Mechanistic Unlearning: Locating and Erasing Information in Large Language Models

Nikolaos Kordas, Natioanal Technical University of Athens

Abstract—**The rapid adoption of Large Language Models (LLMs) has intensified concerns about privacy breaches and copyright violations, given these models' capacity to retain sensitive information encountered during training. Machine Unlearning (MU) has emerged as a solution, proposing to remove specific information without full retraining while preserving the model's overall performance. However, many unlearning techniques fail to truly erase knowledge, merely suppressing it while it remains encoded in the model's internal representations. This project investigates the efficacy of unlearning methodologies through the lens of Mechanistic Interpretability (MI)—a field dedicated to reverse-engineering the specific circuits and features that govern model behavior. I present a comprehensive survey of mechanistic unlearning techniques and propose experiments on small-scale transformer models (e.g., nanoGPT, GPT-2 Small) that enable detailed inspection of weight updates. The primary objective is to reproduce established unlearning baselines on these tractable architectures and utilize MI tools to distinguish between superficial suppression and genuine knowledge erasure.**

Index Terms—**Mechanistic Unlearning, Mechanistic Interpretability, Transformers, Large Language Models, Privacy, Sparse Autoencoders, Suppression, Erasure**

I. Introduction

THE rapid development of Large Language Models (LLMs) has revolutionized Natural Language Processing (NLP), demonstrating remarkable results in a variety of tasks [1]. However, this success is accompanied by concerns regarding data privacy and copyright compliance. Modern LLMs are trained on massive, indiscriminately scraped datasets, leading to the unintended memorization of sensitive information, such as Personally Identifiable Information (PII) and copyright-protected content [2]. This memorization process poses legal and ethical risks, particularly when models regurgitate training data during deployment [3] [4] [5]. These risks highlight the importance to selectively remove specific knowledge from a model without having to bear the cost retraining it from scratch. Consequently, effective Machine Unlearning (MU) would be a great contribution to safe and moral AI advancement.

The primary goal of MU is to erase the influence of specific data samples (the ”forget set”) without degrading the model's performance on the remaining data (the ”retain set”) or necessitating a computationally prohibitive retraining from scratch [6]. Current state-of-the-art techniques, such as Gradient Ascent and Preference Optimization, attempt to achieve this by maximizing the loss on the target data. However, recent studies suggest that these methods may not result in true erasure. Instead, they often lead to suppression, where the model learns to mask the output while the underlying knowledge remains dormant but retrievable under adversarial prompting or specific internal states [7] [8] [9].

Mechanistic Interpretability (MI) is an emerging field that seeks to reverse engineer deep learning models, decomposing complex behaviors into understandable parts like features (understandable input properties encoded in representations and activations) and circuits (sub-networks responsible for specific behaviors) [10] [11].

This paper is structured as follows. First, I present key concepts of MI establishing a way of analyzing model internals. Second, I survey prominent MU algorithms, techniques and benchmarks. Finally, I propose three experiments on small-scale transformer architectures—specifically nanoGPT, Pythia-160M, and GPT-2 Small. My goal is to reproduce unlearning results on those models and assess wether these methods achieve genuine knowledge erasure rather than superficial suppression.

II. Mechanistic Interpretability

This section is based on the comprehensive survey [11], which introduces novice readers to the concepts and techniques of mechanistic interpretability through detailed explanations and appropriate references.

A. Definition and Objects of Study

The objective of Mechanistic Interpretability is to decode a model's internal decision-making processes into a human-friendly form. This is achieved by studying its individual components and their relationships, piecing together a comprehensive explanation of the model's overall behavior.

This influential paper [12] distinguished three MI areas of study:

1) *Features:* They are properties derived from the input that have special human meaning and are embedded into the model's activations. For example, the input token ”skrew driver” may induce features like ”tool” or ”metal”. These extracted features are used by the models as fundamental units of computation for downstream tasks, such as classification, prediction, and generation [13] [14].
2) *Circuits:* It is helpful to perceive neural networks as computational graphs. This viewpoint is adopted by PyTorch [15], too, one of the most widespread frameworks for deep learning models. A circuit is a sub-graph of this

computational graph, responsible for specific Language Model's (LM) behaviors. Although there exist generalizations for this definition, I find the one presented above the most intuitive and practically useful.

3) *Universality:* This area explores wether similar features, circuits and other computational archetypes are formed across different LMs and learning tasks.

B. Features' Workflow

Survey [11] introduces a workflow to tackling interpretability tasks concerning features. The primary distinction lies in whether the analysis targets a predetermined feature believed to exist within the model, or alternatively employs an exploratory approach to discover features derived from the input.

