UNLEARNING AS MULTI-TASK OPTIMIZATION: A NORMALIZED GRADIENT DIFFERENCE APPROACH WITH ADAPTIVE LEARNING RATE

Anonymous authors
Paper under double-blind review

ABSTRACT

Unlearning techniques have been proposed as a cost-effective post-training way to remove undesired knowledge learned by large language models (LLMs). However, existing methods often fail to effectively unlearn the targeted information or cause a significant drop in model performance. In this paper, we frame machine unlearning as a multi-task optimization problem to balance this tradeoff – one task maximizes forgetting loss, while the other minimizes retaining loss. We introduce a novel unlearning method, Normalized Gradient Difference (NGDiff), which guarantees Pareto optimality upon convergence. Specifically, NGDiff dynamically normalizes task gradients, enabling the model to unlearn targeted forgetting data while preserving utility on the retaining set. We also identified that unlearning methods are sensitive to learning rate and integrate an automatic learning rate scheduler that selects the locally optimal learning rate to stabilize and accelerate the convergence. Experiments with various LLMs demonstrate that NGDiff outperforms state-of-the-art unlearning methods on the *TOFU* and *MUSE* datasets.

1 INTRODUCTION

Large language models (LLMs) are trained on a huge collection of data from various sources, such as books and websites. For example, LLAMA3 is pre-trained on over 15T tokens Dubey et al. (2024), while Falcon Penedo et al. (2023b), OLMo Groeneveld et al. (2024) are each pre-trained on over 1T tokens. However, this extensive data pre-training raises serious concerns about data risks for the following reasons: (1) Certain data sources, despite being publicly available, contain potentially harmful or sensitive content, such as nudity, personal information, copyright-protected information, etc. (2) LLMs can memorize these data during training, and then re-generate them. For example, Exhibit J from The New York Times Times (2023) shows examples of outputs from GPT-4 that "contain large spans that are identical to the actual text of the article from The New York Times," which are copyrighted; some attacks can extract an individual's name, email address, phone number, and physical address from LLMs Carlini et al. (2021). Furthermore, the risk of memorization increases with the size of the model (see Figure 1 and 3 in Carlini et al. (2022)), exposing bigger models with higher utility to greater risks of data memorization.

Although retraining the models by removing the problematic data can resolve this issue, this approach is not feasible given that LLMs cost millions of dollars and take months to train, and the cost escalates each time new data are identified as problematic. Recently, researchers have developed a number of machine unlearning methods, which are applied after the models have completed the training and memorized the data. Specifically, we divide the data into two classes: the retaining set (R), on which the model can memorize and have high performance, and the forgetting set (F), which the model should not memorize. The goal of unlearning is to continue training the model in a way, so that knowledge from the forgetting set is effectively removed, and that the unlearned model behaves similarly to one that is retrained solely on the retaining set.

Existing machine unlearning methods are formulated primarily as optimization techniques aimed at minimizing memorization through the language model loss Liu et al. (2023); Chen et al. (2024); Liu et al. (2024b). For instance, the Gradient Ascent (GA) method seeks to maximize the language

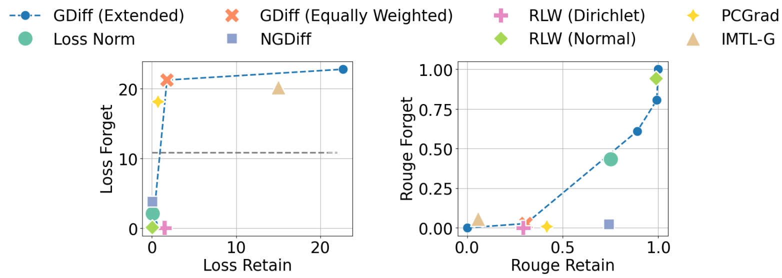

Figure 1: Loss values and ROUGE scores on the forgetting and retaining data from the *TOFU* dataset using different unlearning methods on the Phi-1.5 language model. We apply the *extended GDiff* with various coefficients (see (2), $0 \leq c \leq 1$) and connect the results with a blue dashed line. We denote MTO methods as different markers, and use a grey dashed line to represent the loss of random guess.

model loss on the forgetting data (i.e., minimize the negative language model loss). However, this approach can also negatively affect the utility of the model. To address this, the Gradient Difference (GDiff) method selects a subset of the training data as a retaining set, minimizing the sum of the negative language model loss on the forgetting set and the standard language model loss on the retaining set. This approach has been empirically shown to effectively preserve the model's performance Liu et al. (2022); Maini et al. (2024). Similarly, Negative Preference Optimization (NPO) Zhang et al. (2024) assigns a lower likelihood to forgetting data, thereby balancing the unlearning process while maintaining model utility.

However, in our preliminary experiments, we observe two key issues preventing these methods from being practically applied. First, balancing retaining and forgetting losses is difficult. In Figure 1, we observe a trade-off between the performance on R and F, where some methods fail to unlearn F (points in the upper-right corner of the left figure), and some do not maintain utility in R (points in the bottom-left corner of the left figure). The blue dotted line in Figure 1 further illustrates the trade-off in GDiff by sweeping a hyper-parameter $c \in [0, 1]$, which is used to balance the losses on the forgetting and retaining data (see Eq. (2)). Picking an appropriate c to balance the two terms is often challenging. Secondly, the optimization methods for unlearning are usually sensitive to the learning rate. As illustrated in Figure 2, even for the same algorithm, various learning rates lead to substantial changes in the ROUGE scores and loss values, making the unlearning methods unstable and difficult to use in practice.

