Effective Skill Unlearning through Intervention and Abstention

Yongce Li
UCSD HDSI
yol013@ucsd.edu

Chung-En Sun
UCSD CSE
cesun@ucsd.edu

Tsui-Wei Weng
UCSD HDSI
lweng@ucsd.edu

Abstract

Large language Models (LLMs) have demonstrated remarkable skills across various domains. Understanding the mechanisms behind their abilities and implementing controls over them is becoming increasingly important for developing better models. In this paper, we focus on skill unlearning in LLMs, specifically unlearning a particular skill while retaining their overall capabilities. We introduce two lightweight, training-free machine skill unlearning techniques for LLMs. First, we observe that the pre-activation distribution of neurons in each Feed-Forward Layer (FFL) differs when the model demonstrates different skills. Additionally, we find that queries triggering the same skill cluster within the FFL key space and can be separated from other queries using a hypercube. Based on these observations, we propose two lightweight, training-free skill unlearning methods via *intervention* and *abstention* respectively: `Neuron Adjust` and `Key Space Detection`. We evaluate our methods on unlearning math-solving, Python-coding, and comprehension skills across seven different languages. The results demonstrate their strong unlearning capabilities for the designated skills. Specifically, `Key Space Detection` achieves over 80% relative performance drop on the forgetting skill and less than 10% relative performance drop on other skills and the model's general knowledge (MMLU) for most unlearning tasks. [1]

1 Introduction

In recent years, the superior capabilities demonstrated by Large Language Models (LLMs) have attracted significant research interest. Without training on task-specific datasets, LLMs exhibit strong skills in various domains such as math (Wei et al., 2022; Imani et al., 2023; Cobbe et al., 2021), coding (Austin et al., 2021; Li et al., 2022), and language comprehension (Shi et al., 2023). Understanding the mechanisms behind these abilities and implementing controls over them are becoming increasingly important for developing stronger, safer, and more interpretable models.

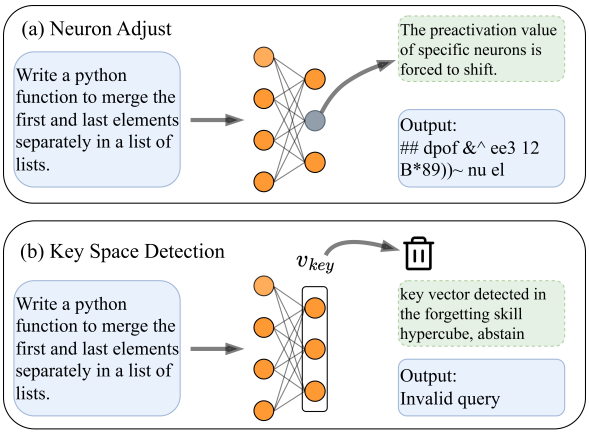

Figure 1: An overview of the proposed skill unlearning methods: `Neuron Adjust` (through *intervention*) and `Key Space Detection` (through *abstention*). This example illustrates forgetting coding skill.

A recent line of research focuses on machine unlearning (Yao et al., 2023; Liu et al., 2024), which aims to remove the knowledge LLMs have acquired from specific datasets while maintaining their causally unrelated knowledge. In this work, we focus on a variant of machine unlearning called skill unlearning, which aims to remove a specific skill (e.g., coding skill, elementary math-solving skill) from the LLM while retaining its other skills. Skill unlearning helps researchers control certain behaviors of LLMs, providing insights into when and how a model demonstrates a particular skill.

Currently, most unlearning methods (Lu et al.,

[1] Our code is available at https://github.com/Trustworthy-ML-Lab/effective_skill_unlearning

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 6358–6371
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

	(I) Efficiency		(II) Performance		(III) Scalability
Method:	Does not require training	No Inference time cost	High quality unlearning	Maintain model overall capability	Applicable to large models
Retrain from scratch	No	**Yes**	**Yes**	**Yes**	No
Fine-tuning based unlearning	No	**Yes**	**Yes**	No	**Yes**
In-context unlearning	**Yes**	**Yes**	No	**Yes**	**Yes**
`Selective Pruning`	**Yes**	**Yes**	**Yes**	No	**Yes**
`Neuron Adjust` (**Ours**)	**Yes**	$O(1)$	**Yes**	Partial	**Yes**
`Key Space Detection` (**Ours**)	**Yes**	$O(1)$	**Yes**	**Yes**	**Yes**

Table 1: Comparison of our method against existing machine unlearning methods, including retraining, fine-tuning based methods, in-context unlearning, and a prune-based unlearning method `Selective Pruning`.

2022; Jang et al., 2023; Wang et al., 2023; Yu et al., 2023; Eldan and Russinovich, 2023; Chen and Yang, 2023; Yao et al., 2023) rely on fine-tuning, which becomes increasingly costly as LLMs grow larger. Other unlearning methods (Wu et al., 2023; Pochinkov and Schoots, 2023) involve pruning dataset-related sets of neurons, which we show can harm the model's overall capabilities. In this paper, we introduce two new machine skill unlearning methods that are *training-free* and have minimally impact the model's overall capabilities. We first observe that feed-forward layer neurons exhibit different pre-activation distributions when the model demonstrates different skills. Based on this observation, in section 3 we propose `Neuron Adjust`, which probabilistically shifts neuron pre-activation values to retain the desired skill distribution during inference through *intervention*. By considering the correlation among neurons, we further observe that neuron activation vectors cluster within different hypercubes in the feed-forward layer's key space when the model demonstrates certain skills. Building on this, we introduce `Key Space Detection (KSD)` in section 4, which detects and blocks specific skill-related activations in the key space through *abstention* by preventing query vectors from accessing the skill-specific hypercube.

Our contributions can be summarized as follows:

1. Motivated by the shift in neuron pre-activation distributions and the modularity of skill-triggering queries in the feed-forward layer's key space, we propose two novel machine skill unlearning methods, `Neuron Adjust` and `Key Space Detection`, which are scalable, training-free, and maintain the model's overall capabilities with minimal degradation.

