pm 0.06}$	0.02$_{\pm 0.02}$	0.01$_{\pm 0.00}$	**1**$_{\pm 0.00}$	**1**$_{\pm 0.00}$	0.93$_{\pm 0.02}$	0.99$_{\pm 0.00}$	0.96$_{\pm 0.02}$	0.93$_{\pm 0.02}$
	Avg. (↑)	0.56$_{\pm 0.02}$	0.28$_{\pm 0.08}$	0.50$_{\pm 0.06}$	0.04$_{\pm 0.01}$	0$_{\pm 0.00}$	0.6$_{\pm 0.01}$	0.72$_{\pm 0.00}$	0.70$_{\pm 0.01}$	0.86$_{\pm 0.03}$	0.84$_{\pm 0.02}$	**0.92**$_{\pm 0.01}$
1000	Rel. (↑)	0.69	0.15	0.50	0.04	0.04	0.4	**1**	0.59	0.65	0.80	0.92
	Gen. (↑)	0.65	0.15	0.48	0.04	0.04	0.39	0.02	0.56	0.63	0.74	**0.86**
	Loc. (↑)	0.07	0.07	0.40	0.02	0.02	**1**	**1**	0.92	0.99	0.75	0.82
	Avg. (↑)	0.47	0.12	0.46	0.03	0.03	0.6	0.67	0.69	0.76	0.76	**0.87**

Batch Editing Results. Following Fang et al. (2025), we perform a sequential batch editing experiment where a total of $\tau = 2000$ samples are edited sequentially in batches of 100. We present the results in Table 10 for Llama-3-8B and Table 11 for Mistral-7B on the ZsRE dataset, showing that LOKI outperforms AlphaEdit as well the fine-tuning (FT) baseline in the batch sequential editing.

F Combining AlphaEdit with LOKI layer selection

To further isolate the effect of our layer selection algorithm, we perform an experiment combining the LOKI dynamic layer selection algorithm, with the knowledge insertion algorithm of AlphaEdit. Meaning, for each sample, first the editing layers are selected using our HSIC-based dynamic layer

Table 13: Performance comparison on the SelfCheckGPT hallucination correction task with standard deviations. Metrics include Reliability (Rel. ↓) and Locality (Loc. ↑) across τ sequential edits. For this task, Rel. is measured as perplexity (PPL) of the target text. Generalization is not available in this dataset since no rephrasing examples are provided. **Bold** indicates the best result and underlining indicates the second-best result for each metric. LOKI demonstrates superior Reliability throughout the editing process across both Llama-3-8B and Mistral-7B. Overall, it maintains competitive performance and avoids the reliability collapse observed in methods at higher edit scales.