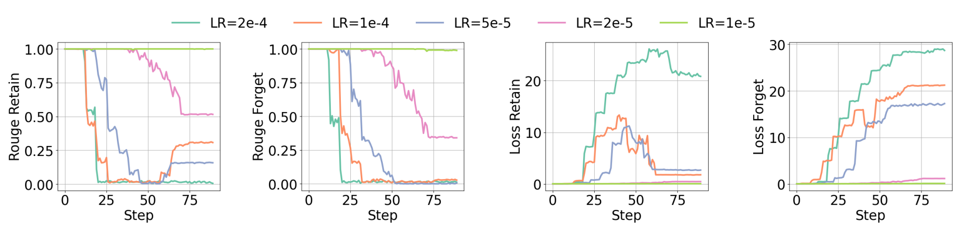

Figure 2: ROUGE scores and loss values during unlearning with vanilla GDiff (equally weighted), under different learning rates to which the unlearning performance is highly sensitive.

In this paper, we formulate the unlearning as a multi-task optimization (MTO) problem Chen et al. (2021); Xin et al. (2022): we aim to minimize the loss (or maximize the utility) on the retaining set and maximize the loss on the forgetting set, simultaneously. To solve this two-task problem, we leverage the rich literature of multi-task methods to achieve the Pareto optimality of two tasks, e.g. IMTL Liu et al. (2021), GradNorm Chen et al. (2018a), RLW Lin et al., PCGrad Yu et al. (2020), and scalarization Boyd & Vandenberghe (2004). In particular, we leverage the linear scalarization as a simple and strong candidate among all multi-task methods, which minimizes a linearly weighted average of the task losses (see (2)). This is motivated by two reasons: theoretically, linear scalarization is guaranteed to be Pareto optimal (see Lemma 2); empirically, it outperforms or at least is on par with other MTO methods in a variety of language and vision experiments Xin et al. (2022).

Contributions We focus on and extend the linear scalarization for machine unlearning as follows:

- We formally formulate machine unlearning as a multi-task optimization problem and establish the Pareto optimality for scalarization-based unlearning methods (including ours), as demonstrated in Theorem 3.
- Through the lens of multi-task optimization (two-task specifically), we propose a novel unlearning method *NGDiff*, which leverages the gradient norms to *dynamically* balance the forgetting and retaining tasks. In particular, NGDiff improves both tasks simultaneously and monotonically under the proper learning rate, as demonstrated in Theorem 5.
- We integrate the automatic learning rate from GeN Bu & Xu (2024), which adaptively and dynamically selects the learning rate based on the Hessian information, and thereby achieving stable convergence.
- We empirically showcase the effectiveness of our method through extensive experiments on multiple datasets, different LLMs and vision models. For example, on the TOFU dataset, our method achieves 40% higher model utility while maintaining comparable unlearning performance with the Llama2-7B model.

1.1 Related Work

This work is closely related to machine unlearning methods, multi-task optimization, and learning-rate-free techniques, which are discussed in Section 2.2 and Appendix D with more details.

2 Unlearning as Multi-task Optimization

In this section, we connect machine unlearning to multi-task optimization (MTO), specifically the two-task optimization: denoting the retaining set as R and the forgetting set as F, we study

$$\min_{\theta} L_{\mathrm{R}}(\boldsymbol{\theta}) \& \max_{\theta} L_{\mathrm{F}}(\boldsymbol{\theta})$$

where L is the cross-entropy loss and $\boldsymbol{\theta}$ is the model parameters.

To optimize two tasks, it is critical to consider the Pareto optimality in Definition 1 as MTO may have infinitely many solutions.

Definition 1 (Pareto optimality in unlearning)**.** *For two models $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, if $L_{\mathrm{R}}(\boldsymbol{\theta}) \geq L_{\mathrm{R}}(\boldsymbol{\theta}')$ and $L_{\mathrm{F}}(\boldsymbol{\theta}) \leq L_{\mathrm{F}}(\boldsymbol{\theta}')$ with at least one inequality being strict, then $\boldsymbol{\theta}$ is dominated by $\boldsymbol{\theta}'$. The model $\boldsymbol{\theta}$ is Pareto optimal if it is not dominated by any other models.*

In practice, the machine unlearning solutions generally exhibit a trade-off between the performance on R and F (see Figure 1): without the unlearning, both R and F have high performance and high memorization; in order to forget F, one may unlearn other general knowledge such as grammar rules, which oftentimes sacrifices the performance on R.

We will illustrate several MTO methods and show that the Pareto optimality is guaranteed upon the convergence of these methods.

2.1 Static linear scalarization

The scalarization method — linearly combining multiple tasks into a single reweighted task, is arguably the most widely-used MTO method. It defines the linear scalarization problem (LSP) as

$$\mathrm{LSP}(\boldsymbol{\theta}; c) = c \cdot L_{\mathrm{R}}(\boldsymbol{\theta}) - (1-c) \cdot L_{\mathrm{F}}(\boldsymbol{\theta}) \tag{1}$$

where at each iteration, c is fixed and $\boldsymbol{g}_{\text{static}}(c)$ lies within the linear span of per-task gradients as shown in Figure 3 (yellow area),

$$\boldsymbol{g}_{\text{static}}(c) = \frac{\partial \mathrm{LSP}}{\partial \theta} = c\boldsymbol{g}_{\mathrm{R}} - (1-c)\boldsymbol{g}_{\mathrm{F}}. \tag{2}$$

Remark 2.1. *We term the static linear scalarization as the **extended GDiff** in this work. Some special cases in unlearning are GD ($c = 1$), GA ($c = 0$), and vanilla GDiff ($c = 0.5$, equally weighted), which is proposed in Liu et al. (2022).*

A nice property of linear scalarization is the Pareto optimality at the convergence, which we state in Lemma 2 for the static c and later extend to Theorem 3 for the dynamic c_t in Section 2.2.