2. Our experiments on math, code, and language skill unlearning demonstrate the effectiveness of the proposed two methods with $> 80\%$ relative performance drop on the target forgetting skill and $< 10\%$ drop on the model's general knowledge (MMLU (Hendrycks et al., 2021)) and other skills. Specifically, `Key Space Detection` achieves nearly perfect skill unlearning with negligible overall capability drop.

Table 1 compares our methods with traditional unlearning methods and the skill unlearning method `Selective Pruning` (Pochinkov and Schoots, 2023) across the dimensions of Efficiency, Performance, and Scalability.

2 Related Work & Background

Large language model machine unlearning. This line of work aims to remove the influence of specific data points and the corresponding model capabilities without retraining the model from scratch. Most previous works on machine unlearning have focused on fine-tuning-based approaches (Lu et al., 2022; Jang et al., 2023; Wang et al., 2023; Yu et al., 2023; Eldan and Russinovich, 2023; Chen and Yang, 2023; Yao et al., 2023), which become increasingly costly as models grow larger. Pawelczyk et al. (2024) introduced in-context unlearning, which provides contextual inputs to the language model during the inference stage. Despite its cost-efficiency, it lacks unlearning quality and is difficult to generalize to large-scale unlearning.

Other training-free approaches focus on pruning or removing specific sets of behavior-related neurons in the model. `DEPN` (Wu et al., 2023) is a pruning-based unlearning approach that removes neurons based on their cumulative privacy gradient. `Selective Pruning` (Pochinkov and Schoots, 2023) is another pruning-based method, which removes neurons based on their relative importance to the forgetting dataset and the retaining dataset. However, the extent to which pruning-based meth-

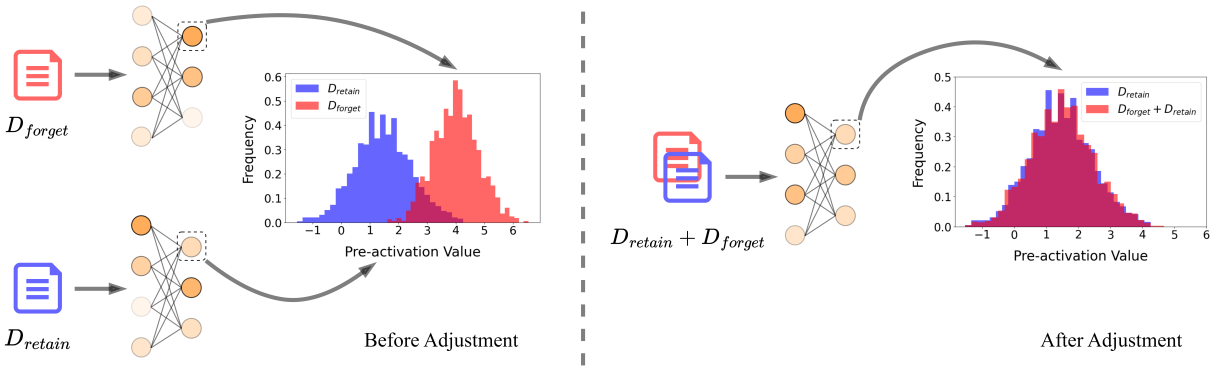

Figure 2: An overview of the `Neuron Adjust` method in section 3. Before adjusting a neuron, the neuron has different distributions under the forgetting and retaining datasets. During inference time, `Neuron Adjust` algorithm will edit neurons with large distribution shift such that its pre-activation distribution will be close to the retaining distribution.

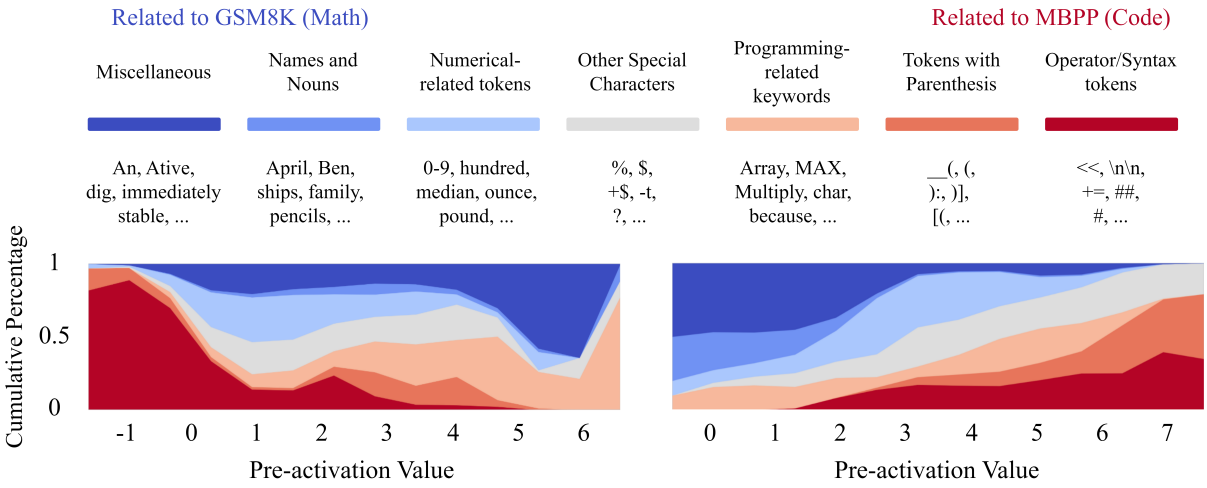

Figure 3: Category distribution over different pre-activation value ranges for neuron at layer 17, index 693 (left), and layer 0, index 13366 (right).

ods affect the model's overall capabilities remains unknown and unjustifiable.

Feed-forward neuron interpretability. This line of work focuses on the interpretation of individual neurons, meaning that individual neurons represent meaningful concepts, both in vision models (Bau et al., 2020; Hernandez et al., 2022; Oikarinen and Weng, 2023) and language models (Bills et al., 2023; Lee et al., 2023). Recent works have shown that neurons exhibit multisemanticity (Elhage et al., 2022; Bricken et al., 2023; Huben et al., 2024), with some being expressible as a linear combination of concepts (Oikarinen and Weng, 2024). By considering neuron activation vectors, we can also treat LLM's feed-forward layers (FFLs) as key-value memories, with neuron activation vectors as keys and the output of the FFLs as values (Geva et al., 2021; Meng et al., 2022).

