Second-Order Information Matters: Revisiting Machine Unlearning for Large Language Models

Kang Gu
Dartmouth College
USA

Md Rafi Ur Rashid
Penn State
USA

Najrin Sultana
Penn State
USA

Shagufta Mehnaz
Penn State
USA

Abstract

With the rapid development of Large Language Models (LLMs), we have witnessed intense competition among the major LLM products like ChatGPT, LLaMa, and Gemini. However, various issues (e.g. privacy leakage and copyright violation) of the training corpus still remain underexplored. For example, the Times sued OpenAI and Microsoft for infringing on its copyrights by using millions of its articles for training. From the perspective of LLM practitioners, handling such unintended privacy violations can be challenging. Previous work addressed the "unlearning" problem of LLMs using gradient information, while they mostly introduced significant overheads like data preprocessing or lacked robustness. In this paper, contrasting with the methods based on first-order information, we revisit the unlearning problem via the perspective of second-order information (Hessian). Our unlearning algorithms, which are inspired by classic Newton update, are not only data-agnostic/model-agnostic but also proven to be robust in terms of utility preservation or privacy guarantee. Through a comprehensive evaluation with four NLP datasets as well as a case study on real-world datasets, our methods consistently show superiority over the first-order methods.

Keywords

Machine Unlearning, Data Privacy, Large Language Models, Second-Order Information

1 Introduction

Recent years have witnessed the prosperity of commercial products powered by large language models (LLMs). From the breakthrough of ChatGPT [19] to the rise of Midjourney [3], the capacity of LLMs has gone beyond typical text-related tasks such as question answering and code generation. However, ensuring responsible usage of LLM services is of utmost importance, given the lessons of previous privacy violations [14, 23]. In 2021, the first legal case concerning an AI chatbot happened in South Korea, where the chatbot *Iruda* violated the Personal Information Protection Act after generating the exact home addresses and bank account numbers of actual individuals unintentionally. Very recently, the Times sued OpenAI and Microsoft, accusing the companies of infringing on its copyrights by using millions of its articles to train ChatGPT [20].

To ensure data safety, data privacy regulations such as GDPR [22] have granted users the right to revoke the use of their data by commercial services. However, from the perspective of service providers, making a trained model forget about the knowledge of specific training samples can be much more challenging than just deleting the user data from the database. In this paper, we study the problem of how a service provider handles incoming user requests regarding the removal of their data during the life cycle of the LLM service. Retraining the model from scratch ensures the erasure of target samples, while it is extremely expensive to practice for LLMs. It raises the question of how LLM practitioners can unlearn the models with much less time and computational resources.

Recently, efficient unlearning approaches using gradient information have been explored [10, 11, 15]. Eldan *et al.* studied a novel approach to replace sensitive tokens within target samples with generic counterparts and finetune the LLMs on the modified samples [11]. It was shown that they effectively weakened the ability of Llama2 7B model to generate or recall Harry Potter-related content. Nevertheless, this approach does not generalize easily to non-fiction data since identifying private information is non-trivial and the standard of privacy differs by each individual [5]. Jang *et al.* showed that updating the model parameters by inverting the direction of gradients (gradient ascent) can erase the knowledge of target samples [15]. However, since the gradient ascent neglects the rest of the training data, it is uncontrollable how well the knowledge of the remaining data is maintained. The performance of the final model can vary drastically with different random seeds, which leaves the utility questionable. Chen and Yang [10] proposed an efficient unlearning approach by plugging lightweight unlearning layers into transformers. During training, the unlearning layers are learned to forget the requested data while the original LLM layers remained unchanged. Despite its effectiveness, this method modifies the architecture of the original models and has only been evaluated on T5 models.

Challenges. The tremendous size of parameters as well as the high non-convexity make it challenging to design unlearning algorithms for LLMs. So far, existing LLM unlearning methods either: 1) compromise the generalizability in exchange for better performance on specific data/models [10, 11], or 2) run efficiently while lacking robustness [15]. However, an ideal unlearning method should not

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–15

https://doi.org/XXXXXXX.XXXXXXX

only avoid introducing overhead (e.g. data engineering and architecture engineering) but also demonstrate robustness with respect to the effectiveness of erasure or the preservation of model utility.

Motivation. To this end, we explore novel unlearning strategies for LLMs that can satisfy the aforementioned properties. For shallow (convex) machine learning models, certified removal [13] can largely remove the influence of deleted data points and derive an upper bound of residual information. Although certified removal cannot be applied directly to DNNs due to their non-convexity, it still implies the crucial role of second-order information in retaining the knowledge of remaining data and stabilizing the unlearning process. Intuitively, the model produced by an optimal unlearning algorithm should be indistinguishable from another model that has not seen the unlearning subset, which can be measured by Shannon Information, i.e. Kullback-Leibler (KL) divergence. Golatkar *et al.* [12] showed that employing Newton update as the unlearning algorithm is sufficient to make the aforementioned KL divergence converge to zero.

Our Work. We first show that second-order information (Hessian) of LLMs can be approximated efficiently. Since the complexity of analytically calculating Hessian is quadratic with respect to the parameter size. Although it is feasible to do so for shallow models (e.g. logistic regression) [13], it quickly scales up to be prohibitively expensive when it comes to LLMs. Various approximation methods have been studied for accelerating the computation of the Hessian [12, 17, 28]. Kurtic *et al.* [17] proved that inverse empirical Fisher estimation can be accurate and scalable to the dimensionality of BERT models for pruning tasks. We adapt the inverse empirical Fisher estimation to multi-gpu setting such that it supports the scale of OPT 2.7B [31] as well as even larger models. Our empirical study shows that second-order information can be accurately approximated for unlearning.

Thus, we propose two unlearning algorithms for LLMs, namely *Fisher Removal* and *Fisher Forgetting*, which are both derived from Newton update. Compared with gradient ascent, Fisher Removal provides a stronger guarantee for the erasure of the unlearning subset while maintaining the LLM utility at a higher level. However, they both update the model parameters in an aggressive way such that the utility might degenerate noticeably after a few iterations of unlearning. To provide the utility guarantee, we further introduce Fisher Forgetting, a variant of Fisher Removal, which maintains the accuracy of LLMs even after going through multiple unlearning cycles.

Contributions We present two versions of unlearning algorithms for LLMs, which are derived from Newton update, namely *Fisher Removal* and *Fisher Forgetting*. An empirical study on four widely used NLP datasets is conducted to compare our methods with other baselines including finetuning, retraining, and gradient ascent. We further explore their mitigation effects for unintended memorization on two real-world datasets. Moreover, we uncover the relationship between unlearning and DP-SGD [1] through the study of the privacy-utility trade-off. The contributions of our work are summarized as follows:

(1) We introduce two novel unlearning strategies for LLMs and show that second-order information plays an important role in achieving robust unlearning outcomes.
(2) We extensively evaluate the capacity of each unlearning approach on four widely used NLP datasets as well as two real-world datasets. The codebase will be released to enable reproducibility.
(3) We discover that DP-SGD does not guarantee an equally optimal/suboptimal trade-off across different datasets, which implies it cannot replace the role of unlearning.
(4) We discuss the limitations such as the cost of approximating Hessian, and also point out directions for future work.

2 Preliminary

We first revisit the notion of unlearning from the perspective of KL divergence. Then we further discuss unlearning strategies for convex and non-convex models respectively. The taxonomy is defined in Table 1.

Table 1: Parameter Definition

Name	Definition
D^-	data samples to be removed
D^+	data samples left
D	full training data $D = D^+ \bigcup D^-$
θ	model parameters
l	coefficient for gradient ascent
λ	parameter for initializing Empirical Fisher
m	the number of recursions for Empirical Fisher
γ	coefficient for Fisher Removal
μ	parameter for Fisher Forgetting
σ	parameter for Fisher Forgetting

2.1 Unlearning for General Models

Let θ denote the parameters of a model trained on $D = D^- \cup D^+$ and $S(\theta, D)$ be an unlearning function that modifies the parameters such that $\theta' = S(\theta, D)$. The goal of unlearning is to ensure an adversary with access to θ' cannot reconstruct information about D^- via some readout function $f(\cdot)$.

Definition 1. Given an optimal unlearning algorithm $S_1(\theta, D)$, there is another function $S_2(\theta, D^+)$ that does not depend on D^- such that:

$$KL(\mathcal{P}(f(S_1(\theta, D))||\mathcal{P}(f(S_2(\theta, D^+)))) = 0 \quad (1)$$

Where KL stands for Kullback–Leibler divergence and $\mathcal{P}(\cdot)$ is the distribution of unlearned weights. Intuitively, the optimal unlearning outcome of $S_1(\theta, D)$ should be indistinguishable from a model that has never seen D^-. Satisfying this condition may be trivial, e.g. letting $S_1(\theta, D) = S_2(\theta, D^+) = c$ be constant. The challenge lies in how to do so while preserving as much knowledge as possible about D^+. We may define this relation from the perspective of Shannon Information [12]:

Proposition 1 Let the forgetting set D^- be a random variable such as a random sampling of D. Let Y be an attribute of interest for D^-.

$$I(Y; f(S_1(\theta, D))) \leq \mathbb{E}_{D^-}[KL(\mathcal{P}(f(S_1(\theta, D))) || \mathcal{P}(f(S_2(\theta, D^+))))] \tag{2}$$

Where $I(\cdot)$ represents mutual information. In general, we may not know what reconstruction function an adversary will use, and hence this relation should be robust against every $f(\cdot)$. therefore, the following lemma is more practical:

Lemma 1 For every $f(\cdot)$, we have:

$$KL(\mathcal{P}(f(S_1(\theta, D))) || \mathcal{P}(f(S_2(\theta, D^+)))) \leq KL(\mathcal{P}(S_1(\theta, D)) || \mathcal{P}(S_2(\theta, D^+))) \tag{3}$$

Thus, we should focus on minimizing the quantity.

$$KL(\mathcal{P}(S_1(\theta, D)) || \mathcal{P}(S_2(\theta, D^+)))$$

which guarantees robustness to any readout function. Now we can show how to derive unlearning strategies using this idea.