Lemma 2 (restated from Xin et al. (2022))**.** *For any* $0 < c < 1$*, the model* $\boldsymbol{\theta}^*_{LSP}(c) = argmin_\theta \text{LSP}(\boldsymbol{\theta}; c)$ *is Pareto optimal.*

Proof of Lemma 2. We show that the solution of LSP cannot be a dominated point, and therefore it must be Pareto optimal. Consider a solution $\boldsymbol{\theta}^* = \text{argmin}_\theta \text{LSP}(\boldsymbol{\theta}; c)$, and suppose it is dominated by some $\boldsymbol{\theta}'$, i.e. $L_\text{F}(\boldsymbol{\theta}^*) \leq L_\text{F}(\boldsymbol{\theta}'), L_\text{R}(\boldsymbol{\theta}^*) \geq L_\text{R}(\boldsymbol{\theta}')$ with at least one inequality being strict. This contradicts that $\boldsymbol{\theta}^*$ is minimal as $cL_\text{R}(\boldsymbol{\theta}^*) - (1-c)L_\text{F}(\boldsymbol{\theta}^*) > cL_\text{R}(\boldsymbol{\theta}') - (1-c)L_\text{F}(\boldsymbol{\theta}')$. □

Lemma 2 suggests[1] that we can sweep through $c \in [0, 1]$ and construct the Pareto frontier after sufficiently long training time as in Figure 1. However, while any c leads to a Pareto optimal point, the solution may be useless: e.g. perfect memorization on (R, F) that fails to unlearn is also Pareto optimal. Next, we investigate different choices of c by extending the static scalarization in (2)

2.2 DYNAMIC SCALARIZATION

In deep learning, the loss is minimized iteratively by the gradient method:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t[c\boldsymbol{g}_\text{R}(\boldsymbol{\theta}_t) - (1-c)\boldsymbol{g}_\text{F}(\boldsymbol{\theta}_t)]$$

which extends (2) to a broad range of methods if we set $c = c_t$,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \boldsymbol{g}_\text{UN}(\boldsymbol{\theta}_t; c_t) \text{ where } \boldsymbol{g}_\text{UN}(\boldsymbol{\theta}; c_t) := c_t \boldsymbol{g}_\text{R}(\boldsymbol{\theta}) - (1-c_t)\boldsymbol{g}_\text{F}(\boldsymbol{\theta}) \quad (3)$$

Importantly, instead of defining $\boldsymbol{\theta}^* = \text{argmin}_\theta \text{LSP}$ at the loss level, we can define it at the gradient level based on the stationary condition of the training dynamics, i.e. $\boldsymbol{g}_\text{UN}(\boldsymbol{\theta}^*) = \boldsymbol{0}$.

In light of (3), we summarize some unlearning and MTO methods below

1. Gradient descent (GD on R), $c = 1$
2. Gradient ascent (GA on F), $c = 0$
3. Gradient difference (vanilla GDiff), $c = 0.5$
4. Loss normalization (LossNorm), $\frac{c_t}{1-c_t} = \frac{L_\text{F}}{L_\text{R}}$
5. RLW, $c_t = \frac{e^{\lambda_1}}{e^{\lambda_1} + e^{\lambda_2}}$ with $\lambda_i \sim N(0, 1)$
6. PCGrad, $\frac{c_t}{1-c_t} = 1 + \frac{\boldsymbol{g}_\text{F}^\top \boldsymbol{g}_\text{R}}{|\boldsymbol{g}_\text{R}|^2}$
7. IMTL-G, $c_t = \boldsymbol{g}_\text{F}^\top (\frac{\boldsymbol{g}_\text{F}}{||\boldsymbol{g}_\text{F}||} - \frac{\boldsymbol{g}_\text{R}}{||\boldsymbol{g}_\text{R}||}) / (\boldsymbol{g}_\text{F} - \boldsymbol{g}_\text{R})^\top (\frac{\boldsymbol{g}_\text{F}}{||\boldsymbol{g}_\text{F}||} - \frac{\boldsymbol{g}_\text{R}}{||\boldsymbol{g}_\text{R}||})$

Despite the different designs of $\{c_t\}$, we show in Theorem 3 that all $\boldsymbol{\theta}^*(\{c_t\})$ are Pareto optimal, including our NGDiff to be introduced in Section 3.2.

Theorem 3. *For any* $\{c_t\}$ *with* $0 \leq c_t \leq 1$ *that converges as* $t \to \infty$*, the model* $\boldsymbol{\theta}^*(\{c_t\}) = \lim_{t\to\infty} \boldsymbol{\theta}_t$ *in* (3) *is Pareto optimal.*

Proof of Theorem 3. Denoting $c = \lim_t c_t$, then (3) gives that $\boldsymbol{g}_\text{UN}(\boldsymbol{\theta}_t) = c_t \boldsymbol{g}_\text{R}(\boldsymbol{\theta}_t) - (1 - c_t)\boldsymbol{g}_\text{F}(\boldsymbol{\theta}_t) \to c\boldsymbol{g}_\text{R}(\boldsymbol{\theta}^*) - (1-c)\boldsymbol{g}_\text{F}(\boldsymbol{\theta}^*) = \boldsymbol{0}$ as $t \to \infty$. Note $\boldsymbol{\theta}^*(\{c_t\})$ is equivalent to the LSP solution $\boldsymbol{\theta}^*_\text{LSP}(c) = \text{argmin}_\theta \text{LSP}(\boldsymbol{\theta}; c)$ as the latter has the same stationary condition, which is Pareto optimal by Lemma 2. □

[1]We note that Lemma 2 is only applicable to the global minimum of LSP, which is not always achievable in deep learning. Therefore, this result has its limitations and requires empirical validation.