First I will address the case of an existing target feature.

1) *Hypothesis Generation:* The proposal of the presence of a specific feature in our model's representations
2) *Hypothesis Validation:* The conducting of tests that confirm or deny the existence of the proposed feature in the LM. This can be done either by probing for the feature directly or by extracting every feature present in the model and then examining if the target is one of them.

In the case of an open-ended feature study, we make observations in our model's activations and try to interpret those as features. These two steps are called *observation* and *explanation* in [11]. This is usually done via visualization of the activations and the intervention of a human observer that will distinguish the feature present in those based in the input and context. Additionally, someone can utilize unsupervised learning techniques and perform a kind of clustering using cosine similarity or other statistical information.

C. Circuits' Workflow

The study area of circuits can also be divided into two categories. One is about interpreting an LLM's behavior and the second one about reverse engineering an LM component.

In the first case, our methodology consists of two discrete steps:

1) *LM behavior and dataset:* The researcher selects an LM behavior and an appropriate dataset. The task is to pinpoint the circuit responsible for that behavior.
2) *Computational Graph Definition:* Describe the model in terms of a computational graph M. As I mentioned in this subsection, this gives as a way to describe the circuit as a sub-graph of M.
3) *Localization:* This step is responsible for the identification of the components that contribute to a specific behavior that we want to explore. It corresponds to the localization of the nodes and the edges of the computational sub-graph that I mentioned in the step above.
4) *Interpretation:* Trying to understand how these components implement the behavior that we observe
5) *Evaluation:* The discovered circuits are evaluated in terms of faithfulness, minimality and completeness.

In the second case the researcher focuses on a single component of the LM and try to understand its role in the computations, independently of input and task. It is the same as the fourth step of the first case.

D. Universality's Workflow

Universality examines different LMs and tasks are tackled by the emergence of the same circuits and features. The established workflow [11] can be described using five steps:

1) *Scope of Universality:* It distinguishes between the study of universality in features and the one corresponding to circuits.
2) *Dimension of Variations:* Definition of the ways that LLMs studied differ (e.g. size, training data, task).
3) *Feature and Circuit Study:* Implement the workflows discussed in the two previous subsections.
4) *Evaluation of Feature Universality:* This involves the inference of the models using different datasets. Then, their activations are examined for common patterns. The similarity among activations is measured using Pearson similarity.
5) *Evaluation of Circuit Universality:* We evaluate the circuit universality by examining the common structures in components responsible for the same behaviors across the different LLMs.

E. MI Techniques and Evaluation Methods

The activations of the final layer of a transformer are called logits. To transform logits to human language, we multiply them with the unembedding matrix W_U [14]. Vocabulary Projection Methods (VPMs) take advantage of that fact and multiply W_U with intermediate representations in order to turn them to human-interpretable language, too. The underlying assumption is that all the layers of an LLM operate in the same embedding space, which is generally not true. This assumption and the application of SVD in W_U constitute the different VPMs mentioned in [11].

Another idea is to directly change the values of some intermediate representations of the transformer circuit and observe the variations in its output. The noising-base interventions aim to uncover parts of the LLM necessary to exhibition of a specific behavior by removing some parts of it during forward inference. Denoising-based interventions find sufficient components for the restoration of a model's behavior. First we remove components till an output Y stops coming up. Then we introduce the removed circuits one by one till the response emerges again [11].

This paper [14] mentions that each neuron is polysemantic. This happens due to the phenomenon of superposition [13]. It suggests that the vector space that is learned by our network is represented in it by non-orthogonal vectors. Those basis can encoded exponentially more vectors than orthogonal ones. So, we observe the same neurons firing for seemingly different concepts like a ship and a cow. SAE attempt to solve this problem by transforming the intermediate representation on to a space with a larger number of dimensions. We hope that our feature will be disentangled in this higher-dimensionality space and we will be able to distinguish them by examining the intermediate representation of the sparse autoencoder [11].

A final approach called probing suggests the use of a classificator (probe) that will distinguish pre-defined features in the activations. But this is limited by the fact that potential classification by the probe indicates only correlation and not causal relation among features and activations that caused the classification [11].

III. Machine Unlearning

A. Unlearning Methodologies

In this subsection I want to mention some methods for MU that are covered in papers [16], [17] and [18]. I present the methods examined by these authors, showcasing their main ideas, advantages and disadvantages.

Both [16] and [17] mention Gradient Ascend (GA) method. The idea is to fine tune the model by negating the training influence of the forget set D_f. This means that we should minimize the reverse log-likelihood on D_f.

$$\mathcal{L}_{GA}(\theta) = -\mathcal{L}_{train}(\theta) = \mathcal{E}_{(x,y)\ \ D_f}[-\log(p(y|x;\theta))]$$

Its main drawback is that it degrades the model's performance and it is relatively sensitive to specific hyperparameters [6] [19].