Unlearning settings. In this paper, we focus on unlearning a specific skill or capability of a language model while retaining another. In the following sections, we denote D_{retain} as the dataset capturing the skill we want the model to retain performance, and D_{forget} as the dataset capturing the skill we want the model to forget. Our methods involve operations on the FFL in large pretrained autoregressive transformer decoder models, which takes the layer-normed input $z \in \mathbb{R}^H$ from the residual stream:

$$\mathsf{FFL}^{(l)}(z) = W^{(l)}_{\text{down}}\sigma\left(W^{(l)}_{\text{up}}z\right),$$

where z is first mapped to a higher-dimensional space by an up-projection linear transformation $W^{(l)}_{\text{up}}$ and a non-linear activation function σ to obtain neuron activations, and then mapped back to $\mathbb{R}^H$ space with a down-projection linear transformation $W^{(l)}_{\text{down}}$. Modern LLMs also utilize gated linear units (GLUs) in FFLs. Instead of directly activating each neuron, GLUs use a gating mecha-

nism to control the information flow of each neuron:

$$\texttt{FFL}^{(l)}(z) = W_{\text{down}}^{(l)} \left(\sigma \left(W_{\text{gate}}^{(l)} z \right) \odot W_{\text{up}}^{(l)} z \right),$$

where $\odot$ is element-wise vector multiplication. In the following sections, we consider $W_{\text{up}}^{(l)} z$ in traditional FFL and $W_{\text{gate}}^{(l)} z$ in GLU-FFL as the neuron pre-activations, and vectors after activation function as the key vectors (See Figure 4 for illustration), i.e. we have $v_{key}^{(l)} = \sigma \left(W_{\text{up}}^{(l)} z \right)$ and $v_{key}^{(l)} = \sigma \left(W_{\text{gate}}^{(l)} z \right) \odot W_{\text{up}}^{(l)} z$ in regular FFL and GLU-FFL respectively.

3 Inference Time Neuron Adjustment

In this section, we introduce `Neuron Adjust`, a post-hoc, training-free machine unlearning technique for large language models achieved by inference time neuron pre-activation value adjustment. An overview of `Neuron Adjust` method is shown in Figure 2. In section 3.1, we show the motivation of the method that some neurons' pre-activation distributions differ when the model demonstrates different capabilities. Based on the observation, we describe the `Neuron Adjust` method in section 3.2.

3.1 Case Study: Neuron Pre-Activation Distribution Shift

We perform a case study to show how neuron pre-activation distribution changes when the model demonstrates math and coding skills separately. We choose GSM8K (Cobbe et al., 2021) and MBPP (Austin et al., 2021) as the two datasets that characterize the math and Python coding skills of the model, and `Gemma-2b-it` (Team et al., 2024) as the subject model to study. We probe the model with the two datasets, and document each neuron's (token, pre-activation) pair, and then prompt GPT-4 (OpenAI et al., 2024) to categorize tokens into meaningful categories as in Figure 3. For the two neurons, although they are highly activated by "Programming-related keywords," "Operator/Syntax tokens," and "Tokens with Parentheses," they both show positive activations for "Numerical-related tokens." This case study demonstrates that neurons are multi-functional, and simply pruning them would be harmful to the model's overall capabilities.

3.2 `Neuron Adjust` Algorithm

Based on the observation that neuron pre-activations exhibit different distributions when the model demonstrates different skills and the polysemantic property of neurons, we propose `Neuron Adjust`, a probabilistic skill unlearning technique applied to the subject model during inference time. `Neuron Adjust` unlearns one skill of the model while retaining another skill by shifting each neuron's pre-activation from the forgetting skill distribution to the retaining skill distribution. Algorithm 1 shows the pseudo-code of `Neuron Adjust`, which mainly consists of two parts:

Algorithm 1 `Neuron Adjust` Algorithm for neuron n_i and inference time pre-activation v

Input: D_{retain}, D_{forget}, v
Probing the model with D_{retain} and D_{forget}, approximate sample mean and std:

$$n_i | D_{retain} \sim \mathcal{N}(\mu_r, \sigma_r)$$

$$n_i | D_{forget} \sim \mathcal{N}(\mu_f, \sigma_f)$$

Calculate $p_r = P(v | \mathcal{N}(\mu_r, \sigma_r))$
Calculate $p_f = P(v | \mathcal{N}(\mu_f, \sigma_f))$
if $p_r < p_f$ **then**
    $\alpha \leftarrow \frac{p_f}{p_r + p_f}$
    $v_{\text{adjust}} \leftarrow 2\mu_r - \left(\frac{v - \mu_f}{\sigma_f} \sigma_r + \mu_r \right)$ with probability α
    $v_{\text{adjust}} \leftarrow v$ with probability $1 - \alpha$
else
    $v_{\text{adjust}} \leftarrow v$
end if
Output: v_{adjust}

1. Probe the model with the forgetting and retaining datasets. For each neuron, assume that the forgetting and retaining pre-activation distributions follow a normal distribution. Approximate the means and standard deviations (stds) of these two distributions using sample pre-activation values.

2. During inference, when a neuron has a pre-activation value v, calculate the likelihood of v being sampled from each of the two distributions. If v is more likely to be sampled from the retaining distribution, keep the value. Otherwise, shift v towards the retaining distribution. Additionally, take a symmetric adjust-

ment based on the mean of the retaining distribution (this step serves as an adaptive penalty), with a probability based on how likely v is sampled from the forgetting distribution.