3 Unlearning with normalized gradient difference

While Theorem 3 shows the Pareto optimality of $\boldsymbol{\theta}^*$ as $t \to \infty$, it does not shed insight on the convergence through intermediate steps $\boldsymbol{\theta}_t$. Put differently, although many MTO and unlearning methods are all Pareto optimal upon convergence, they may converge to different Pareto points at different convergence speed. Therefore, it is important to understand and control their algorithm dynamics in order to maintain high performance for R throughout the training. To be specific, the dynamics is determined by their choices of $\boldsymbol{g}_{\text{UN}} \in \mathbb{R}^d$ and $\eta_t \in \mathbb{R}$ in (3).

In this section, we propose to use gradient normalization for $\boldsymbol{g}_{\text{UN}}$ and automatic learning rate for η_t, so as to achieve stable convergence, effective unlearning, high retaining utility, without manually tuning the learning rate.

3.1 Loss landscape of unlearning

Applying the Taylor expansion on (3), we can view the local loss landscape as a quadratic function.

$$\begin{aligned} L_{\text{R}}(\boldsymbol{\theta}_{t+1}) - L_{\text{R}}(\boldsymbol{\theta}_t) &= -\eta_t \boldsymbol{g}_{\text{R}}^\top \boldsymbol{g}_{\text{UN}}(c_t) + \frac{\eta_t^2}{2} \boldsymbol{g}_{\text{UN}}^\top \mathbf{H}_{\text{R}} \boldsymbol{g}_{\text{UN}} + o(\eta_t^2) \\ L_{\text{F}}(\boldsymbol{\theta}_{t+1}) - L_{\text{F}}(\boldsymbol{\theta}_t) &= -\eta_t \boldsymbol{g}_{\text{F}}^\top \boldsymbol{g}_{\text{UN}}(c_t) + \frac{\eta_t^2}{2} \boldsymbol{g}_{\text{UN}}^\top \mathbf{H}_{\text{F}} \boldsymbol{g}_{\text{UN}} + o(\eta_t^2) \end{aligned} \tag{4}$$

Here $\mathbf{H} = \frac{\partial^2 L}{\partial \boldsymbol{\theta}^2}$ is the Hessian matrix, which empirically gives $\boldsymbol{g}_{\text{UN}}^\top \mathbf{H} \boldsymbol{g}_{\text{UN}} > 0$ and renders L_{R} and L_{F} locally and directionally convex along the gradients. This allows the existence of a minimizing learning rate to be characterized in Section 3.3. We visualize the loss landscape in Figure 4 and observe that the quadratic functions in (4) are well-fitted in most iterations.

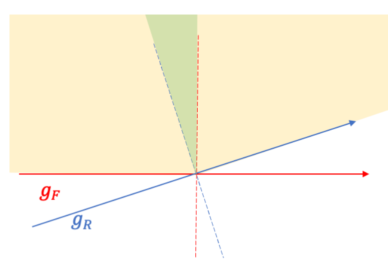

Figure 3: Demonstration of gradient space in 2-dimension. $\boldsymbol{g}_{\text{F}}$ is the forgetting gradient and $\boldsymbol{g}_{\text{R}}$ is the retaining gradient, each with a perpendicular dashed line. Yellow area is the linear span (2) by scalarization. Green area is positively correlated to $\boldsymbol{g}_{\text{R}}$ and negatively correlated to $\boldsymbol{g}_{\text{F}}$ by (5), whereas NGDiff always stays within at each iteration.

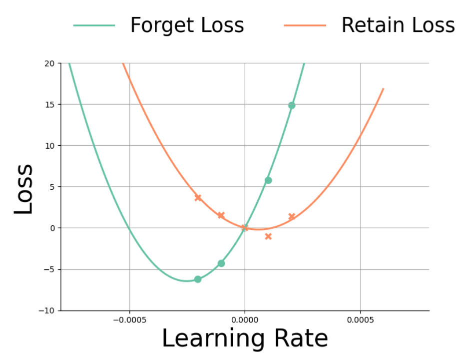

Figure 4: Loss values of retaining and forgetting sets with respect to different learning rates. Markers are $L_{\text{R}}(\boldsymbol{\theta}_t - \eta \boldsymbol{g}_{\text{R}})$ and $L_{\text{F}}(\boldsymbol{\theta}_t - \eta \boldsymbol{g}_{\text{F}})$ using the *TOFU* dataset on Phi-1.5 at step 10, on which the curves are fitted as quadratic functions.

3.2 Normalized Gradient Difference

In order for L_{F} to increase as well as L_{R} to decrease, we want to construct $\boldsymbol{g}_{\text{UN}}$ such that

$$\boldsymbol{g}_{\text{R}}^\top \boldsymbol{g}_{\text{UN}}(c_t) \geq 0 \geq \boldsymbol{g}_{\text{F}}^\top \boldsymbol{g}_{\text{UN}}(c_t). \tag{5}$$

To satisfy (5) , we propose to dynamically set

$$c_t = \frac{1/||\boldsymbol{g}_{\text{R}}||}{1/||\boldsymbol{g}_{\text{R}}|| + 1/||\boldsymbol{g}_{\text{F}}||} \Longrightarrow \boldsymbol{g}_{\text{NGDiff}}(\boldsymbol{g}_{\text{R}}, \boldsymbol{g}_{\text{F}}) := \frac{\boldsymbol{g}_{\text{R}}}{||\boldsymbol{g}_{\text{R}}||} - \frac{\boldsymbol{g}_{\text{F}}}{||\boldsymbol{g}_{\text{F}}||}$$

where a common factor is omitted. In words, we normalize the retaining and forgetting gradients[2], respectively, and state that (5) is satisfied at all iterations in Lemma 4.