2.2 Unlearning for Convex Models

If the loss function is convex, e.g. quadratic, we can construct the following unlearning strategies:

$$S_1(\theta, D) = h(\mathcal{A}(D, \epsilon)) + n$$

$$S_2(\theta, D^+) = \mathcal{A}(D^+, \epsilon) + n'$$

where $\mathcal{A}(D, \epsilon)$ stands for a stochastic training algorithm with random seed ϵ, $n, n' \sim \mathcal{N}(0, \Sigma)$ is Gaussian noise and $h(\cdot)$ is a deterministic function. We have $\mathcal{P}(S_1(\theta, D) \sim \mathcal{N}(h(\mathcal{A}(D, \epsilon)), \Sigma)$ and $\mathcal{P}(S_2(\theta, D^+)) \sim \mathcal{N}(\mathcal{A}(D^+, \epsilon), \Sigma)$. Then we have the following:

$$KL(\mathcal{P}(S_1(\theta, D)) || \mathcal{P}(S_2(\theta, D^+))) \leq \frac{1}{2}\mathbb{E}_\epsilon[U^T \Sigma^{-1} U] \tag{4}$$

Where $U = h(\mathcal{A}(D, \epsilon)) - \mathcal{A}(D^+, \epsilon)$.

This means that we can derive an upper bound for the complex KL quantity $KL(\mathcal{P}(S_1(\theta, D)) || \mathcal{P}(S_2(\theta, D^+))$ by simply averaging the results of training and unlearning with different random seeds. In fact, this KL quantity converges to zero if we simply apply the Newton update. Details can be found in Appendix A.1.

2.2.1 Newton Update Without loss of generality, we aim to remove the last training sample (x_n, y_n). Let the full dataset be denoted by D and the left dataset $D^+ = D \setminus x_n$. Firstly, the loss gradient with respect to x_n can be calculated by $\Delta = \nabla \mathcal{L}(\theta x_n, y_n)$, where $\mathcal{L}$ is usually cross entropy loss and θ is the current model weight. Secondly, the Hessian of $\mathcal{L}(:, D^+)$ at θ is $H_\theta = \nabla^2 \mathcal{L}(\theta, D^+)$. Finally, the Newton update removal mechanism is as follows:

$$S_1(\theta, D) = \theta + H_\theta^{-1} \Delta \tag{5}$$

For shallow(convex) machine learning models, Newton update can perform unlearning with a bounded approximation error [13].

2.3 Gradient Ascent

An intuitive way to perform unlearning is inverting the direction of gradients. Jang *et al.*[15] first studied this approach in the context of LLMs and we also include it due to its efficiency. However, it has not yet been proven whether the gradient ascent is robust in maintaining accuracy on D^+.

As the opposite of the original training objective, gradient ascent (Algorithm 1) aims to maximize the loss function such that the knowledge of certain samples is destroyed.

Algorithm 1 Gradient Ascent for LLM

```
procedure GRADIENT_ASCENT(D⁻, θ, l)
    for i = 1,2...n do
        Δθ ← L(B_i⁻, θ)        ▷ Gather the gradient for one batch in D⁻
        θ ← θ + l * Δθ          ▷ Update parameters
    end for
end procedure
```

3 Unlearning for LLMs

However, LLMs [24] as well as other deep neural networks usually do not satisfy the convexity. To relax the assumption, we exploit the limited degree of freedom by introducing noise into the general unlearning schema as discussed in Section 2.2.

Consider noisy Newton update as follows:

$$S_1(\theta, D) = \theta + H_\theta^{-1} \Delta + (\mu \sigma^2)^{\frac{1}{4}} H_\theta^{-\frac{1}{4}} \tag{6}$$

Here, μ controls the trade-off between residual information about the forgetting subset D^-, and accuracy on the remained data D^+. δ reflects the error in approximating the SGD behavior with a continuous gradient flow.

However, to calculate Hessian analytically requires $O(d^2)$ time, where d is the size of the weight. For example, the embedding layer in GPT-2 small version has a size of $50257 \times 768 = 38597376$. Calculating the Hessian for this weight alone will require over 10^{15} operations. Not to mention that calculating the inverse of a matrix typically requires at least $O(d^{2.373})$ time [29].

3.1 Inverse Empirical Fisher

Therefore, we estimate the inverse Hessian of LLMs via inverse empirical Fisher. Specifically, we employ the Woodbury/Sherman-Morrison (WSM) formula [25]. Given a sum of $A + uv^T$ an invertible matrix A and an outer product of vectors u and v, the inverse $(A + uv^T)^{-1}$ can be exactly calculated as $A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1+v^T A^{-1} u}$. Combining the formula with empirical Fisher, we can obtain the following recursive formulation:

$$F_m^{-1}(w) = (F_{m-1}(w) + \frac{1}{m} \nabla \mathcal{L}(w, \hat{D}) \nabla^T \mathcal{L}(w, \hat{D}))^{-1} \tag{7}$$

Where m is the number of gradients used for the approximation, equal to the size of $\hat{D}$. With $F_0^{-1}(w) = \frac{1}{\lambda} I_d$, the above recursion can

be rewritten as:

$$F_m^{-1}(w) = \frac{1}{\lambda} I_d - \sum_{i=1}^{m} \frac{(F_{i-1}^{-1}(w)\nabla\mathcal{L}_i(w))(F_{i-1}^{-1}(w)\nabla\mathcal{L}_i(w))^T}{m + \nabla\mathcal{L}_i(w)^T F_{i-1}^{-1}\nabla\mathcal{L}_i(w)} \quad (8)$$

Here, λ is an initialization parameter for F_0^{-1}.

3.2 Efficient GPU Implementation

It can even be prohibitively expensive to compute and store empirical Fisher $F^{-1}(w) \in \mathbb{R}^{d\times d}$ for LLMs due to the huge size of the model weights. However, by adapting the diagonal block-wise trick [25], empirical Fisher can be accurately approximated. Specifically, we can only focus on blocks of width B along the main diagonal, which brings down the computation from quadratic $O(d^2)$ to linear $O(Bd)$.

On the implementation side, we have identified general hyperparameters $B = 48$ for the block size, and $m = 1024$ for the maximum number of recursions which yield satisfying performances. A 16GB GPU can accommodate a 125M model (e.g. GPT2-small) with $B = 48$. As for large LLMs, we have designed a multi-GPU strategy to store Fisher matrices on different GPUs, which makes this approach scalable to state-of-the-art LLMs.

3.3 Fisher Removal

Although gradient descent is highly efficient, it does not account for information from D^+. Following the idea of noisy Newton update, we propose Fisher removal, which combines second-order information obtained from D^+ with first-order information obtained from D^-. Formally, Fisher Removal is formulated as below:

$$S_1(\theta, D) = \theta + \gamma\hat{H}_\theta^{-1}\Delta$$

where unlearning rate γ is adopted to adjust the unlearning effects for LLMs.

Algorithm 2 Fisher Removal for LLM

procedure FISHER_REMOVAL($D^-, D^+, \theta, \lambda, \gamma$)
  for i=1,2...,n **do** ▷ Iterate through batches in D^-
    $\Delta\theta \leftarrow L(B_i^-, \theta)$ ▷ Gather the gradient for one batch in D^-
    $F_0^{-1} = \lambda^{-1} * I_d$ ▷ Initialize Fisher Inverse with proper shape
    for j=1,2...m **do** ▷ Iterate through batches in D^+
      $\Delta_{B_j^+} \leftarrow L(B_j^+, \theta)$ ▷ Gather gradient for one batch in D^+
      $F_j^{-1} \leftarrow update(F_{j-1}^{-1}, \Delta_{B_j^+})$ ▷ Invoke Empirical Fisher recursion
    end for
    $\hat{H}^{-1} = F_m^{-1}$
    $\theta \leftarrow \theta + \gamma\hat{H}^{-1}\Delta\theta$ ▷ Update parameters
  end for
end procedure

3.4 Fisher Forgetting

Since $\hat{H}^{-1}$ is an empirical approximation, it introduces additional uncertainty into the unlearning process, which can cause an unexpected loss of accuracy on D^+. Therefore, we present Fisher Forgetting to perturb the memory of D^- by adding Gaussian noise to the neurons [12].

$$S_1(\theta, D) = \theta + (\mu\sigma^2)^{\frac{1}{4}} H_\theta^{-\frac{1}{4}} \odot M$$

Where $M \sim \mathcal{N}(0, 1)$ is Gaussian noise, μ, σ are parameters introduced in Section 3.