4 Feed-Forward Layers Key Space Hypercube Detection

One limitation of `Neuron Adjust` is that it treats each neuron individually, without considering their correlations. However, neurons often work together in a coordinated way, contributing to the model's overall behavior. In this section, we first present our observation that query vectors evoking a specific skill tend to cluster in the key space of the feed-forward layers, as described in Section 4.1. Building on this clustering phenomenon, we introduce `Key Space Detection (KSD)` in Section 4.2, a machine unlearning technique that prevents query embedding vectors from accessing designated hypercubes.

4.1 Neurons are Correlated Features in High-Dimensional Space

As the $\vec{v}_{\text{key}}$ in Figure 4(b), we define the vector before the down-projection matrix W_{down} in each FFL as the neuron activation vector, which forms the key space of each FFL. Each FFL has a unique key space. We use `llama-3-8b` as the subject language model, MBPP as the probing dataset that triggers the model's Python coding skill, and GSM8K as the probing dataset that triggers the model's grade school math problem-solving skill. We probe the model with the two training datasets and document two probing activation vector sets of the last token of each query, $\{\vec{v}^{(l)}\}_{\text{mbpp}}$ and $\{\vec{v}^{(l)}\}_{\text{gsm8k}}$, for the l^{th} FFL. We calculate the mean and std vector, $\vec{\mu}_D^{(l)}$ and $\vec{\sigma}_D^{(l)}$, of each $\{\vec{v}^{(l)}\}_D$, where

$$(\vec{\mu}_D^{(l)})_i \coloneqq \frac{1}{|D|}\sum_{j=1}^{|D|}(\vec{v}_j^{(l)})_i$$

$$(\vec{\sigma}_D^{(l)})_i \coloneqq \sqrt{\frac{1}{|D|}\sum_{j=1}^{|D|}\left((\vec{v}_j^{(l)})_i - (\vec{\mu}_D^{(l)})_i\right)^2},$$

and bound the two vector sets with hypercubes

$$\{\vec{\mu}_D \pm \alpha\vec{\sigma}_D\} \coloneqq \{\vec{u} \mid \vec{\mu}_D^{(l)} - \alpha\vec{\sigma}_D^{(l)} \prec \vec{u} \prec \vec{\mu}_D^{(l)} + \alpha\vec{\sigma}_D^{(l)}\},$$

where α is a hyperparameter that controls the size of the hypercube, and $\prec$ denotes element-wise less-than comparison. Figure 6 shows the percentage of vectors contained in the hypercube $\{\vec{\mu}_{\text{gsm8k}} \pm \alpha\vec{\sigma}_{\text{gsm8k}}\}$ when increasing α from 0 to 30 in the last layer of the model. We observe that when $\alpha = 15$, nearly all math query vector embeddings are encompassed within the hypercube. As α increases from 15 to 20, a gap forms between the math and code query clusters: all math queries remain within the hypercube, but no code queries are included. When α exceeds 20, a few code queries begin to fall within the hypercube.

We further analyze the changes in size and distance within and between the math query cluster and the code query cluster across different layers. Specifically, for each layer l, we calculate the smallest hypercube that encompass all math and code queries, respectively, and compare their volumes and the distance between their centers.

Figure 5 (left) shows the log ratio of the volume of the l^{th} layer hypercube to the volume of the first layer hypercube. As the layers get deeper, the hypercube becomes smaller, indicating denser clustering. Figure 5 (right) shows the Euclidean, Manhattan, and cosine distances between $\vec{\mu}_{\text{gsm8k}}^{(l)}$ and $\vec{\mu}_{\text{mbpp}}^{(l)}$ for each l. We observe that the Euclidean and Manhattan distances gradually increase as the layers get deeper, except for high peaks in the very first and last layers. Additionally, for most layers, the cosine distance fluctuates around 1.0, indicating the orthogonality of the two clusters.

4.2 Machine Unlearning via Key Space Detection

The idea of `Key Space Detection` is as follows: we first identify the sample mean vector $\vec{\mu}_D^{(l)}$ and the sample standard deviation vector $\vec{\sigma}_D^{(l)}$ of each FFL activation by probing the model with the forgetting dataset D. Then, we create a hypercube

$$\{\vec{\mu}_D^{(l)} \pm \alpha\vec{\sigma}_D^{(l)}\}$$

in the key space as in section 4, where α is a hyperparameter we select to balance the trade-off between the quality of forgetting and maintaining the overall capability of the model. During inference, if we detect a query vector falling within the hypercube, we abstain the model's output and replace it with a system message "`Your query is not valid.`" instead.

5 Experiments

In this section, we present experimental results of math/code skill unlearning in Section 5.1 and

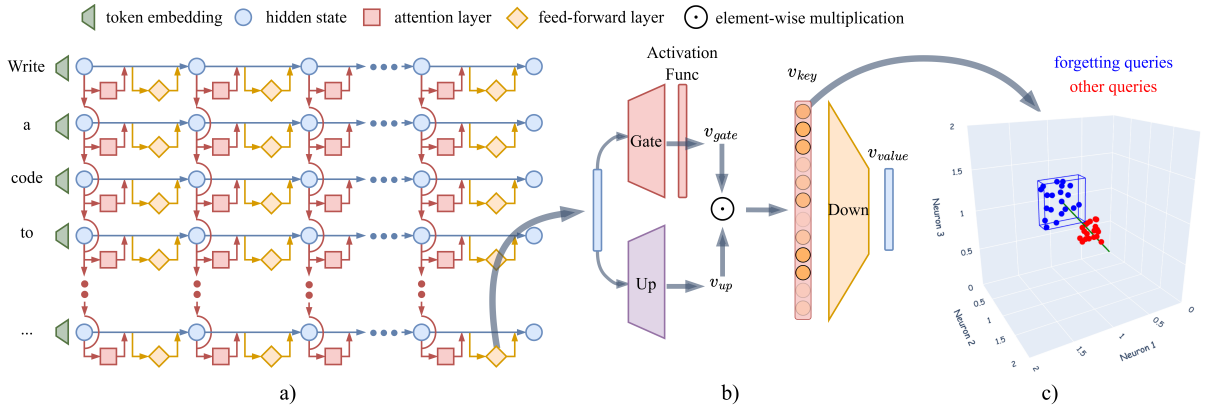

Figure 4: Overview of the `Key Space Detection` method in section 4. a) shows the structure of decoder-based large language models. b) shows the components of a GLU-based feed-forward layer in the LLM, where v_{key} is located in the key space we aim to prune. c) is an example of a 3-neuron key space. The blue hypercube is formed by $\{\vec{\mu}_D \pm \alpha\vec{\sigma}_D\}$, where $\vec{\mu}_D$ and $\vec{\sigma}_D$ are the sample mean vector and standard deviation vector of v_{key} when probing the model with the forgetting dataset. During every inference step, if we detect $v_{\text{key}} \in \{\vec{\mu}_D \pm \alpha\vec{\sigma}_D\}$, we prohibit the model from generating the output.