[2] We illustrate in Appendix A that NGDiff is critically different and simpler than GradNorm (Chen et al., 2018a).

Lemma 4. *$g_{\text{NGDiff}}(g_{\text{R}}, g_{\text{F}})$ satisfies (5) for any $g_{\text{R}} \in \mathbb{R}^d$ and $g_{\text{F}} \in \mathbb{R}^d$.*

In Theorem 5 (see proof in Appendix F), we can leverage Lemma 4 to claim the the local loss improvement under appropriate learning rate, which will be implemented adaptively in Section 3.3.

Theorem 5. *Consider $\theta_{t+1} = \theta_t - \eta g_{\text{NGDiff}}$. (1) Unless g_{R} is exactly parallel to g_{F}, for any sufficiently small learning rate η, there exist two constants $\epsilon_{\text{R},1} = o(\eta), \epsilon_{\text{F},1} = o(\eta)$ such that*

$$L_{\text{R}}(\theta_{t+1}) - L_{\text{R}}(\theta_t) < \epsilon_{\text{R},1}, \text{ and } L_{\text{F}}(\theta_{t+1}) - L_{\text{F}}(\theta_t) > \epsilon_{\text{F},1}.$$

(2) If additionally $g_{\text{NGDiff}}^\top \mathbf{H}_{\text{R}} g_{\text{NGDiff}} > 0$ and $g_{\text{NGDiff}}^\top \mathbf{H}_{\text{F}} g_{\text{NGDiff}} > 0$, then for any learning rate $0 < \eta < \frac{2 g_{\text{R}}^\top g_{\text{NGDiff}}}{g_{\text{NGDiff}}^\top \mathbf{H}_{\text{R}} g_{\text{NGDiff}}}$, there exist two constants $\epsilon_{\text{R},2} = o(\eta^2), \epsilon_{\text{F},2} = o(\eta^2)$ such that

$$L_{\text{R}}(\theta_{t+1}) - L_{\text{R}}(\theta_t) < \epsilon_{\text{R},2}, \text{ and } L_{\text{F}}(\theta_{t+1}) - L_{\text{F}}(\theta_t) > \epsilon_{\text{F},2}.$$

Remark 3.1. *The condition that $g_{\text{NGDiff}}^\top \mathbf{H} g_{\text{NGDiff}} > 0$ in part (2) of Theorem 5 may not always hold in deep learning. However, it empirically holds in most iterations across models and datasets in our experiments (c.f. our Figure 4 and Figure 2 in Bu & Xu (2024)), and we can stablize the training by not updating η when the condition fails.*

To interpret Theorem 5, we view $\epsilon \approx 0$ as η is generally small (say $\eta \sim 10^{-4}$ in our experiments), and hence NGDiff is optimizing on R and F simultaneously. Visually speaking, Lemma 4 constrains NGDiff's gradient to stay in the green area in Figure 3 unless $g_{\text{F}} \parallel g_{\text{R}}$, whereas other methods do not explicitly enforce (5) and may consequently harm the retaining utility.

3.3 Automatic learning rate adaption for unlearning

In order for NGDiff to work as in Theorem 5, the learning rate schedule needs to be carefully selected so that $0 < \eta_t < \frac{2 g_{\text{R}}^\top g_{\text{NGDiff}}}{g_{\text{NGDiff}}^\top \mathbf{H}_{\text{R}} g_{\text{NGDiff}}}$ at each iteration. In Algorithm 1, we adapt GeN (or AutoLR) in (Bu & Xu, 2024) to the unlearning setting and dynamically set the learning rates[3] as the minimizer of (4): to locally optimize L_{R} and to monotonically increase L_{F}, we use

$$\eta_t^* = g_{\text{R}}^\top g_{\text{NGDiff}} / g_{\text{NGDiff}}^\top \mathbf{H}_{\text{R}} g_{\text{NGDiff}}. \tag{6}$$

We devote Appendix B to explain how GeN works and how we have modified GeN for the unlearning problem, such as only forward passing on R but not F in (6). At high level, GeN estimates two scalars – the numerator and denominator of (6) by analyzing the difference of loss values, thus the high-dimensional Hessian matrix $\mathbf{H}_{\text{R}}$ is never instantiated.

Remark 3.2. *There is a computational overhead to use GeN, as it requires additional forward passes to estimate η_t^*. Nevertheless, we only update the learning rate every 10 iterations so that the overhead is averaged out and thus negligible in practice.*

We monitor the gradient norms and the learning rate in Figure 5 when applying Algorithm 1. We observe that the automatic learning rate is indeed effective, picking up from $5e-5$ to around $2e-4$, and that NGDiff tends to assign a smaller learning rate to the forgetting gradient, not perturbing the model too much to maintain the high utility on the retaining set.

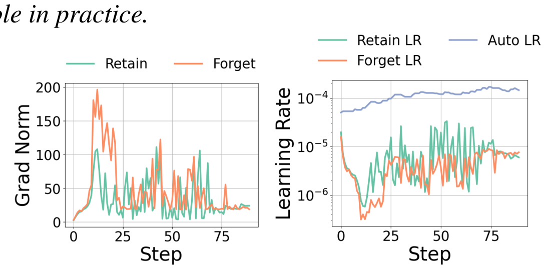

Figure 5: Gradient norms and learning rates during the unlearning on *TOFU* dataset using NGDiff and Phi-1.5 model. The AutoLR scheduler assigns a smaller learning rate to the forgetting gradient, effectively preserving model utility on the retaining set.

[3]We note other parameter-free methods such as D-adaptation, Prodigy, and DoG can also set the learning rate automatically. However, these methods need to be tailored for different gradient methods, hence not compatible to NGDiff or the unlearning algorithms in general. We give a detailed explanation in Appendix D.