Algorithm 3 Fisher Forgetting for LLM

procedure FISHER_FORGETTING($D^-, D^+, \theta, \lambda, \mu, \sigma$)
  for i=1,2...,n **do** ▷ Iterate through batches in D^-
    $\Delta\theta \leftarrow L(B_i^-, \theta)$ ▷ Gather the gradient for one batch in D^-
    $F_0^{-1} = \lambda^{-1} * I_d$ ▷ Initialize Fisher Inverse with proper shape
    for j=1,2...m **do** ▷ Iterate through batches in D^+
      $\Delta_{B_j^+} \leftarrow -L(B_j^+, \theta)$ ▷ Gather gradient for one batch in D^+
      $F_j^{-1} \leftarrow update(F_{j-1}^{-1}, \Delta_{B_j^+})$ ▷ Invoke Empirical Fisher recursion
    end for
    $\hat{H}^{-1} = F_m^{-1}$
    $M \sim \mathcal{N}(0, 1)$ ▷ Sampling Gaussian Noise
    $\theta \leftarrow \theta + (\mu\sigma^2)^{\frac{1}{4}}\hat{H}^{-\frac{1}{4}} \odot M$ ▷ Update parameters
  end for
end procedure

Property of Fisher Forgetting

Assuming $||H_\theta^{-\frac{1}{4}}||_2 \leq 1$, which can be easily satisfied if λ is set to 1 when initializing F_0^{-1}, we can derive the following property:

$$||(\mu\sigma^2)^{\frac{1}{4}} H_\theta^{-\frac{1}{4}} \odot M||_2 \leq (\mu\sigma^2)^{\frac{1}{4}}||M||_2$$

Considering Fisher Forgetting has been performed k times in a roll on a LLM parameterized by θ, the sum of all k updates to the weights can be calculated by:

$$\sum_{i=1}^{k} (\mu\sigma^2)^{\frac{1}{4}} H_\theta^{-\frac{1}{4}} \odot M_i = \sum_{i=1}^{k} \mathcal{N}(0, ((\mu\sigma^2)^{\frac{1}{4}} H_\theta^{-\frac{1}{4}}))^2)$$

Here M_i are i.i.d Gaussian variables $\sim \mathcal{N}(0, 1)$. By the property of Gaussian distributions:

$$\sum_{i=1}^{k} \mathcal{N}(0, ((\mu\sigma^2)^{\frac{1}{4}} H_\theta^{-\frac{1}{4}}))^2) \sim N(0, \sum_{i=1}^{k} ((\mu\sigma^2)^{\frac{1}{4}} H_\theta^{-\frac{1}{4}}))^2)$$

Again, with $||H_\theta^{-\frac{1}{4}}||_2 \leq 1$, the quantity can be approximated as $\mathcal{N}(0, k(\mu\sigma^2)^{\frac{1}{2}})$. This means the cumulative updates of k consecutive Fisher Forgetting can be bounded by a new Gaussian distribution

$\mathcal{N}(0, k(\mu\sigma^2)^{\frac{1}{2}})$, where $\mu, \sigma << 1$. As a result, the model performance can be largely preserved.

4 Experimental Setup

Figure 1 illustrates how a service provider handles incoming removal requests in our setting. In the life cycle of an LLM service, such requests for removal are likely to arrive stochastically. To protect users' right to be forgotten, the server may need to perform unlearning more than once, depending on the predetermined terms.

In our experiments, we consider two scenarios of unlearning: 1) perform only one unlearning cycle (128 samples), 2) perform two unlearning cycles (256 samples) consecutively.

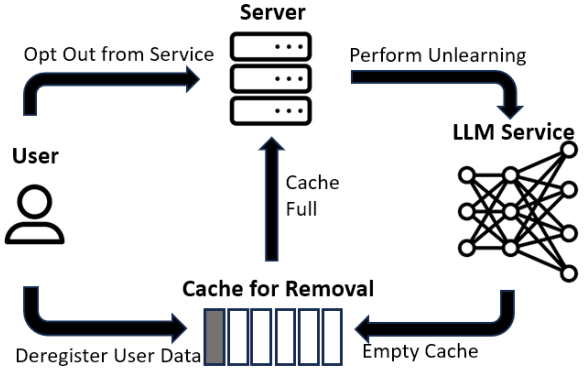


Figure 1: The cycle of unlearning of the LLM service. The model has been trained on the user database for downstream tasks. Since user requests arrive stochastically, a cache is used to store them. Once the cache is full or the waiting time has exceeded the threshold, the server will invoke the unlearning process and clean the cache to free up space. Without the loss of generality, we assume the size of the cache is 128 in our experiments.

4.1 Datasets

We evaluate LLMs on four general NLP datasets to show the trade-offs between unlearning effects and resulting performances on downstream tasks.

4.1.1 ARC-Easy. This dataset contains 5000 in grade-school level, multiple-choice scientific questions. For example, given the question of "Which piece of safety equipment is used to keep mold spores from entering the respiratory system?" and options "[A: safety goggles, B: breathing mask, C: rubber gloves, D: lead apron]", the answer is B. The dataset is split into train/val/test sets with 2251, 570, and 2376 questions respectively.

4.1.2 Piqa. It is a binary physical question-answering dataset. To solve the questions correctly, the models are required to understand physical commonsense. An example question is "To separate egg whites from the yolk using a water bottle, you should ...", and the answer is "Squeeze the water bottle and press it against the yolk. Release, which creates suction and lifts the yolk. " This dataset consists of 16113 and 1838 questions for training and validation respectively.

4.1.3 MathQA. It is a large-scale dataset of math world problems, which has provided the questions, options, answers as well as rationales. The train/val/test splits have 29837, 4475, and 2985 questions.

4.1.4 Lambada. This dataset evaluates the language modeling capacity by the task of last-word prediction. Human subjects can guess the last word with at least 50 tokens of context, which requires long-term dependencies. The dataset contains 10022 passages from BookCorpus and is further divided into 4869 for development and 5153 for testing.

4.2 Evaluation Metrics

We propose to evaluate unlearning algorithms from three dimensions: 1) the removal of selected data, 2) the preservation of model utility, and 3) the used time as in [15]. For simplicity, let θ be the parameters of an initial language model requiring unlearning, D^- be a batch of data to unlearn, D_{test} be a standalone test dataset, U be an unlearning algorithm, and θ' be the model after being unlearned.

4.2.1 Efficacy. The most important property of an unlearning algorithm is ensuring successful removal. Although certified unlearning [13] can guarantee this removal, there is no such guarantee for non-convex models. Perplexity or validation loss may not be sufficient to distinguish θ and θ' [21]. Therefore, we refer to *exposure metrics* [8] to measure the efficacy of unlearning quantitatively.

$$exp_\theta(s) = \log_2(|Q|) - \log_2 rank_\theta(s) \tag{9}$$

Here, s is a text sample and Q is a set of possible sequences with the same length. The $rank_\theta(s)$ function returns the rank of s given model θ and the set Q.

Based on the exposure metrics of a single sample, we propose $\Delta_{exp}(\theta, \theta')$ to measure the change of expected exposure values after unlearning is performed.

$$\Delta_{exp}(\theta, \theta') = \mathbb{E}_{s \in D^-}[exp_{\theta'}(s) - exp_\theta(s)] \tag{10}$$

The above metrics reflect the extent to which the unlearning algorithm U erases the set D^-.

Approximating Exposure: Given the length l, it is not practical to iterate through all combinations of tokens in the model vocabulary V, which leads to $|V|^l$ possibilities. We choose to approximate the exposure by sampling [8]. Specifically, we refer to a standalone corpus and sample a set S of sequences with length l. S can be viewed as a random sampling of the full space Q such that $|S| << |Q|$.

4.2.2 Fidelity. Although the removal of D^- matters, the unlearning algorithm U should also largely protect the performance of the model θ'. Without loss of generality, we use accuracy to evaluate model performance. Formally, let $acc_\theta(D_{test})$ stand for the accuracy of model θ on test set D_{test}, $acc_{\theta'}(D_{test})$ is expected to be close to the initial model θ. Similarly, we propose $\Delta_{acc}(\theta, \theta')$ to reflect the change of accuracy.

$$\Delta_{acc}(\theta, \theta') = acc_{\theta'}(D_{test}) - acc_\theta(D_{test}) \tag{11}$$

4.2.3 Efficiency. Finally, the unlearning algorithm should be efficient compared with retraining. As a result, we show both the time complexity and actual running time of each unlearning method in our experiments.

4.3 Unlearning Settings

Target LLMs. We test the unlearning methods on both GPT-Neo [2] and OPT LMs [31] as in [15]. For each model, three variants (125M, 1.3B, 2.7B) are included.

Table 2: The Comparison of Selected Existing Unlearning Methods

Method	Data Engineering	Architecture Engineering	Reproducible/ Open-Sourced
Retraining	x	x	✓
Finetuning	x	x	✓
Gradient Ascent [15]	x	x	✓
SISA [4]	✓	x	✓
Revise & Finetuning [11]	✓	x	x
EUL [10]	x	✓	x
Ours*	x	x	✓

Unlearning Baselines. We show the comparison of selected unlearning methods in Table 2. Only methods that can be reproducible on any dataset or LLM are considered in our experiments. However, we exclude SISA [4] because dividing the training data into disjoint shards and training a LLM on each shard will compromise the performance of LLMs, especially when the number of shards is large. As a result, in addition to the aforementioned unlearning Algorithm 1-3, we also consider retrain and finetuning as unlearning baselines:

- **Retraining:** We discard the current model and retrain the LLM using D^+ for 5 epochs. It is the benchmark for our evaluation of other baselines.
- **Finetuning:** We continue to finetune the current model on D^+ for 1 epoch. Intuitively, the model's memory of D^- will be weakened.