Method	$\tau = 1$		$\tau = 10$		$\tau = 100$		$\tau = 600$	
	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)
Llama-3-8B-Instruct								
FT	1.01$_{\pm 0.00}$	0.95$_{\pm 0.07}$	1.12$_{\pm 0.10}$	0.84$_{\pm 0.04}$	1.44$_{\pm 0.12}$	0.65$_{\pm 0.02}$	2.02	0.45
LoRA	**1.00**$_{\pm 0.00}$	0.82$_{\pm 0.13}$	3.52$_{\pm 2.03}$	0.69$_{\pm 0.09}$	6.55e1$_{\pm 66.4}$	0.55$_{\pm 0.10}$	2.48e2	0.49
AdaLoRA	1.05$_{\pm 0.24}$	0.98$_{\pm 0.03}$	5.03$_{\pm 2.40}$	0.93$_{\pm 0.02}$	3.41e2$_{\pm 6.41e2}$	0.85$_{\pm 0.02}$	1.27e2	0.76
ROME	1.39$_{\pm 1.05}$	0.97$_{\pm 0.04}$	4.74e1$_{\pm 175}$	0.61$_{\pm 0.09}$	5.00e4$_{\pm 7.07e4}$	0.02$_{\pm 0.01}$	2.89e4	0.01
MEMIT	1.32$_{\pm 0.52}$	0.99$_{\pm 0.02}$	1.84e3$_{\pm 1.36e4}$	0.93$_{\pm 0.11}$	6.25e10$_{\pm 1.40e11}$	0.01$_{\pm 0.01}$	3.75e4	0.05
DEFER	2.34e1$_{\pm 1.25e2}$	0.80$_{\pm 0.15}$	6.86e1$_{\pm 335}$	**1.00**$_{\pm 0.01}$	7.12e1$_{\pm 95.9}$	**1.00**$_{\pm 0.00}$	7.12e1	**1.00**
GRACE	1.40$_{\pm 1.98}$	**1.00**$_{\pm 0.00}$	7.10e1$_{\pm 335}$	**1.00**$_{\pm 0.00}$	7.17e1$_{\pm 95.4}$	**1.00**$_{\pm 0.00}$	7.54e1	**1.00**
WISE	1.44$_{\pm 3.18}$	0.97$_{\pm 0.10}$	3.47$_{\pm 12.2}$	0.97$_{\pm 0.04}$	3.54$_{\pm 0.21}$	0.99$_{\pm 0.00}$	2.63e2	0.99
AlphaEdit	1.29$_{\pm 1.04}$	**1.00**$_{\pm 0.01}$	1.49$_{\pm 0.66}$	0.99$_{\pm 0.01}$	5.04$_{\pm 0.97}$	0.96$_{\pm 0.01}$	9.04e3	0.05
LOKI (Ours)	1.01$_{\pm 0.00}$	0.98$_{\pm 0.02}$	**1.01**$_{\pm 0.01}$	0.98$_{\pm 0.01}$	**1.03**$_{\pm 0.02}$	0.94$_{\pm 0.02}$	**1.14**	0.85
Mistral-7B								
FT	1.02$_{\pm 0.07}$	0.77$_{\pm 0.12}$	1.41$_{\pm 0.36}$	0.44$_{\pm 0.06}$	1.94$_{\pm 0.34}$	0.22$_{\pm 0.01}$	7.26	0.20
LoRA	**1.00**$_{\pm 0.00}$	0.70$_{\pm 0.13}$	2.61$_{\pm 1.18}$	0.61$_{\pm 0.10}$	1.58e1$_{\pm 3.54}$	0.54$_{\pm 0.10}$	4.65e1	0.41
AdaLoRA	1.01$_{\pm 0.04}$	0.97$_{\pm 0.03}$	2.36$_{\pm 1.01}$	0.92$_{\pm 0.03}$	3.19e1$_{\pm 30.4}$	0.84$_{\pm 0.01}$	1.53e2	0.75
ROME	1.98$_{\pm 1.20}$	0.99$_{\pm 0.02}$	2.58$_{\pm 1.34}$	0.91$_{\pm 0.02}$	6.77e3$_{\pm 3.80e3}$	0.03$_{\pm 0.00}$	7.87e3	0.01
MEMIT	1.59$_{\pm 1.15}$	**1.00**$_{\pm 0.01}$	3.13e2$_{\pm 1.49e3}$	0.93$_{\pm 0.20}$	9.03e4$_{\pm 1.31e5}$	0$_{\pm 0.01}$	3.33e6	0.00
DEFER	1.76e1$_{\pm 116}$	0.92$_{\pm 0.12}$	3.22e1$_{\pm 162}$	0.99$_{\pm 0.02}$	3.22e1$_{\pm 47.1}$	**1.00**$_{\pm 0.00}$	3.22e1	**1.00**
GRACE	1.92$_{\pm 2.04}$	**1.00**$_{\pm 0.00}$	3.21e1$_{\pm 162}$	**1.00**$_{\pm 0.00}$	3.22e1$_{\pm 47.2}$	**1.00**$_{\pm 0.00}$	3.25e1	**1.00**
WISE	1.08$_{\pm 0.66}$	**1.00**$_{\pm 0.02}$	1.43e1$_{\pm 91.5}$	0.98$_{\pm 0.05}$	1.74$_{\pm 0.29}$	0.96$_{\pm 0.01}$	10.0	0.92
AlphaEdit	1.71$_{\pm 1.50}$	**1.00**$_{\pm 0.01}$	1.88$_{\pm 0.77}$	0.99$_{\pm 0.00}$	2.96$_{\pm 0.58}$	0.98$_{\pm 0.00}$	6.43e1	0.87
LOKI (Ours)	1.01$_{\pm 0.00}$	0.99$_{\pm 0.02}$	**1.01**$_{\pm 0.01}$	0.99$_{\pm 0.01}$	**1.04**$_{\pm 0.01}$	0.94$_{\pm 0.01}$	**1.13**	0.87

Table 14: OOD generalization performance on the Temporal dataset (GPT-J-6B) with standard deviations. Metrics include Reliability (Rel. ↑), OOD Generalization (OOD Gen. ↑), and Locality (Loc. ↑) across τ sequential edits, with Avg. denoting their mean. **Bold** indicates the best result and underlining indicates the second-best result for each metric. LOKI maintains high Reliability and OOD Generalization across edit scales. While exhibiting lower Locality compared to specific memory-based methods, it prevents the significant reliability collapse observed in methods like ROME and MEMIT as τ increases.