Algorithm 1 Learning-rate-free NGDiff

```
for t = 1, 2, ... do
    —-NGDiff—-
    Compute retaining loss L_R(θ_t) by one forward pass on R
    Compute retaining gradient g_R(θ_t) by backward propagation
    Compute forgetting loss L_F(θ_t) by one forward pass on F
    Compute forgetting gradient g_F(θ_t) by backward propagation
    Construct unlearning gradient g_NGDiff = g_R/||g_R|| − g_F/||g_F||
    —-AutoLR—-
    if t mod 10 == 0: then
        Compute L_R^± = L_R(θ_t ± ηg) by two forward passes on R
        Fit the quadratic function in (4) from (−η, 0, η) → (L_R^−, L_R, L_R^+)
        Derive the optimal learning rate η_t^* by (6) and set η = η_t^*
    Update θ_{t+1} = θ_t − η g_NGDiff
```

4 Experiments

4.1 Datasets

We evaluate the empirical performance of our proposed method on the following datasets (see more dataset details in Section E.1):

- *Task of Fictitious Unlearning (TOFU)* (Maini et al., 2024). *TOFU* consists of 20 question-answer pairs based on fictitious author biographies generated by GPT-4 (Achiam et al., 2023). In our experiments, we use the *forget10* (10% of the full training set) as the forgetting set and *retain90* (90% of the full training set) as the retaining set.
- *MUSE-NEWS* (Shi et al., 2024). This dataset consists of BBC news articles (Li et al., 2023b) published since August 2023. We use its *train* split to finetune a target model, and then the *raw* set, which includes both the forgetting and retaining data, for the target model unlearning. Finally, the *verbmem* and *knowmem* splits are used to evaluate the unlearned model's performance.

4.2 Unlearning methods

We compare our method with four baseline methods. The first baseline method is the target model without any unlearning, while the remaining three are the state-of-the-art unlearning methods.

No-unlearn. This method utilizes the full training dataset to fine-tune the base model without any unlearning. The trained model is then used as the target model for subsequent unlearning approaches.

Gradient Difference (GDiff) (Liu et al., 2022). As discussed in Section 2.2, GDiff applies gradient descent on the cross-entropy loss of the retaining data and gradient ascent on the cross-entropy loss of the forgetting data with $c = 0.5$. For a thorough comparison, we also test the extended GDiff method, with $c = 0.1$ (close to GA) or $c = 0.9$ (close to GD).

Loss Normalization (LossNorm). As discussed in Section 2.2, this approach computes and normalizes the forget loss and retain loss separately, with the overall loss being $L_{\text{R}}/|L_{\text{R}}| - L_{\text{F}}/|L_{\text{F}}|$.

Negative Preference Optimization (NPO) (Zhang et al., 2024). GA method often struggles to effectively unlearn the forgetting data, thus resulting in significant degradation in the model's performance on the retaining data. The NPO approach addresses this issue based on preference optimization (Ouyang et al., 2022), and the NPO loss is defined as:

$$L_{\text{NPO},\beta}(\boldsymbol{\theta}) = -\frac{2}{\beta}\mathbb{E}_{\text{F}}\Big[\log\sigma\Big(-\beta\log\frac{f(S,w)}{f_{\text{No-unlearn}}(S,w)}\Big)\Big], \tag{7}$$

where S is randomly sampled from F, $\beta > 0$ is the inverse temperature, f is the unlearned model, and $f_{\text{No-unlearn}}$ is the model before unlearning.

4.3 Foundation Models

We test multiple models: LLaMA2-7B (Touvron et al., 2023), Phi-1.5(Li et al., 2023a), Falcon-1B (Penedo et al., 2023a), GPT2-XL (Radford et al., 2019) and Mistral-7B (Jiang et al., 2023). These models are pre-trained and then fine-tuned on datasets in Section 4.1, with AdamW optimizer and carefully tuned learning rates as described in Appendix E.

4.4 Evaluation Metrics

Following the existing work (Shi et al., 2024), we evaluate the unlearning performance based on model's output quality. We expect a good performance should satisfy the following requirements:

No verbatim memorization. After the unlearning, the model should no longer remember any verbatim copies of the texts in the forgetting data. To evaluate this, we prompt the model with the first k tokens in F and compare the model's continuation outputs with the ground truth continuations. We use ROUGE-L recall scores for this comparison, where a lower score indicates more effective unlearning.

No knowledge memorization. After the unlearning, the model should not only forget verbatim texts, but also the knowledge in the forgetting set. For the *MUSE-NEWS* dataset, we evaluate knowledge memorization using the *Knowmem*$_{\mathrm{F}}$ split, which consists of generated question-answer pairs based on the forgetting data. Similar to verbatim memorization, we use ROUGE-L recall scores for the evaluation.

Maintained model utility. An effective unlearning method must maintain the model's performance on the retaining set. We prompt the model with the question from R and compare the generated answer to the ground truth. We use ROUGE-L recall scores for these comparisons. Additionally, we evaluate the model using the Truth Ratio metric. We use the *Retain10-perturbed* split from the *TOFU* dataset, which consists of five perturbed answers created by modifying the facts in each original answer from R. The Truth Ratio metric computes how likely the model generates a correct answer versus an incorrect one, where a higher value indicates better model utility.

4.5 Main Results

The results for *Verbatim memorization (Verbmem)*, *Model utility (Utility)*, *TruthRatio*, and *Knowledge memorization (Knowmem)* using different unlearning methods are presented in Table 1, 2 and 6. We evaluate these metrics using the *TOFU* and *MUSE-NEWS* datasets across large language models. In summary, our NGDiff consistently achieves the best results across all models on both datasets, highlighting its superior performance. In sharp contrast, the baseline unlearning methods (1) either effectively forget R by reducing *Verbmem* and *Knowmem* but fail maintain the *Utility* and *TruthRatio*, such as GDiff with $c \leq 0.5$, NPO; (2) or cannot unlearn F on Phi-1.5 and Mistral-7B, such as LossNorm and GDiff with $c = 0.9$. We highlight that the effectiveness of these unlearning methods are highly model-dependent and dataset-dependent, unlike NGDiff.