4.4 Hyperparameters

- For all the LLMs in our experiments, we set the coefficient for gradient ascent l as $5e-5$, which is the same as the learning rate. As for second-order methods, γ is set as $2.5e-4$. μ and σ are both set to $1e-3$.
- For empirical Fisher, λ and m are set to 1 and 1024, respectively.
- To implement exposure metrics, we refer to the European Court of Human Rights (ECHR) corpus[1]. The size of subspace S is set to 1000.

Note that γ, μ and σ are chosen through grid search where $\gamma, \mu, \sigma \in [1e-4, 1e-3]$.

5 Experimental Results

In this section, we evaluate all unlearning baselines on four datasets respectively and further highlight the overall performances of each baseline on efficacy, fidelity, and efficiency.

[1]https://archive.org/details/ECHR-ACL2019

5.1 Results & Discussions

To improve the interpretability of the results, we utilize color coding[2] to emphasize on distinctive values in Table 3, 4, 5. Due to space constraints, the results of the Lambada dataset (Table 9) can be found in the Appendix.

5.1.1 Efficacy Evaluation. With retraining as the benchmark, we observe that Fisher Removal can effectively reduce exposure values. For example, in Table 4, Fisher Removal achieves a stronger efficacy guarantee than retraining in all the scenarios, as indicated by blue cells. Fisher Forgetting is less aggressive in terms of erasing the samples, which is consistent with its design.

However, gradient ascent cannot provide an efficacy guarantee as robust as Fisher Removal. In Table 3, the removal effects of gradient ascent are apparently insufficient for GPT-NEO models compared with Fisher Removal. Meanwhile, finetuning frequently causes the samples that should be removed to be more exposed to the adversary, especially in Table 5. It shows that finetuning, which is an intuitively plausible unlearning approach, actually does not erase the information from the samples steadily.

5.1.2 Fidelity Evaluation In the case of a single unlearning cycle (128 samples), finetuning and Fisher Forgetting are relatively closer to retraining in accuracy. Fisher Removal causes slightly more degradation due to a more aggressive parameter update strategy. However, on closer inspection, gradient ascent causes multiple catastrophic outcomes as shown in Table 3 & 4, suggesting gradient ascent is the least robust regarding retaining the model fidelity.

As for the case of two unlearning cycles (256 samples), finetune and Fisher Forgetting still maintain the accuracy as well as retraining. While Fisher Removal is relatively less utility-preserving compared with Fisher Forgetting. As discussed in Section 3.4, Fisher Forgetting has the property of supporting more unlearning cycles. Therefore, We further extend the cycles of unlearning and report the fidelity in Section 9.2.

5.1.3 Efficiency Evaluation. We first formally present the time complexity of each method. Let the size of D^- be t and the size of D^+ be r respectively. Besides, recall that m is the number of recursions to estimate second-order information in Section 3.1.

For simplicity, we assume the time cost for one backpropagation pass is $O(1)$. The time complexity of each method is shown in Table 6. It is noticeable that gradient ascent is the most efficient method among them given $t << r$ can be easily satisfied. Moreover, the complexity of second-order methods is $O(tm)$. In fact, if the dataset is small such that $tm > r$, second-order methods are likely to be expensive. In contrast, when the dataset is large (which is more realistic for LLM applications), second-order methods can be relatively more efficient.

Runtime Comparison. In addition to the time complexity, we also show the runtime of GPT-NEO-125M on various datasets in Figure 2. Note that ARC ($\sim 2k$) and Lambada ($\sim 5k$) datasets are small,

[2]Blue cells stand for that corresponding methods are superior to retraining in the current settings while gray cells mark catastrophic unlearning outcomes, i.e. the accuracy drops significantly ($> 10\%$) or the exposure metric increases instead of decreasing ($\Delta_{EXP} > 0$).

Table 3: Unlearning Results on ARC Dataset

Parameter	Unlearning	128 samples				256 samples			
		NEO		OPT		NEO		OPT	
		Δ_{ACC}	Δ_{EXP}	Δ_{ACC}	Δ_{EXP}	Δ_{ACC}	Δ_{EXP}	Δ_{ACC}	Δ_{EXP}
125M	Retraining	-0.00	-0.24	-0.71	-0.05	-0.30	-0.21	+0.04	-0.10
	Finetuning	+2.00	+0.04	+0.25	-0.00	+1.20	-0.00	-0.08	-0.08
	Gradient Ascent	-0.74	-0.12	-6.35	-0.40	-10.0	-1.24	-12.20	-6.08
	Fisher Removal	-2.20	-0.37	-4.84	-0.07	-4.40	-1.24	-10.27	-5.93
	Fisher Forgetting	-0.60	-0.04	-0.67	-0.01	-1.40	-0.11	-1.18	-0.08
1.3B	Retraining	-0.80	-0.48	+0.17	-0.05	+0.51	-0.37	-1.48	-0.06
	Finetuning	-0.37	+0.02	+1.10	-0.01	+1.48	+0.02	-0.42	-0.00
	Gradient Ascent	-0.04	-0.01	-0.80	-0.02	-0.00	-0.02	-20.12	-1.22
	Fisher Removal	-0.21	-0.04	-7.93	-0.20	-2.52	-0.23	-10.79	-4.09
	Fisher Forgetting	-0.71	-0.02	-0.34	-0.02	-0.96	-0.02	-2.52	-0.03
2.7B	Retraining	-1.01	-0.53	+0.59	-0.10	+0.80	-0.47	+0.21	-0.17
	Finetuning	-0.51	-0.08	+1.14	-0.02	+1.22	-0.05	+2.28	-0.03
	Gradient Ascent	-1.52	-0.14	-21.46	-6.28	-1.10	-0.28	-21.42	-6.14
	Fisher Removal	-4.04	-0.27	-4.68	-1.40	-14.33	-3.40	-20.38	-6.03
	Fisher Forgetting	-1.52	-0.08	-0.33	-0.03	-2.23	-0.10	-1.60	-0.03

Table 4: Unlearning Results on Piqa Dataset

Parameter	Unlearning	128 samples				256 samples			
		NEO		OPT		NEO		OPT	
		Δ_{ACC}	Δ_{EXP}	Δ_{ACC}	Δ_{EXP}	Δ_{ACC}	Δ_{EXP}	Δ_{ACC}	Δ_{EXP}
125M	Retraining	-0.50	-0.03	+1.14	-0.02	-0.06	-0.14	+1.09	-0.01
	Finetuning	-0.20	+0.17	+0.27	+0.07	+1.20	+1.14	+1.30	+0.06
	Gradient Ascent	-3.50	-3.03	-10.29	-3.32	-12.9	-3.53	-10.07	-1.12
	Fisher Removal	-0.36	-1.95	-5.62	-4.84	-2.65	-4.57	-9.67	-4.28
	Fisher Forgetting	+0.07	-0.06	-0.17	-0.02	-0.85	-0.05	-1.47	-0.02
1.3B	Retraining	+0.66	-0.10	-1.52	-0.05	+1.15	-0.15	-0.76	-0.00
	Finetuning	+1.15	+0.13	-0.70	+0.05	+1.47	+0.09	-1.14	+0.03
	Gradient Ascent	-0.16	-0.14	-1.03	-0.03	-10.35	-4.86	-17.73	-4.29
	Fisher Removal	+0.22	-0.16	-5.13	-0.84	-2.28	-3.39	-7.95	-5.91
	Fisher Forgetting	-0.22	-0.03	-0.41	-0.01	-0.27	-0.01	-0.68	-0.01
2.7B	Retraining	-0.33	-0.15	-0.49	-0.07	-0.76	-0.22	-0.54	-0.02
	Finetuning	-0.44	+0.01	-1.19	+0.12	-1.19	+0.12	-0.76	+0.15
	Gradient Ascent	-0.33	-0.28	-5.11	-0.27	-6.97	-5.09	-18.49	-5.57
	Fisher Removal	-1.36	-0.79	-6.97	-1.12	-6.08	-4.77	-12.60	-3.40
	Fisher Forgetting	-0.60	-0.00	-0.45	-0.01	-0.63	-0.09	-1.14	-0.02

while Piqa ($\sim$ 16k) and MathQA ($\sim$ 30k) datasets are relatively larger.

In general, the results are consistent with previous analysis. Although second-order methods are almost as expensive as retraining on ARC and Lambada datasets, their runtime is close to finetuning on the MathQA dataset.

5.1.4 Overall Evaluation. To understand the performances more comprehensively, we aggregate the results on individual datasets by: 1) calculating the rank of each method in each scenario[3], and 2) reporting the mean and standard deviation of the ranks for each method. Their overall rankings on different dimensions are shown in Table 7. Note that their performances on the two target LLMs are similar. Therefore, we conclude the behaviors of each unlearning approach as follows:

- **Retraining** achieves the optimal balance between efficacy and fidelity as the benchmark for unlearning. However, it is the least efficient baseline as expected.
- **Finetuning** cannot provide any efficacy guarantee, therefore it is not suitable for the purpose of unlearning.