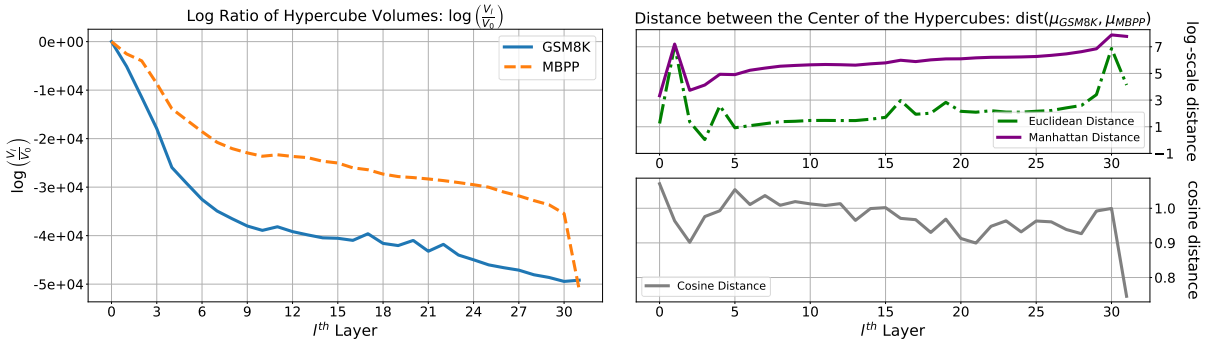

Figure 5: Relationship of the smallest hypercube containing all query vectors across layers. The left figure shows that different skill vectors cluster more tightly as the layer gets deeper. The right figure shows the distance between the centers of two hypercubes under different metrics.

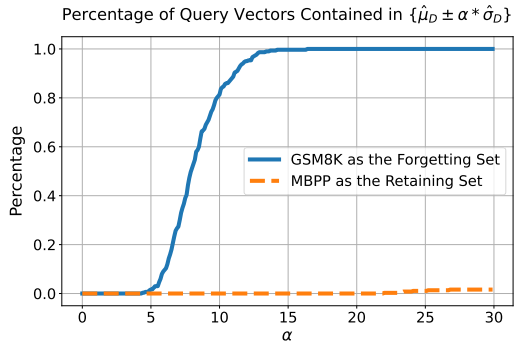

Figure 6: Percentage of Query Vectors contained in the Hypercube $\{\vec{\mu}_{\text{gsm8k}} \pm \alpha\vec{\sigma}_{\text{gsm8k}}\}$

Algorithm 2 `KSD` Algorithm for layer l

Input: size hyperparameter α, inference time activation key vector v_{key}, forgetting dataset D
Probing the model with D, estimate sample mean and std vectors $\vec{\mu}_D^{(l)}, \vec{\sigma}_D^{(l)}$.
During inference step k, current output o:
if $v_{\text{key}} \in \{\vec{\mu}_D^{(l)} \pm \alpha\vec{\sigma}_D^{(l)}\}$ **then**
    $o :=$ "`Your query is not valid.`"
    Stop inference.
else
    o += token$_k$.
    Continue with the next token inference step $k+1$.
end if
Output: o

language unlearning in Section 5.2.

We test the performance of `Neuron Adjust (NA)` and `Key Space Pruning (KSD)` on each skill unlearning task. For the `NA` method, we rank neurons by their difference in distribution mean, $\mu_f - \mu_r$, and select the top β neurons with β set to 0.5%, 1.5%, and 3.0%. For the `KSD` method, we choose size coefficient α such that `KSD` either matches or outperforms the best forgetting