For the *TOFU* dataset, we observe that some unlearning methods fail to unlearn the forget data effectively. For example, GDiff-0.9 and LossNorm do not unlearn effectively when applied to Phi-1.5, Llama2-7B and Mistral-7B. In fact, GDiff-0.9 has $80\% \sim 100\%$ *Verbmem* and LossNorm has $> 40\%$ *Verbmem* on Phi-1.5. However, they are effective on Falcon-1B and GPT2-XL, even though these models have similar sizes ($\approx 1B$ parameters) to Phi-1.5. On the other hand, some methods fail to preserve the model utility after unlearning. For example, GDiff-0.1 has close to 0 *Utility* on Phi-1.5, Falcon-1B, GPT2-XL and Llama2-7B; similarly, NPO also experiences a significant drop in *Utility* on Phi-1.5 model, Falcon-1B and GPT2-XL, but not so on Llama2-7B. In contrast, our NGDiff remains effective in unlearning F and maintaining R across the models. In addition, NGDiff achieves the best *TruthRatio* on all models except Llama2-7B (which is still on par with the best), indicating that the model's answers remain factually accurate for questions in the retaining data.

For the *MUSE-NEWS* dataset, our NGDiff also outperforms the baseline methods on Llama2-7B and Mistral-7B models by achieving a lower *Verbmem* and a higher *Utility*. Furthermore, the *Knowmem* results indicate that NGDiff not only unlearns the verbatim copies of the forgetting texts, but also successfully removes the associated knowledge. The model capacities of Phi-1.5 and Falcon-1B are smaller, limiting their ability to learn knowledge effectively after fine-tuning on the full dataset, as shown in Table 6. Despite this, our method still successfully unlearns the forgetting data while preserving model utility.

Table 1: *Verbatim memorization*, *Model utility*, and *TruthRatio* on *TOFU* dataset (*forget10/retain90)* with different unlearning methods and different models. Lower *Verbmem* along with higher *Utility* and *TruthRatio* indicate a more superior unlearning strategy.

Base Model	Metric	Method						
		No-unlearn	*GDiff-0.9*	*GDiff-0.5*	*GDiff-0.1*	*NPO*	*LossNorm*	*NGDiff*
Phi-1.5	Verbmem ↓	1.000	0.805	0.027	0.000	0.000	0.432	**0.024**
	Utility ↑	1.000	0.992	0.308	0.000	0.000	0.752	**0.747**
	TruthRatio ↑	0.385	0.205	0.216	0.221	0.179	0.214	**0.353**
Falcon-1B	Verbmem ↓	1.000	0.041	0.001	0.000	0.017	0.055	**0.021**
	Utility ↑	1.000	0.434	0.305	0.000	0.114	0.521	**0.428**
	TruthRatio ↑	0.408	0.237	0.244	0.217	0.184	0.252	**0.354**
GPT2-XL	Verbmem ↓	1.000	0.029	0.001	0.000	0.031	0.022	**0.046**
	Utility ↑	0.999	0.381	0.250	0.000	0.136	0.376	**0.792**
	TruthRatio ↑	0.412	0.186	0.278	0.133	0.179	0.196	**0.399**
Llama2-7B	Verbmem ↓	1.000	0.810	0.011	0.000	0.709	0.010	**0.002**
	Utility ↑	1.000	0.851	0.324	0.000	0.682	0.264	**0.724**
	TruthRatio ↑	0.490	0.340	0.364	0.161	0.329	0.329	**0.334**
Mistral-7B	Verbmem ↓	1.000	1.000	0.945	0.410	0.385	0.259	**0.009**
	Utility ↑	1.000	0.999	0.944	0.517	0.341	0.925	**0.996**
	TruthRatio ↑	0.344	0.345	0.366	0.374	0.364	0.358	**0.379**

Table 2: *Verbatim memorization*, *Model utility*, and *Knowledge memorization* on the *MUSE-NEWS* dataset with different unlearning methods on Llama2-7B and Mistral-7B models. Lower *Verbmem* and *Knowmem* along with higher *Utility* indicate a more superior unlearning strategy.

Base Model	Metric	Method						
		No-unlearn	*GDiff-0.9*	*GDiff-0.5*	*GDiff-0.1*	*NPO*	*LossNorm*	*NGDiff*
Llama2-7B	Verbmem ↓	0.561	0.555	0.043	0.004	0.000	0.388	**0.036**
	Utility ↑	0.646	0.641	0.275	0.000	0.000	0.506	**0.556**
	Knowmem ↓	0.755	0.717	0.287	0.000	0.000	0.514	**0.455**
Mistral-7B	Verbmem ↓	0.578	0.177	0.000	0.000	0.113	0.196	**0.098**
	Utility ↑	0.411	0.339	0.000	0.000	0.316	0.343	**0.354**
	Knowmem ↓	0.416	0.257	0.000	0.000	0.343	0.293	**0.165**

To further illustrate the performance of our proposed method during the training, in addition to the last iterate results, we plot the ROUGE scores and loss terms during the unlearning process in Figure 6. We apply the extended GDiff, LossNorm, and NGDiff methods, to the Phi-1.5 model using the *TOFU* dataset. While GDiff with $c = 0.5$ and $c = 0.7$, and NGDiff are effective in unlearning, only NGDiff preserve the model utility above 75% ROUGE score. A closer look at the second and the fourth plots of Figure 6 shows that NGDiff exhibits the fastest and most stable convergence on F while maintaining a low retaining loss ≤ 0.1.