[3]The range is within 1 to 5, with the best method ranked 1_{st}

Table 5: Unlearning Results on MathQA Dataset

Parameter	Unlearning	128 samples				256 samples			
		NEO		OPT		NEO		OPT	
		Δ_{ACC}	Δ_{EXP}	Δ_{ACC}	Δ_{EXP}	Δ_{ACC}	Δ_{EXP}	Δ_{ACC}	Δ_{EXP}
125M	Retraining	+0.40	-0.03	-0.51	-0.01	+0.20	-0.02	-0.04	-0.00
	Finetuning	-0.40	+0.31	-0.61	+0.01	-0.30	+0.28	-0.27	+0.02
	Gradient Ascent	-1.90	-0.23	+0.59	-6.49	-2.50	-2.43	+1.04	-6.30
	Fisher Removal	-0.56	-1.13	+0.63	-5.88	-1.33	-4.63	+0.80	-3.31
	Fisher Forgetting	+0.11	-0.03	+0.83	-0.02	+1.38	-0.02	+0.03	-0.00
1.3B	Retraining	+1.94	-0.02	+0.47	-0.00	+1.07	-0.00	+0.64	-0.00
	Finetuning	+1.25	+0.32	+0.07	+0.03	+1.14	+0.19	+0.10	+0.03
	Gradient Ascent	-1.31	-0.11	-1.34	-3.05	-2.48	-3.87	-1.26	-1.28
	Fisher Removal	-0.30	-0.12	-0.36	-2.11	-3.69	-4.42	-1.07	-4.39
	Fisher Forgetting	-0.37	-0.02	-0.10	-0.01	-1.04	-0.00	-0.43	-0.02
2.7B	Retraining	-0.77	-0.04	+0.10	-0.03	-0.44	-0.11	+0.37	-0.02
	Finetuning	+1.13	+0.33	+0.20	+0.07	+1.13	+0.22	+0.84	+0.08
	Gradient Ascent	-1.58	-0.08	+0.07	-0.03	-3.46	-1.79	-0.43	-1.63
	Fisher Removal	-3.49	-0.71	-0.50	-0.32	-5.64	-5.21	+1.51	-6.28
	Fisher Forgetting	+0.23	-0.00	-0.16	+0.01	-0.67	-0.01	-0.23	-0.01

Table 6: Time complexity of each unlearning baseline

Method	Time Complexity
Retraining	$O(r)$
Finetuning	$O(r)$
Gradient Ascent	$O(t)$
Fisher Removal	$O(tm)$
Fisher Forgetting	$O(tm)$

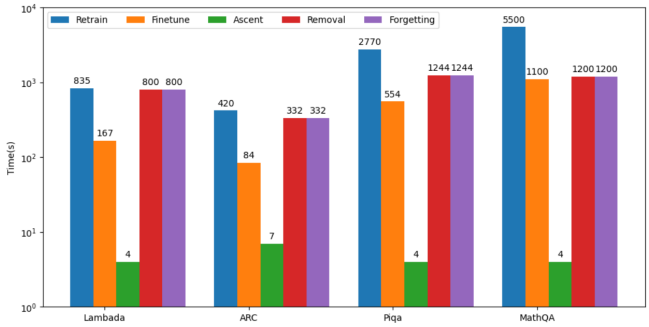


Figure 2: The runtime of each unlearning method on GPT-NEO-125M.

- **Gradient Ascent** is unstable with high variations in both efficacy and fidelity. For example, its fidelity rank on the GPT-NEO models yields a standard deviation of 2.69. However, it can still serve as a baseline due to its extreme efficiency.
- **Fisher Removal** demonstrates the strongest efficacy by outperforming both retraining and gradient ascent. However, its fidelity score is compromised because of the aggressive removal effect.
- **Fisher Forgetting** is on contrary to fisher removal. It reduces the strength of erase in exchange for more robust accuracy on the main task.

6 Unlearning for Unintended Memorization

The phenomenon where LLMs tend to memorize certain training samples is called *Memorization*. Carlini *et al.* [9] extracted hundreds of GPT-2 training samples, including sensitive information such as address, email, phone, etc. The memorization can be troublesome if LLMs are trained on highly sensitive domain data, e.g. medical. Previous work studied the memorization of canaries and unlearned canaries from the training set of an LSTM model [28]. However, the canaries can be viewed as outliers compared with the distribution of the original training set, which makes the removal easier.

To evaluate the effectiveness of the aforementioned unlearning methods against general memorization, we conduct two case studies on LLMs trained on a real-world medical dataset and an email dataset.

6.1 Datasets

Medical Transcription(MT): MT dataset[4] contains sample medical transcriptions for various medical specialties, including surgery, consultation, etc. There are 5k long medical transcriptions in total. We further split each transcription into shorter segments such that their lengths are below the input limit.

Enron Email: This dataset contains approximately 500k emails generated by employees of the Enron Corporation[5]. It was obtained by the Federal Energy Regulatory Commission during its investigation of Enron's collapse. There exist sensitive entities such as phone numbers and passwords in those emails.

[4]https://www.kaggle.com/datasets/tboyle10/medicaltranscriptions
[5]https://www.kaggle.com/datasets/wcukierski/enron-email-dataset

Table 7: Overall Rankings of all Unlearning Methods

	NEO			OPT		
Methods	Efficacy(↓)	Fidelity(↓)	Efficiency(↓)	Efficacy(↓)	Fidelity(↓)	Efficiency(↓)
Retraining	2.54+-0.91	2.21+-1.08	5+-0.00	2.92+-0.86	2.04+-1.02	5+-0.00
Finetuning	4.75+-0.52	2.00+-0.96	2+-0.00	4.88+-0.33	2.00+-1.09	2+-0.00
Gradient Ascent	2.29+-1.06	4.17+-2.69	1+-0.00	1.88+-1.05	4.42+-1.00	1+-0.00
Fisher Removal	1.33+-0.47	4.08+-1.08	3+-0.00	1.58+-0.64	3.92+-0.95	3+-0.00
Fisher Forgetting	3.71+-0.61	2.79+-1.12	3+-0.00	3.58+-0.49	2.63+-0.86	3+-0.00

6.2 Training and Inference

Training. For each dataset mentioned above, we first randomly select a subset of 100k samples and continue to train a pre-trained GPT-NEO-125M/OPT-125M on it for an extra 15 epochs.

Inference. We randomly sample a batch of 128 texts containing more than 200 tokens from each dataset for the purpose of inference. After applying an unlearning method to the trained LLM once, we perform inference on the model using the prefixes of 10 tokens. Top-k [18] sampling is applied and each prefix is repeatedly inferred 10 times.

6.3 Results

Table 8 displays how effective unlearning methods are at reducing memorization. Since the memorized samples are part of the real training set instead of canaries, the complete forgetting of them is more challenging. Although retraining can guarantee the complete removal, it takes 10X longer time than other methods. Among other unlearning baselines, Fisher Removal is the most effective, while Fisher Forgetting and gradient ascent are relatively weaker. Their performances are consistent with their efficacy scores in Table 7.

6.4 Qualitative analyses

To understand the reason behind indelible memorizations, we manually inspect the training samples in the MT dataset. As shown in Table 10, these training samples have highly similar substrings. It has been proven that the rate at which LLMs regenerate training sequences is superlinearly related to a sequence's count in the training set [16]. In order to handle these strong memorizations, data deduplication methods should be involved before the training stage. However, data deduplication falls in the category of pre-processing not unlearning.

7 DP-SGD vs. Unlearning

Differential privacy (DP) guarantees a bound of how much an individual sample can influence the output of a specific function. In the context of deep learning, DP-SGD [1] is usually applied during the training phase. We aim to study to what extent a DP-SGD-trained LLM can protect the training set as well as compare DP-SGD against unlearning.

7.1 DP-SGD

We follow the standard definition in [1]. A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies $(\epsilon, \delta)-$ differential privacy if any two adjacent inputs $d, d' \in \mathcal{D}$ for any subset of outputs $S \subseteq \mathcal{R}$ it holds that

$$\mathcal{P}[\mathcal{M}(d) \in S] \leq e^{\epsilon}\mathcal{P}[\mathcal{M}(d') \in S] + \delta$$

To investigate whether DP-SGD offers guaranteed privacy in practice, we audit DP-SGD under varying privacy budget ϵ values. Specifically, we observe the exposure scores of the 128-sample subset by querying the DP-SGD trained models. Finally, the privacy-utility trade-off is obtained.

7.2 Privacy-Utility Trade-off

Likewise, we apply the previous unlearning methods to remove the same 128 samples from the model trained without DP-SGD and report the final exposure scores. By varying the coefficients (in Table 1), we can generate a privacy-utility trade-off for each method. To this end, the trade-offs of DP-SGD as well as the unlearning approaches are summarized in Figure 3.

First, DP-SGD does not guarantee an equally optimal/suboptimal trade-off on different datasets. Although DP almost dominates other methods on the Lambada dataset, its behavior is not robust on the Piqa dataset. Second, DP-SGD introduces additional computational costs into the trivial training process, making the development of LLMs more expensive. Therefore, DP-SGD may serve as a general solution to protect training samples, but it cannot play the same role of unlearning.