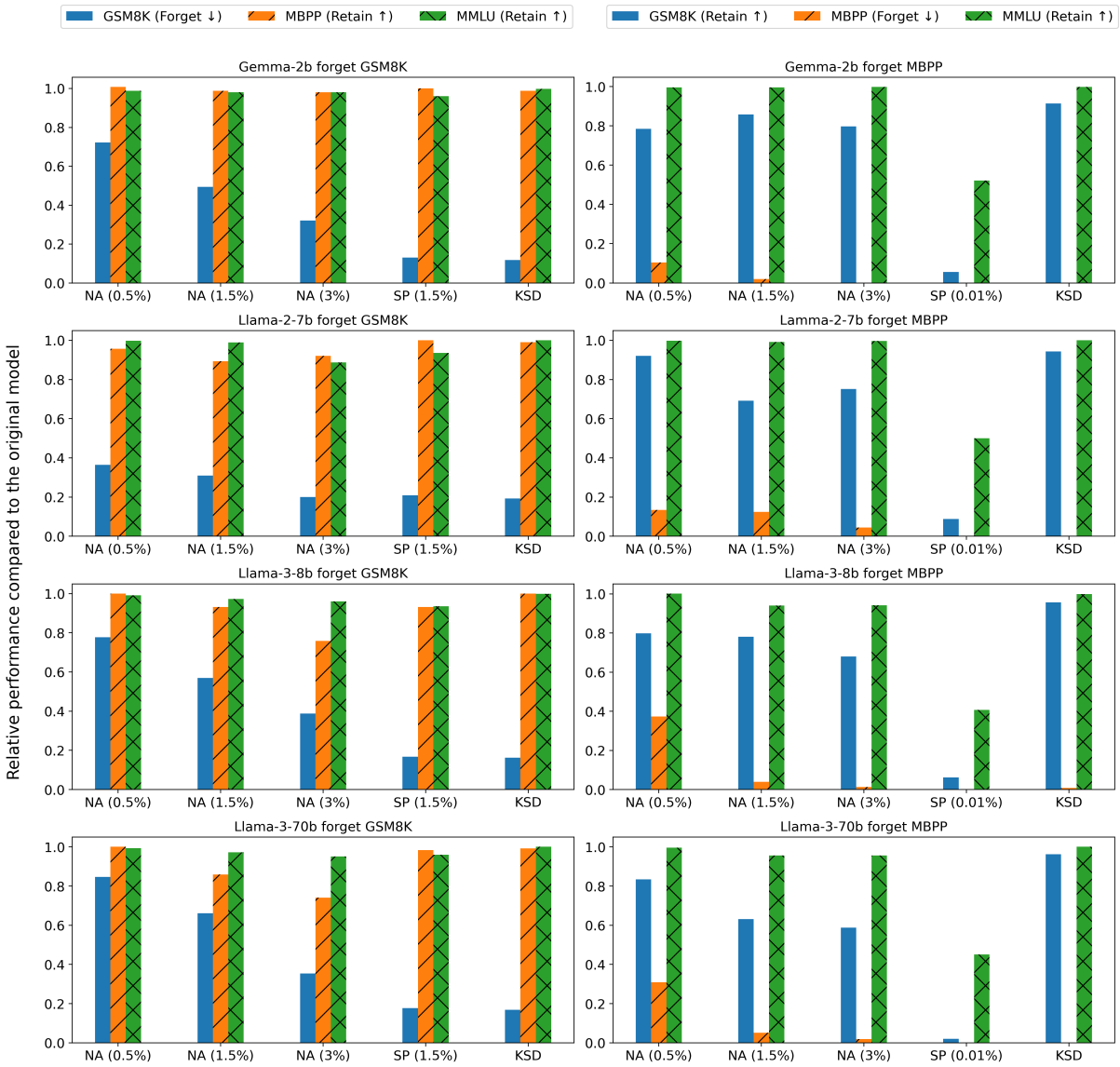

Figure 7: Performance of `Neuron Adjust` on Math/Code Skill Unlearning. On the horizontal axis, `NA`, `SP`, and `KSD` stand for `Neuron Adjust (ratio)`, `Selective Pruning (ratio)`, and `Key Space Detection`, respectively. The vertical axis represents the relative performance of the model compared to the original model after applying each unlearning method.

quality of `Neuron Adjust`. For the math/code skill unlearning task, we use `Selective Pruning` (Pochinkov and Schoots, 2023), a pruning-based skill unlearning method, as our baseline. This method prunes neurons in the FFLs based on their relative importance to each dataset. After skill unlearning, we want the model to unlearn only the specific skill while maintaining its overall capabilities. Therefore, we also evaluate the effect of each method on 5-shot MMLU accuracy. For each experiment, we run it three times and report the best result.

5.1 Math/Code Skill Unlearning

For math/code skill unlearning, we choose the training splits of MBPP and GSM8K as the forgetting datasets. The MBPP dataset contains code-based programming problems for evaluating LLMs' Python code generation abilities, while the GSM8K dataset consists of grade school-level math problems for assessing their problem-solving skills in elementary mathematics. We use these two datasets to capture the model's Python coding skills and elementary math problem-solving skills. After unlearning, we test the models on the testing split of each dataset. For Python coding skills, we also test on MBPP+ (Liu et al., 2023), which contains 35x

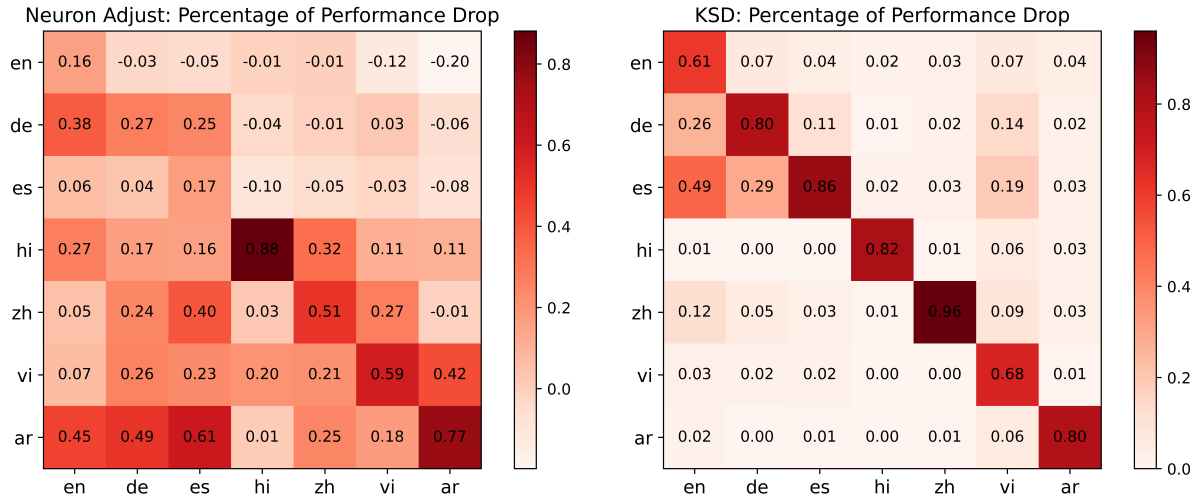

Figure 8: Results of unlearning one language in MLQA dataset while retaining the others with `Neuron Adjust 5%` and `Key Space Detection` on `llama-3-8b`. The i-th row shows the model's performance on different languages after unlearning the i-th language.