Negative Performance Optimization (NPO) is a variant of GA. It is an attempt to prevent the model from producing low quality output. Basically, we frame the forget set D_f as a preference set D_p that is constructed through human ordering of the outputs. Then, we maximize the following metric:

$$\mathcal{L}_{NPO}(\theta) = -\frac{2}{\beta}\mathcal{E}_{(x,y)\ \ D_f}\left[\log\left(\sigma\left(-\beta\log\left(\frac{p(y|x;\theta)}{p(y|x;\theta_{ori})}\right)\right)\right)\right]$$

A final GA variant mentioned in [16] is based on the idea of splitting the model's loss in two terms. One responsible for the retain set D_r and one for the forget loss on D_f. So, the metric that is going to be minimized is:

$$\mathcal{L}_u = \mathcal{L}_f + \lambda\mathcal{L}_r$$

where $\mathcal{L}_r$ is expressed in terms of KL-divergence.

Another approach is to edit the model's learned weight space and as a consequence, alter its outputs [16] [17].

$$\Delta\theta_T = \theta_T^{'} - \theta^{(0)}$$

$\Delta\theta_T$ is the difference between fine-tuned on a task and the original weights of the model. So unlearning is achieved through the following subtraction:

$$\theta = \theta^{(0)} - \lambda\Delta\theta_T\ , \lambda > 0$$

B. Unlearning Benchmarks

Some existing MU benchmarks mentioned in [17] and [16] are:

- *WHP:* Who is Harry Potter consists of a dataset made of Harry Potter's books (2.1M tokens) and Harry Potter-related synthetic text (1M tokens). It evaluates unlearning with 330 questions concerning Harry Potter, scored with GPT-4.
- *TOFU:* TOFU is a synthetic dataset of 200 fake authors and some fake books. The dataset also include 100 real author profiles and 117 world facts. The last contribute to the model's quality assessment after the unlearning process (unlearning quality is also evaluated).
- *WMDP:* It consists of 3668 questions about hazardous knowledge from different fields. It evaluates LLMs hazardous knowledge unlearning.
- *RWKU:* It is basically a forget set of 200 public figures and several forget probes. Moreover, it offers some prompts trying to cause jailbreak after poor unlearning, uncovering its poor quality (model has merely suppressed the output).
- *MUSE:* Its pipeline is similar to TOFU's but it uses a novel dataset (not present in any LLM's training) and provides better metrics.

IV. Experimental Process

The design of my experiments proceeded through three stages:

1) Selection of appropriate models amenable to training with constrained resources.
2) Identification of unlearning techniques to be investigated.
3) Establishment of an evaluation framework for each experiment.

A. Model Selection

The first step was to assess available computational resources. Given that free tiers of online platforms often impose strict time limits (e.g., 30 hours of GPU access), I designed the experimental requirements to be compatible with my local laptop environment. This ensured the experiments were not dependent on constrained external resources.

My GPU has 6GB of VRAM, imposing a strict upper bound on model size. After evaluating available options, I selected these three models:

- *gptNano:* This model consists of 85.584 parameters and its authors offer a custom dataset (tiny_shakespear) for fine tuning.
- *GPT-2:* This is the smallest version of GPT-2 model (GPT-2 Small), with 124M parameters.
- *Pythia-160M:* As its online resource proposes, this model is not suitable for deployment. But, the 154 checkpoints provided and its compact size make it suitable for testing the behavior, functionality and limitations of LLMs.

I found this useful visualization that give us a perspective on the architectural and size differences of the first two models.

B. The Experiments

The first experiment will simulate the existence of sensitive data in the training set. We will add the phrase *"The password for NTUA server is: 42_@nswer_t0_l1fe"* in 100-200 places of the dataset and fine-tune gptNano with it. This should make our model memorize the password [2]. Then I will prompt the model to confirm that the password is indeed memorized. Then, I will apply Gradient Ascent (GA) on that specific input sentence to penalize its generation. Finally, I will check if the password is still provided by the LLM when prompted and visualize it attention heads with CircuitsVis (the model is tiny, so this is feasible).

The second experiment will incorporate MI techniques to the unlearning process, too. I find SAEs extremely promising as an interpretability technique. They are trying to tackle the superposition problem [13] in a straightforward manner and may provide useful insight to the researcher, as we highlighted in the MI section of this paper. SAELens will provide the SAE framework that we will need to identify features and semantic connections such as Athens-Parthenon, London-Big Ben or Paris-Eiffel Tower. Then, I will apply Negative Performance Optimization (NPO) on the specific activations identified with the SAE. Finally, I will prompt the model with prompts that relate implicitly to the concept that it tried to unlearn. If the same activations of SAE fire again, then the unlearning was only superficial. Otherwise, the erasure will be characterized as genuine-deep.