8 Related Work

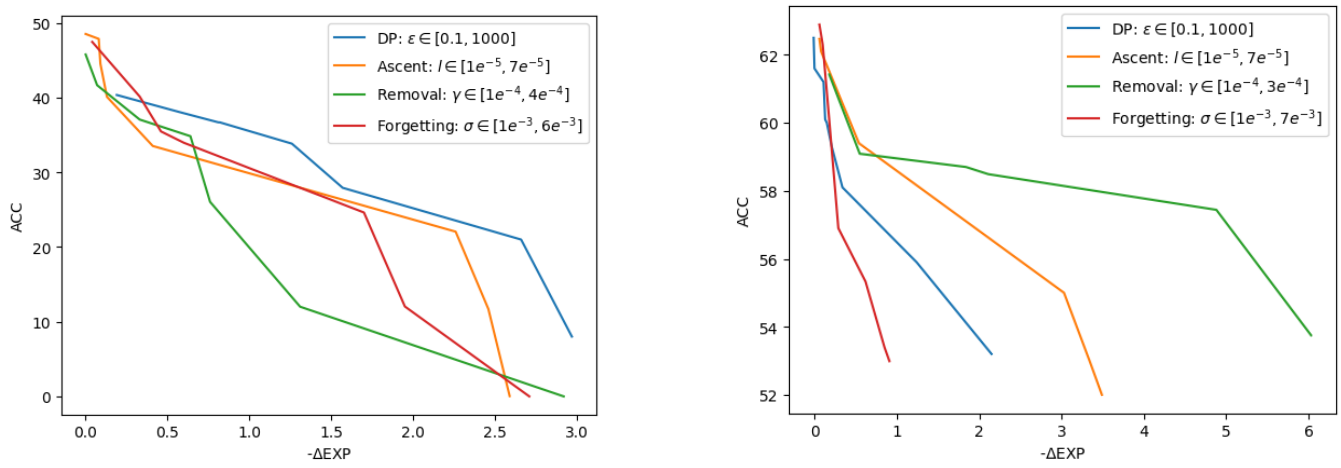
8.1 Cost of LLMs

The capacities of LLMs are usually accompanied by tremendous GPU hours. Recently, Touvron *et al.* [26] revealed that training a LLaMA-7B model requires 82432 hours on Nvidia A100 80GB GPU, not to mention larger models such as OPT-175B [31]. Given the enormous amount of resources required for a single training session, erasing target data by training from scratch is impractical and economically prohibitive for LLMs. For example, retraining a model like OPT-175B would entail not only substantial computational power but also significant time and energy costs, translating into substantial financial burdens.

For service providers, efficient unlearning methods maybe a practical solution to significantly reduce computational overhead while still observing the data regulations.

Table 8: Unlearning on MT and Enron Email Dataset

Methods	MT				Enron			
	NEO		OPT		NEO		OPT	
	# of Mem	Runtime (s)	# of Mem	Runtime (s)	# of Mem	Runtime (s)	# of Mem	Runtime (s)
No Unlearning	16	-	15	-	20	-	22	-
Retraining	0	5.69×10^4	0	5.81×10^4	0	5.05×10^4	0	5.13×10^4
Finetuning	16	3.79×10^3	15	3.87×10^3	20	3.42×10^3	22	3.43×10^3
Gradient Ascent	8	4.00×10^0	7	7.00×10^0	9	2.30×10^1	10	2.40×10^1
Fisher Removal	4	1.57×10^3	4	1.56×10^3	5	1.10×10^3	5	1.12×10^3
Fisher Forgetting	8	1.57×10^3	7	1.56×10^3	8	1.10×10^3	10	1.12×10^3



(a) Lambada **(b) Piqa**

Figure 3: Privacy-utility trade-offs of DP-SGD and unlearning approaches on (a) Lambda and (b) Piqa Dataset. The model under evaluation is OPT-125M.

8.2 Machine Unlearning

Cao *et. al* [6] firstly introduced machine unlearning for statistical query learning. They proposed a layer of summations between the learning algorithm and the training data. The learning algorithm relied only on these summations, each of which was the sum of some transformations of the training data samples. The summations were computed during the training phase and used to update the model during the unlearning phase. To forget a specific data sample, the approach simply updated a small number of summations that depended on that sample and then updated the model by resuming the learning algorithm for a few more iterations.

Certified data removal [13] was later proposed, which provided a theoretical guarantee for removal from all convex models without the need to retrain them from scratch. The key method involved using Newton update to modify model parameters and reduce the influence of deleted data points. This process was complemented by masking any residual information with random perturbations, ensuring that removed data cannot be inferred. The method was shown to be practical in various settings, providing a balance between data removal efficacy and model performance.

As for more general removal that is compatible with deep neural networks, Bourtoule *et. al* [4] proposed a sharding approach by diving the training data into multiple disjoint shards and training a model on each shard. When the request for unlearning arrived, only the constituent model associated with the specific data shard needed to be retrained. While effectively reducing the computational overhead of unlearning in machine learning models without significantly impacting the accuracy for simpler learning tasks, it introduces several challenges while working with LLMs. Assuming that each shard adequately represents the overall data distribution is not realistic. Moreover, training multiple models on large shards while tuning the hyperparameter for each constituent model also demands significant computational resources. This involves not just the processing power but also memory requirements, as each constituent model needs to be stored, trained, and optimized separately before integration.

More recently, the unlearning of LLMs has attracted increasing attention. Yao *et al.* [30] applied gradient ascent to unlearn undesirable behaviors in LLMs by aligning the model using negative examples. Wang et al. [27] presented a novel approach that focused on aligning the knowledge gap between models trained on different

datasets, suggesting that unlearned models should treat the deleted data as unseen data. However, acquiring high-quality data from the same distribution can be particularly challenging, especially when the availability of such data is limited. In addition, we aim to generally remove the effects of unlearning subsets, rather than focusing on aligning LLMs with human preferences or knowledge gaps.

Chen and Yang [10] introduced an efficient unlearning framework designed to update LLMs effectively without the need for full retraining. This was achieved through the integration of lightweight unlearning layers into transformers, which were specifically trained on a set of data designated for forgetting, enabling it to effectively discard specific knowledge. Elden *et al.* [11] proposed to replace sensitive tokens of Harry Potter-related text with generic counterparts and finetune the LLMs on the modified samples. Nevertheless, the replacement process is non-trivial and does not easily generalize to other domains. Jang *et al.* [15] showed that applying gradient descent directly can make LLMs forget about target samples, while no guarantee is provided for model utility.

The aforementioned work has only explored the unlearning task with first-order information, which is reasonable due to its extreme efficiency. However, most of them either require data engineering [11, 27, 30] or specific architectural design [10], which limits the generalizability. We explore novel unlearning approaches for LLMs relying on second-order information. We show that Hessian can be estimated with affordable computational cost and our methods demonstrate superior robustness with respect to gradient ascent.

9 Discussions

9.1 Interpreting Unlearning via Weight Distributions

Recall that the distribution of the model parameters after an optimal unlearning algorithm should be indistinguishable from a model that has never seen the unlearning subset (eq. 2.1). Specifically, the KL quantity $KL(\mathcal{P}(S_1(\theta, D))||\mathcal{P}(S_2(\theta, D^+)))$ should be small (or close to zero in ideal setting). To better illustrate the effects of unlearning, we visualize the weight of the last layer in GPT-NEO 125M after performing different unlearning algorithms in Figure 4.

First, it is noticeable that the weight distribution after applying the gradient ascent deviates far from that of retraining, leading to a large KL divergence of 1176792. However, our second-order-based methods achieve unlearning without pushing the weight distribution too far from retraining. The quantitative KL divergence values by Fisher Removal and Fisher Forgetting are merely 425 and 4174 respectively, which are consistent with the histograms.

9.2 Extended Unlearning Cycles

We have discussed the property of Fisher Forgetting in Section 3.4, here we compare the performance of each baseline when unlearning cycles are further extended.

The GPT-NEO-125M performance curves after consecutive unlearning cycles are displayed in Figure 5. As indicated by the theoretical analysis, Fisher Forgetting maintains the model utility robustly on both the Lambada and Piqa datasets. However, it is not guaranteed that other methods will retain a similar level of utility. Specifically, we observe that gradient ascent causes model performance to drop rapidly as the unlearning subset grows in both scenarios.

9.3 Onion Effect

Carlini *et al.* [7] revealed and analyzed an *Onion Effect* of memorization: removing the "layer" of outlier points that are most vulnerable to a privacy attack exposes a new layer of previously safe points to the same attack. In the context of machine unlearning, we explore how the removal of certain data samples impacts the rest of the training data.

We conduct a control experiment on the ARC dataset using GPT-NEO-125M. Specifically, we first obtain the exposure score of each training sample by querying the model before unlearning. Second, samples with exposure scores higher than 6 are removed. Finally, we retrain the model using the modified training dataset and recalculate the exposure scores. As shown in Figure 6, some initially "safe" samples become more exposed after the most exposed samples are removed, which validates the existence of the onion effect.

The results imply that machine unlearning may contradict the goal in the case of eliminating the most vulnerable samples. However, in general, in cases where we assume the unlearning subset is randomly sampled, no significant shift of exposure distributions is observed after unlearning. Figure 7 demonstrates the stability of exposure distributions after unlearning.