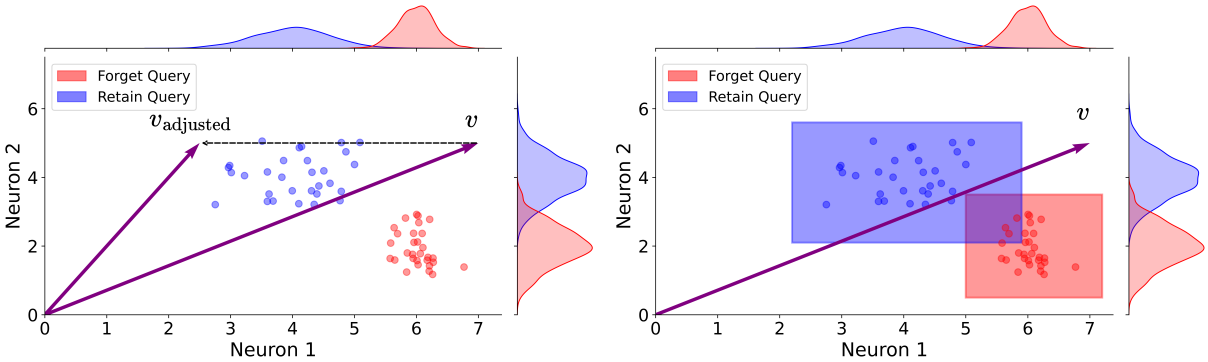

Figure 9: A case in a 2-neuron key space to explain how inference-time unrelated key vector would be affected by `Neuron Adjust` (left, get adjusted) and `Key Space Detection` (right, unaffected). In this case, an adjustment to v is unfavorable.

more Python test cases to evaluate the robustness of the unlearning process.

Figure 7 compares the performance of the unlearning methods on four models: `Gemma-2b` (Team et al., 2024), `Llama-2-7b` (Touvron et al., 2023), `Llama-3-8b`, and `Llama-3-70b` (AI@Meta, 2024). `NA`, tested at different adjustment ratios, shows that higher ratios lead to more forgetting with minimal impact on MMLU accuracy, though MBPP retention decreases slightly. `KSD` achieves the highest forgetting rates with negligible effects on MMLU accuracy and retention, proving its efficiency in unlearning while maintaining overall performance. In contrast, `SP` effectively forgets math skills and retains coding performance but produces nonsensical output when asked to forget coding skills while retaining math, even with just 0.01% neuron pruning. This suggests pruning can harm overall model capabilities, likely due to shared neurons between coding and math tasks. In this case, both `NA` and `KSD` outperform `SP`.

5.2 Language Skill Unlearning

For the language skill unlearning task, we use the MLQA (Lewis et al., 2019) dataset as our evaluation benchmark. We use each language's context data as the forgetting dataset. Figure 8 shows the results of `NA` (left) and `KSD` (right) in a heatmap view. The k^{th} row of the heatmap indicates the percentage decrease in each language's performance after forcing the model to forget the k^{th} language on the vertical axis. Ideally, we aim to maximize the diagonal entries while keeping the other entries small. From the heatmap, we observe that `NA` performs well in forgetting English (en), Spanish (es), and Hindi (hi). However, when forgetting German (de), Chinese (zh), Vietnamese (vi), and Arabic (ar), the performance of one or more languages in the re-

taining dataset also decreases significantly. This suggests that the model may utilize a shared set of neuron values when demonstrating these languages. In contrast, KSD shows a better ability to maintain the model's performance on other languages while achieving substantial forgetting quality on most of the languages. After forgetting each language, we also tested the model's performance on the MMLU task. Both methods demonstrate a performance decrease of less than 5%. We also observe that KSD consistently outperforms other methods in retaining the model's overall capability. Figure 9 illustrates the reason. For an out-of-distribution knowledge query vector v, NA may adjust some of its dimensions because it only considers operations for single neurons. In contrast, KSD considers the correlations among all neurons, prohibiting query vectors from accessing a much smaller area in the key space. Therefore, it is guaranteed to have no negative effect on out-of-hypercube queries.

6 Conclusion

In this paper, we propose two lightweight, training-free machine skill unlearning methods, `Neuron Adjust` and `Key Space Detection`, which have minimal impact on the model's overall capabilities. `Neuron Adjust` achieves unlearning by shifting the pre-activation of feed-forward layer neurons from the forgetting distribution to the retaining distribution. KSD achieves unlearning by prohibiting query vectors from accessing skill-specific key space. We evaluate our methods on unlearning math-solving, Python-coding, and comprehension skills across seven different languages with GSM8K, MBPP, and MLQA datasets respectively. Both methods show strong skill unlearning performance with minimal hurt to the model's overall capability. Experiments demonstrate the effectiveness of the two methods with > 80% relative performance drop on the target forgetting skill and < 10% drop on the model's general knowledge and other skills. Specifically, Key Space Detection achieves nearly perfect skill unlearning with negligible overall capability drop.

Our findings provide insights into how neuron activations cluster in key spaces and how these spatial properties can be leveraged for skill unlearning. These observations not only enhance our understanding of model behavior but also offer a promising direction for more targeted and interpretable unlearning techniques. We believe these insights will contribute to ongoing investigations into model interpretability, safety, and control. For example, certain knowledge, such as personal privacy data or adversarial attack inputs, may be localized in more fine-grained regions of the model's key space. Understanding how these spatial properties can be systematically utilized needs further exploration.

Acknowledgement

The authors are partially supported by National Science Foundation under Grant No. 2107189, 2313105, 2430539, Hellman Fellowship, and Intel Rising Star Faculty Award. The authors would also like to thank anonymous reviewers for valuable feedback to improve the manuscript.

Limitations

Similar to other unlearning methods, our approach can only remove capabilities that can be captured by a dataset. However, in real-world applications, a specific dataset may not always be available for every capability we wish to remove. Unlearning knowledge without a controlled dataset or unlearning out-of-distribution data points presents an interesting yet challenging problem in the field of machine unlearning. Future work could explore techniques to address this challenge. Additionally, unlearning one skill while retaining a highly dependent skill requires a more fine-grained analysis. A promising direction for future research is to leverage spatial correlations among neurons to refine unlearning mechanisms and improve selectivity. Furthermore, we have not yet identified an efficient and automatic way to determine the optimal values for the adjusting ratio and the size hyperparameter α in both methods. In the case of `Neuron Adjust`, the reduction of unintended capabilities in the model is not guaranteed. For `Key Space Detection`, although we can ensure the model's performance for out-of-hypercube queries, it may still lead to the degradation of certain unknown capabilities, as queries may cluster in a non-convex shape within the key space.