If time constraints allow it, I will design and conduct further experiments on more unlearning and interpretability techniques, demonstrating their results on small-scale models such as those mentioned in this section.

References

[1] S. Makridakis, F. Petropoulos, and Y. Kang, “Large language models: Their success and impact,” *Forecasting*, vol. 5, no. 3, pp. 536–549, 2023. [Online]. Available: https://www.mdpi.com/2571-9394/5/3/30

[2] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramer, and C. Zhang, “Quantifying memorization across neural language models,” 2023. [Online]. Available: https://arxiv.org/abs/2202.07646

[3] N. Lucchi, “Chatgpt: A case study on copyright challenges for generative artificial intelligence systems,” *European Journal of Risk Regulation*, vol. 15, no. 3, p. 602–624, 2024.

[4] P. Hacker, A. Engel, and M. Mauer, “Regulating chatgpt and other large generative ai models,” in *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT ’23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1112–1123. [Online]. Available: https://doi.org/10.1145/3593013.3594067

[5] A. Centivany, “Mining, scraping, training, generating: Copyright implications of generative ai,” *Proceedings of the Association for Information Science and Technology*, vol. 61, no. 1, pp. 68–79, 2024. [Online]. Available: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/pra2.1009

[6] S. Liu, Y. Yao, J. Jia, S. Casper, N. Baracaldo, P. Hase, Y. Yao, C. Y. Liu, X. Xu, H. Li, K. R. Varshney, M. Bansal, S. Koyejo, and Y. Liu, “Rethinking machine unlearning for large language models,” 2024. [Online]. Available: https://arxiv.org/abs/2402.08787

[7] J. Wen, A. Zou, N. Carlini, and D. Wagner, “Adversarial prompting of unlearned language models,” 2025, final Project Report, Johns Hopkins University. [Online]. Available: https://www.example-url-where-the-report-is-hosted.edu/report

[8] X. Xu, X. Yue, Y. Liu, Q. Ye, H. Zheng, P. Hu, M. Du, and H. Hu, “Unlearning isn’t deletion: Investigating reversibility of machine unlearning in llms,” 2025. [Online]. Available: https://arxiv.org/abs/2505.16831

[9] Y. Sinha, M. Baser, M. Mandal, D. M. Divakaran, and M. Kankanhalli, “Step-by-step reasoning attack: Revealing ’erased’ knowledge in large language models,” 06 2025.

[10] N. Nanda, “Mechanistic interpretability, variables, and the importance of interpretable bases,” https://www.transformer-circuits.pub/2022/mech-interp-essay, 2022, essay available online on transformer-circuits.pub.

[11] D. Rai, Y. Zhou, S. Feng, A. Saparov, and Z. Yao, “A practical review of mechanistic interpretability for transformer-based language models,” 2025. [Online]. Available: https://arxiv.org/abs/2407.02646

[12] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter, “Zoom in: An introduction to circuits,” *Distill*, 2020, https://distill.pub/2020/circuits/zoom-in.

[13] N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, S. McCandlish, J. Kaplan, D. Amodei, M. Wattenberg, and C. Olah, “Toy models of superposition,” 2022.

[14] N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah, “A mathematical framework for transformer circuits,” 2021.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” 2019. [Online]. Available: https://arxiv.org/abs/1912.01703

[16] J. Ren, Y. Xing, Y. Cui, C. C. Aggarwal, and H. Liu, “Sok: Machine unlearning for large language models,” 2025. [Online]. Available: https://arxiv.org/abs/2506.09227

[17] J. Geng, Q. Li, H. Woisetschlaeger, Z. Chen, F. Cai, Y. Wang, P. Nakov, H.-A. Jacobsen, and F. Karray, “A comprehensive survey of machine unlearning techniques for large language models,” 2025. [Online]. Available: https://arxiv.org/abs/2503.01854

[18] P. Guo, A. Syed, A. Sheshadri, A. Ewart, and G. K. Dziugaite, “Mechanistic unlearning: Robust knowledge unlearning and editing via mechanistic localization,” 2024. [Online]. Available: https://arxiv.org/abs/2410.12949

[19] R. Zhang, L. Lin, Y. Bai, and S. Mei, “Negative preference optimization: From catastrophic collapse to effective unlearning,” 2024. [Online]. Available: https://arxiv.org/abs/2404.05868