10 Limitations & Future Work

10.1 Limitations

10.1.1 *Calculating Hessian is Relatively Costly*

We have shown that the time complexity of estimating Hessian can be reduced to the product between the size of the unlearning subset and the number of recursions. However, it is still relatively costly compared to the calculation of first-order information. For LLMs, extra GPU space may be required to store the Hessian matrices. Both the time complexity and space complexity could be improved by optimizing the current estimation formula.

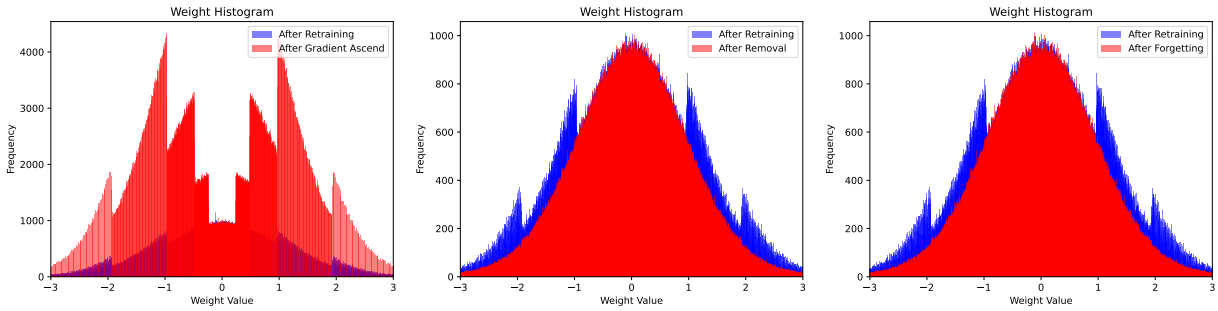
10.1.2 *Erasing Unintended Memorization is Challenging*

Although previous work showed that canaries inserted into the training set can be easily erased from an LSTM model [28], it should be considered a special case of memorization since canaries are outliers in the regular training set. Our evaluation of general memorization reveals that even Fisher Removal cannot guarantee complete forgetting in one shot. However, repeatedly applying the unlearning algorithm may compromise the model's utility. Therefore, it requires further investigation such as data duplication [16] to address indelible memorization.

10.2 Future Work

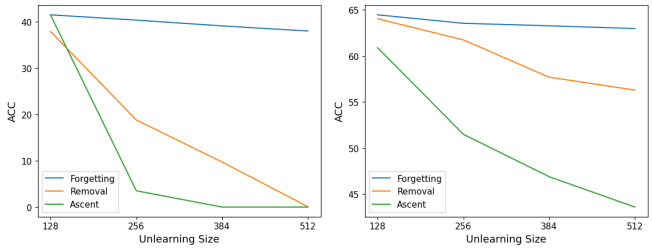
10.2.1 *Towards Better Privacy-Utility Trade-off*

If we consider that retraining represents the optimal trade-off between privacy and utility, Fisher Removal is biased toward the side of privacy, while Fisher Forgetting favors the side of utility. It still remains challenging to design an unlearning algorithm that strongly guarantees both properties. Future work can focus on



(a) KL = 1176792 **(b) KL = 425** **(c) KL = 4174**

Figure 4: Weights distribution of the last layer of GPT-NEO (125M) fine-tuned on ARC dataset. # of bins for plotting the histograms is set to 40k. (a), (b) and (c) display the distribution after applying Gradient Ascent, Fisher Removal, and Fisher Forgetting respectively.



(a) Lambada **(b) Piqa**

Figure 5: Model utility curves after extended unlearning cycles. Both (a) and (b) are generated by GPT-NEO-125M model.

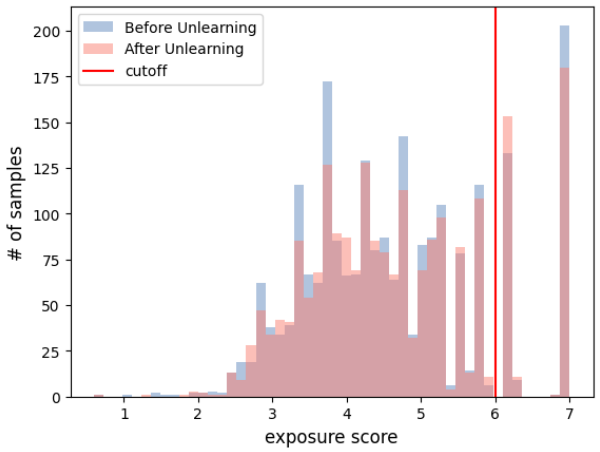


Figure 6: Onion effect on GPT-NEO-125M. In the ARC dataset, we set the cutoff threshold as 6 and remove all the samples with higher exposure scores. After retraining the model, some left samples have their exposure scores increase past the threshold.

improving the privacy-utility trade-off of unlearning. One possible direction is combining different unlearning strategies such as our Fisher Removal and Fisher Forgetting.

10.2.2 *Robustness against Extended Unlearning Cycles*

In practice, an unlearning algorithm should preserve the utility of the LLM even after multiple unlearning procedures. Otherwise, the service provider may still resort to retraining frequently due to the loss of utility, which contradicts the purpose of unlearning. In our evaluation, only Fisher Forgetting possesses this desired robustness. However, this property is imperative for any unlearning algorithm to be useful in real-world applications. Future work should focus on developing and refining unlearning methods that can reliably preserve the utility of LLMs across multiple unlearning cycles.

10.2.3 *Unlearning Larger LLMs & Better Evaluation Metrics*

Due to the hardware constraints, we have mainly performed experiments on GPT-NEO/OPT models up to 2.7B. Since our unlearning methods are model-agnostic, we encourage future work to explore the unlearning process of larger LLMs such as LLama2 7B and Falcon 7B. In addition, we largely follow the previous work to measure how much an LLM still remembers the unlearning subset by exposure score. However, the setup of subspace and the process of iterating through all the samples can be time-consuming. Future work might investigate more efficient/elegant metrics to audit the outcome of unlearning.

11 Conclusion

In this paper, we propose two novel unlearning algorithms for large language models, namely Fisher Removal and Fisher Forgetting. Although they are both derived from Newton update, Fisher Removal provides a stronger guarantee for the erasure of the unlearning subset compared with gradient ascent. Fisher Forgetting is designed to preserve the utility of LLMs against multiple unlearning processes. Through a comprehensive evaluation of four NLP datasets, we demonstrate that second-order information can make the unlearning outcomes more robust compared to using only first-order information.

We further reveal that unlearning can mitigate unintended memorizations, but there still exist indelible memorizations that require additional prevention approaches like data deduplication.

Finally, we uncover the relationship between unlearning and DP-SGD through the privacy-utility trade-off. The results suggest that DP-SGD does not always guarantee an equally optimal/sub-optimal trade-off across different datasets, which implies that it cannot play the same role of unlearning. Instead, DP-SGD may serve as a general solution to protect training samples.

For future work, we suggest improving existing unlearning methods towards better privacy-utility trade-offs as well as robustness against multiple unlearning cycles. Our unlearning algorithms can also benefit from adopting more efficient Hessian approximation formulas.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745* (2022).

[3] Ali Borji. 2022. Generated faces in the wild: Quantitative comparison of stable diffusion, midjourney and dall-e 2. *arXiv preprint arXiv:2210.00586* (2022).

[4] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 141–159.

[5] Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. 2022. What does it mean for a language model to preserve privacy?. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*. 2280–2292.

[6] Yinzhi Cao and Junfeng Yang. 2015. Towards Making Systems Forget with Machine Unlearning. *2015 IEEE Symposium on Security and Privacy* (2015), 463–480. https://api.semanticscholar.org/CorpusID:5945696

[7] Nicholas Carlini, Matthew Jagielski, Chiyuan Zhang, Nicolas Papernot, Andreas Terzis, and Florian Tramer. 2022. The privacy onion effect: Memorization is relative. *Advances in Neural Information Processing Systems* 35 (2022), 13263–13276.

[8] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*. 267–284.

[9] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*. 2633–2650.

[10] 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*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 12041–12052. https://doi.org/10.18653/v1/2023.emnlp-main.738

[11] Ronen Eldan and Mark Russinovich. 2023. Who's Harry Potter? Approximate Unlearning in LLMs. *arXiv preprint arXiv:2310.02238* (2023).

[12] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. 2020. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9304–9312.

[13] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. 2019. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030* (2019).

[14] Melissa Heikkilä. 2022. What does gpt-3 "know" about me? (2022).

[15] Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2022. Knowledge unlearning for mitigating privacy risks in language models. *arXiv preprint arXiv:2210.01504* (2022).

[16] Nikhil Kandpal, Eric Wallace, and Colin Raffel. 2022. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*. PMLR, 10697–10707.

[17] Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. 2022. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. *arXiv preprint arXiv:2203.07259* (2022).

[18] Dong Li, Ruoming Jin, Jing Gao, and Zhi Liu. 2020. On sampling top-k recommendation evaluation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2114–2124.

[19] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, et al. 2023. Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology* (2023), 100017.

[20] Cade Metz. 2023. OpenAI Says New York Times Lawsuit Against It Is 'Without Merit'. (2023).

[21] Fatemehsadat Mireshghallah, Kartik Goyal, Archit Uniyal, Taylor Berg-Kirkpatrick, and Reza Shokri. 2022. Quantifying privacy risks of masked language models using membership inference attacks. *arXiv preprint arXiv:2203.03929* (2022).