Method	$\tau = 10$				$\tau = 75$			
	Rel. (↑)	OOD Gen. (↑)	Loc. (↑)	Avg. (↑)	Rel. (↑)	OOD Gen. (↑)	Loc. (↑)	Avg. (↑)
w/o Editing	0.56	0.21	—	—	0.56	0.21	—	—
FT	0.96$_{\pm 0.02}$	0.35$_{\pm 0.03}$	0.69$_{\pm 0.02}$	0.67$_{\pm 0.01}$	0.93	0.32	0.62	0.62
LoRA	0.13$_{\pm 0.03}$	0.00$_{\pm 0.00}$	0.07$_{\pm 0.03}$	0.07$_{\pm 0.02}$	0.10	0.00	0.04	0.05
AdaLoRA	0.38$_{\pm 0.04}$	0.17$_{\pm 0.03}$	0.24$_{\pm 0.04}$	0.26$_{\pm 0.03}$	0.34	0.20	0.31	0.28
ROME	0.10$_{\pm 0.04}$	0$_{\pm 0.00}$	0.06$_{\pm 0.02}$	0.05$_{\pm 0.01}$	0.02	0.00	0.02	0.01
MEMIT	0.82$_{\pm 0.05}$	0.28$_{\pm 0.02}$	0.99$_{\pm 0.01}$	0.70$_{\pm 0.02}$	0.43	0.14	0.49	0.35
DEFER	0.66$_{\pm 0.02}$	0.29$_{\pm 0.02}$	0.81$_{\pm 0.05}$	0.59$_{\pm 0.02}$	0.55	0.21	0.95	0.57
GRACE	0.59$_{\pm 0.02}$	0.22$_{\pm 0.02}$	**1**$_{\pm 0.00}$	0.60$_{\pm 0.01}$	0.56	0.22	**1**	0.59
WISE	0.99$_{\pm 0.01}$	**0.36**$_{\pm 0.02}$	0.93$_{\pm 0.06}$	**0.76**$_{\pm 0.02}$	**0.98**	**0.38**	0.99	**0.78**
AlphaEdit	0.90$_{\pm 0.03}$	0.27$_{\pm 0.02}$	0.99$_{\pm 0.01}$	0.72$_{\pm 0.01}$	0.87	0.28	0.97	0.71
LOKI (Ours)	**1**$_{\pm 0.00}$	**0.36**$_{\pm 0.02}$	0.82$_{\pm 0.06}$	0.73$_{\pm 0.02}$	**0.98**	0.36	0.81	0.72

selection as described in Section 4, and then knowledge insertion is performed using the same method as used in AlphaEdit. The results are presented in Table 15, and show that while across smaller τ the results are not much different, substantial improvements are observed for $\tau = 1000$, the most challenging scenario. This is while both settings still underperform LOKI in most metrics. We hypothesize that while the layer selection does improve AlphaEdit, its gains are limited by AlphaEdits two step optimization/update scheme, as well as it's requirement to store features of previous edits for each layer in order to avoid catastrophic forgetting. LOKI does not suffer from these limitations.

Table 15: Results on the ZsRE dataset (Llama-3-8B) for combining AlphaEdit with the layer selection algorithm of LOKI. Metrics include Reliability (Rel. ↑), Generalization (Gen. ↑), and Locality (Loc. ↑) across τ sequential edits, with Avg. representing the mean of the three scores. **Bold** is the best result. Using the LOKI layer selection improves AlphaEdit performance substantially for $\tau = 1000$.

Method	$\tau = 1$				$\tau = 10$				$\tau = 100$				$\tau = 1000$			
	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
AlphaEdit	**0.99**	**0.88**	**0.99**	**0.95**	0.92	0.84	**0.98**	0.92	0.90	**0.84**	**0.94**	**0.90**	0.76	**0.88**	0.65	0.76
AlphaEdit + LOKI layer selection	**0.99**	0.86	**0.99**	**0.95**	**0.94**	**0.85**	**0.98**	**0.93**	**0.91**	0.82	0.93	0.89	**0.94**	0.80	**0.78**	**0.84**

G Limitations, Future Works and Broader Impacts

Limitations LOKI attempts to solve the shortcomings of previous knowledge editing methods, such as requiring historical data or inadequate selection of editing parameters. While we achieve competitive performance without historical data and by performing dynamic per-sample layer selection, LOKI has its own limitations. Firstly, computing the SVD of large weight matrices is time-consuming. While LOKI is still faster than WISE and AlphaEdit, it still suffers from some computational overhead. Second, while LOKI is the first to perform dynamic layer selection for knowledge editing, it is constrained to choosing entire layers for editing rather than individual parameters or neurons. Overall we believe the benefits of LOKI, especially bypassing the requirement of historical data and time-consuming pre-processing or additional parameters, outweigh its limitations. We believe LOKI is a first step towards elevating lifelong knowledge editing to new heights.

Future Works LOKI takes an important step towards efficient and sustainable memory-free lifelong knowledge editing. However, there is still room for improvement. Firstly, as mentioned in the limitations section, one can devise an algorithm to dynamically select individual parameters or neurons rather than entire layers, giving more flexibility to the algorithm for more precise knowledge insertion. Second, the weight null-space can be further constrained using heuristics or additional theorems to provide further decoupling from past knowledge, improving locality.

Broader Impacts In this work, we present LOKI, a novel knowledge editing method that aims to facilitate the incorporation of new knowledge into LLMs. As LLMs see widespread use in society, knowledge editing in LLMs can be beneficial in a variety of aspects: fixing model mistakes (Balachandran et al., 2022), correcting hallucinations (Ji et al., 2023), or removing bias (Ferrara, 2023). However, as with any algorithm capable of modifying LLM behaviors, knowledge editing methods can be misused. For example, models can be edited to encourage toxic behavior, generate misinformation, or use inappropriate language. These risks are not exclusive to LOKI, and we believe the behavior and usage of LLMs should be sufficiently regulated to prevent misuse, including with knowledge editing. Overall, we believe that the benefits of LOKI are substantial enough to outweigh potential risks and that LOKI does not raise any new or exclusive ethical concerns.