References

AI@Meta. 2024. Llama 3 model card.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. 2020. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 2023. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Jiaao Chen and Diyi Yang. 2023. Unlearn what you want to forget: Efficient unlearning for LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12041–12052, Singapore. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Ronen Eldan and Mark Russinovich. 2023. Who's harry potter? approximate unlearning in llms. *Preprint*, arXiv:2310.02238.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition. *Transformer Circuits Thread*. https://transformer-circuits.pub/2022/toy_model/index.html.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. 2022. Natural language descriptions of deep visual features. In *International Conference on Learning Representations*.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. MathPrompter: Mathematical reasoning using large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 37–42, Toronto, Canada. Association for Computational Linguistics.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2023. Knowledge unlearning for mitigating privacy risks in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14389–14408, Toronto, Canada. Association for Computational Linguistics.

Justin Lee, Tuomas Oikarinen, Arjun Chatha, Keng-Chi Chang, Yilan Chen, and Tsui-Wei Weng. 2023. The importance of prompt tuning for automated neuron explanations. *Preprint*, arXiv:2310.06200.

Patrick Lewis, Barlas Oğuz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2019. Mlqa: Evaluating cross-lingual extractive question answering. *arXiv preprint arXiv:1910.07475*.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Xiaojun Xu, Yuguang Yao, Hang Li, Kush R Varshney, et al. 2024. Rethinking machine unlearning for large language models. *arXiv preprint arXiv:2402.08787*.

Ximing Lu, Sean Welleck, Jack Hessel, Liwei Jiang, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. 2022. Quark: Controllable text generation with reinforced unlearning. *Advances in neural information processing systems*, 35:27591–27609.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36. ArXiv:2202.05262.

Tuomas Oikarinen and Tsui-Wei Weng. 2023. Clip-dissect: Automatic description of neuron representations in deep vision networks. *International Conference on Learning Representations*.

Tuomas Oikarinen and Tsui-Wei Weng. 2024. Linear explanations for individual neurons. *Preprint*, arXiv:2405.06855.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Martin Pawelczyk, Seth Neel, and Himabindu Lakkaraju. 2024. In-context unlearning: Language models as few shot unlearners. *Preprint*, arXiv:2310.07579.

Nicky Pochinkov and Nandi Schoots. 2023. Dissecting large language models. In *Socially Responsible Language Modelling Research*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth

Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Lingzhi Wang, Tong Chen, Wei Yuan, Xingshan Zeng, Kam-Fai Wong, and Hongzhi Yin. 2023. KGA: A general machine unlearning framework based on knowledge gap alignment. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13264–13276, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. 2023. DEPN: Detecting and editing privacy neurons in pretrained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2875–2886, Singapore. Association for Computational Linguistics.

Yuanshun Yao, Xiaojun Xu, and Yang Liu. 2023. Large language model unlearning. In *Socially Responsible Language Modelling Research*.

Charles Yu, Sullam Jeoung, Anish Kasi, Pengfei Yu, and Heng Ji. 2023. Unlearning bias in language models by partitioning gradients. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6032–6048, Toronto, Canada. Association for Computational Linguistics.

Table of Contents

A Appendix

A.1 Time complexity analysis

`Neuron Adjust` and `KSD` are both plug-in modules applicable to any autoregressive LLMs. For both methods, the implementation involves obtaining the mean and standard deviation of each neuron in the key space for every MLP layer by probing the subject model with the forgetting dataset. Since the forgetting dataset consists of N samples, this requires only N forward passes of the subject model. Given an autoregressive LLM with L MLP layers, each containing K neurons (where L and K are constants), we analyze the time complexity in detail:

`Neuron Adjust:` For each inference step, we need to:

- Determine whether the inference time neuron activation is more likely to be drawn from the forgetting or retaining distribution. This step is $O(KL) = O(1)$.
- Change the neuron activation value if necessary. This step is also $O(KL) = O(1)$.

Therefore, Neuron Adjust has an inference time cost of $O(1)$.

`KSD:` For each inference step, we only need to determine whether the key vector in the last MLP layer is within the forgetting hyper-rectangle. This step is $O(L) = O(1)$. Therefore, `KSD` has an inference time cost of $O(1)$.

In our experiments (real-world setting), it took less than 15 mins to implement our methods on llama-3-8b with a single V100 GPU, thus they are pretty light compared to other training-based methods.

A.2 Sequential forgetting of multiple skills

In this section we present the behavior of our methods when tasked to forget multiple skills sequentially. As shown in section 3 and 4, our methods are designed to be applied as plug-in modules. These modules can be used after each model update or fine-tuning process without affecting the model's ability to learn new tasks or skills.

`Neuron Adjust` is specifically optimized for forgetting a single skill, while `Key Space Detection (KSD)` can be extended to forget multiple skills. `KSD` works by identifying the hypercube that corresponds to each skill and determining whether the inference-time key vector falls within any of these hyper-rectangles. The computational complexity of this approach at inference time is $O(M)$, where M represents the number of skills to be forgotten.

To demonstrate the effectiveness of `KSD` in forgetting multiple skills, we conducted an additional experiment using `Llama-3-70B`, targeting the simultaneous forgetting of two tasks, MBPP and GSM8K. The results, as shown in Table 2, highlight the significant reduction in performance on the forgotten skills while maintaining general MMLU performance.

Method	GSM8K	MBPP	MBPP+	MMLU
Original	47.5%	61.1%	51.1%	64.9%
KSD	8.1%	0.5%	0.5%	64.8%

Table 2: Results of Forgetting MBPP and GSM8K with Llama-3-70B

The results demonstrate that `KSD` achieves a performance drop of over 80% on both GSM8K and MBPP tasks, effectively erasing the learned skills while leaving general knowledge tasks largely unaffected.