[22] Council of the European Union. 2018. General Data Protection Regulation(GDPR). (2018). https://gdpr-info.eu/

[23] Jasmine Park. 2021. The first case where the personal information protection act was appliedto an ai system. (2021).

[24] Md Rafi Ur Rashid, Vishnu Asutosh Dasu, Kang Gu, Najrin Sultana, and Shagufta Mehnaz. 2023. FLTrojan: Privacy Leakage Attacks against Federated Language Models Through Selective Weight Tampering. *arXiv preprint arXiv:2310.16152* (2023).

[25] Sidak Pal Singh and Dan Alistarh. 2020. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems* 33 (2020), 18098–18109.

[26] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[27] 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)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 13264–13276. https://doi.org/10.18653/v1/2023.acl-long.740

[28] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. 2021. Machine unlearning of features and labels. *arXiv preprint arXiv:2108.11577* (2021).

[29] Virginia Vassilevska Williams. 2014. Multiplying matrices in O(n 2:373) time. https://api.semanticscholar.org/CorpusID:124418960

[30] Yao Yuanshun, Xu Xiaojun, and Liu Yang. 2023. Large Language Model Unlearning. *arXiv preprint arXiv:2310.10683* (2023).

[31] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).

A Appendix

A.1 Proof of Newton Update

Let the loss be $L_D(\theta) = L_{D^+}(\theta) + L_{D^-}(\theta)$ and assume that the loss function $L(\cdot)$ is quadratic. The optimization algorithm $\mathcal{A}_t(D, \epsilon)$ is given by the gradient information at time step t with random initialization. Consider the following unlearning process:

$$h(\theta) = \theta + e^{-H_1 t} e^{H_2 t} V_1 + e^{-H_1 t}(V_1 - V_2) - V_2 \tag{12}$$

Where $H_1 = \nabla^2 L_D(\theta), H_2 = \nabla^2 L_{D^+}(\theta), V_1 = H_1^{-1} \nabla L_D(\theta)$, and $V_2 = H_2^{-1} \nabla L_{D^+}(\theta)$. Then $h(\theta)$ satisfies $h(\mathcal{A}_t(D, \epsilon)) = \mathcal{A}_t(D, \epsilon)$ for all random initializations and all times t. In particular, let $S_1(\theta, D) = h(\theta)$, which removes all the information of D^-:

$$KL(\mathcal{P}(S_1(\theta, D)) || \mathcal{P}(\mathcal{A}(D^+, \epsilon)) = 0$$

Note that when $t \to \infty$, that is, when the optimization algorithm has converged, the unlearning process reduces to the simple Newton update.

Table 9: Unlearning Results on Lambada Dataset

Parameter	Unlearning	128 samples				256 samples			
		NEO		OPT		NEO		OPT	
		Δ_{ACC}	Δ_{EXP}	Δ_{ACC}	Δ_{EXP}	Δ_{ACC}	Δ_{EXP}	Δ_{ACC}	Δ_{EXP}
125M	Retraining	-1.61	-0.30	-4.85	-0.43	+0.85	-0.24	-1.34	-0.48
	Finetuning	-1.98	-0.27	-2.48	+0.33	+0.58	+0.38	-1.84	+0.22
	Gradient Ascent	-0.74	-0.12	-8.79	-0.13	-38.66	-1.99	-48.90	+0.35
	Fisher Removal	-4.30	-0.31	-14.05	-0.64	-23.23	-0.99	-48.90	-1.68
	Fisher Forgetting	-0.68	-0.14	-1.43	-0.14	-1.82	-0.04	-4.04	-0.09
1.3B	Retraining	-1.20	-0.49	+1.89	-0.76	-0.72	-0.81	+1.96	-0.46
	Finetuning	-0.89	-0.25	+1.26	+0.52	-0.68	+0.27	+ 1.13	+0.38
	Gradient Ascent	-0.31	-0.28	-28.74	-0.23	-58.43	+0.55	-58.72	-4.16
	Fisher Removal	-2.48	-0.50	-2.72	-0.48	+0.47	-1.81	-24.78	-1.51
	Fisher Forgetting	-0.60	-0.05	-1.30	-0.13	-0.52	+0.05	-4.28	-0.18
2.7B	Retraining	-0.86	-1.10	-1.12	-0.29	+1.32	-1.24	-0.27	-1.25
	Finetuning	-0.92	+0.16	-0.33	+0.10	-2.60	+0.17	-2.71	+0.10
	Gradient Ascent	+1.98	-0.41	-61.63	-3.76	-59.60	-1.76	-61.63	-4.44
	Fisher Removal	-2.97	-1.10	-2.61	-0.25	-56.79	-3.01	-60.62	-1.18
	Fisher Forgetting	-1.07	-0.05	-0.60	-0.05	-2.37	-0.06	-3.22	-0.11

Table 10: Indelible Memorizations in MT dataset

The incision site was closed with 7-0 nylon., The patient tolerated the procedure well and was transferred to the recovery room in stable condition with Foley catheter in position.
She was subsequently consented for a laparoscopic appendectomy.,DESCRIPTION OF PROCEDURE: , After informed consent was obtained, the patient was brought to the operating room, placed supine on the table.
PROCEDURE PERFORMED: , Modified radical mastectomy.,ANESTHESIA: , General endotracheal tube.,PROCEDURE: ,After informed consent was obtained, the patient was brought to the operative suite and placed supine on the operating room table.

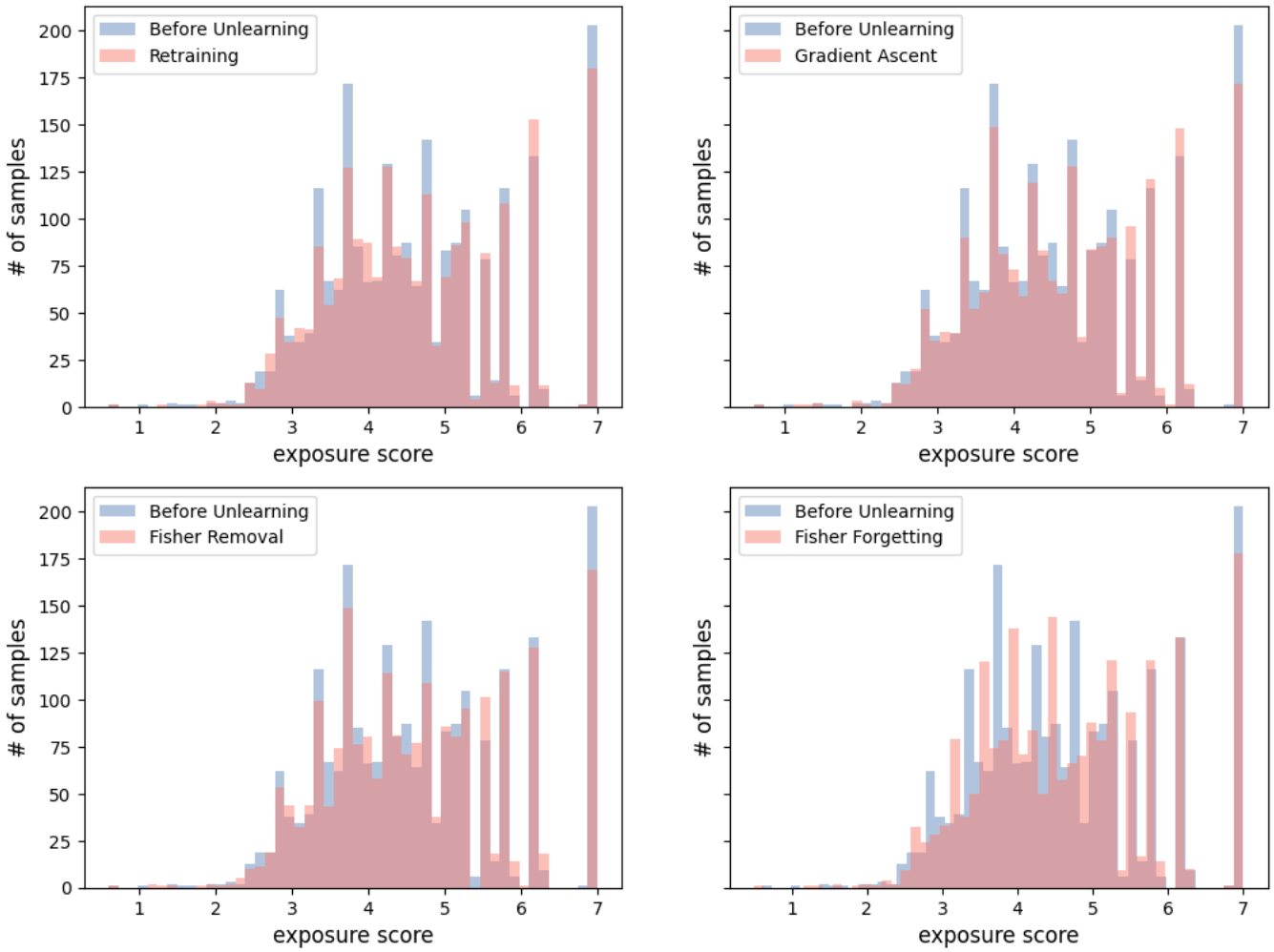


Figure 7: The distributions of exposure scores on ARC before/after unlearning.