4.6 Ablation Study

Effectiveness of NGDiff. In our experiments, we utilize the automatic learning rate scheduler (AutoLR) for NGDiff method. To investigate the impact of NGDiff alone, we compare all methods with or without AutoLR in Table 3. With AutoLR or not (where we use manually tuned learning rates), NGDiff, GDiff ($c = 0.1$ or 0.5) and NPO can effectively unlearn in terms of *Verbmem*. However, among these four methods, NGDiff uniquely retains a reasonable *Utility* between $60 \sim 75\%$, while other methods retains only $0 \sim 30\%$ *Utility*. A similar pattern is observed in terms of *TruthRatio* as well. Overall, NGDiff significantly outperforms other baseline methods with or without AutoLR.

Impact of automatic learning rate. To evaluate the impact of AutoLR scheduler, we see in Table 3 all methods exhibit an increase in the *TruthRatio* metric and a decrease in *Verbmem*, though with

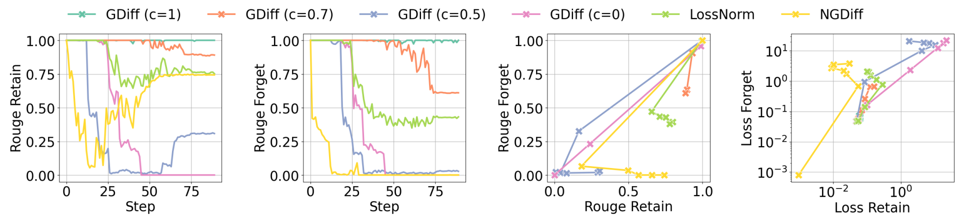

Figure 6: Comparison of different unlearning methods on TOFU dataset. The figures show the ROUGE scores and loss terms during unlearning process with different methods, which includes GDiff, LossNorm, and NGDiff. We observe that NGDiff effectively unlearns the forgetting data while maintaining the performance on the retaining data.

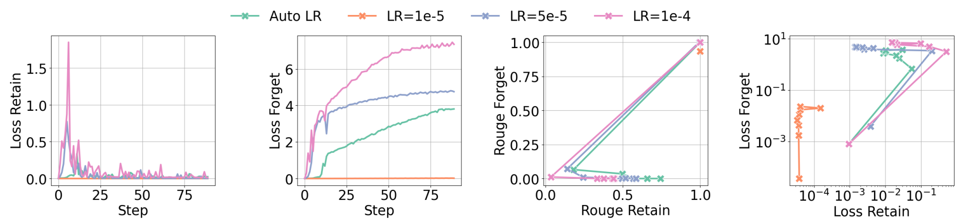

Figure 7: Comparison between AutoLR and different learning rates on NGDiff. The figures show the ROUGE scores and loss values during the unlearning process on TOFU dataset using Phi-1.5 model. We observe that AutoLR outperforms the static learning rates with better model utility and more stable convergence.

some loss in the *Utility*. For instance, *LossNorm* benefits significantly from AutoLR with $\approx 20\%$ decrease in *Verbmem*, and NGDiff increases its retaining *Utility* and *TruthRatio* by $> 22\%$. We specifically demonstrate the impact of AutoLR on NGDiff in Figure 7. Without AutoLR, the model's performance is highly sensitive to the static learning rates: when $\eta = 10^{-5}$, the model fails to unlearn F as indicated by the low loss and high ROUGE score; in contrast, when $\eta = 10^{-4}$, there is a significant drop in ROUGE score on the retain data, falling from 100% to around 50%. However, with the AutoLR scheduler, we observe a steady reduction in the *Verbmem* (with the ROUGE forget close to 0 at convergence) while maintaining high utility (the ROUGE retain is 0.747, which is 19.5% higher than the best results without AutoLR).

Table 3: Results of *Verbmem*, *Utility*, and *TruthRatio* using different unlearning methods with AutoLR on the Phi-1.5 model. AutoLR improves the *TruthRatio* and reduces *Verbmem* across all methods. With or without AutoLR, NGDiff can significantly outperform other baseline methods.

Method	**TOFU** (wo $\rightarrow$ w/ AutoLR)		
	Verbmem $\downarrow$	*Utility* $\uparrow$	*TruthRatio* $\uparrow$
No-unlearn	1.000	1.000	0.385
GDiff c=0.9	0.805 $\rightarrow$ 0.200	0.992 $\rightarrow$ 0.422	0.205 $\rightarrow$ 0.308
GDiff c=0.5	0.027 $\rightarrow$ 0.001	0.308 $\rightarrow$ 0.032	0.216 $\rightarrow$ 0.297
GDiff c=0.1	0.000 $\rightarrow$ 0.000	0.000 $\rightarrow$ 0.000	0.221 $\rightarrow$ 0.229
NPO	0.000 $\rightarrow$ 0.000	0.000 $\rightarrow$ 0.000	0.179 $\rightarrow$ 0.223
LossNorm	0.432 $\rightarrow$ 0.231	0.752 $\rightarrow$ 0.725	0.214 $\rightarrow$ 0.336
NGDiff	0.024 $\rightarrow$ 0.012	0.607 $\rightarrow$ 0.747	0.289 $\rightarrow$ 0.353

5 Discussion

We have formulated the machine unlearning problem as a two-task optimization problem, which can be provably (under Pareto optimality) and effectively solved by our novel unlearning method *NGDiff*. We also adapt GeN to set an automatic and adaptive learning rate scheduler but we believe other learning-rate-free methods can serve as alternatives, maybe after some modifications. While NGDiff is applicable to general problems, most experiments in this work focus on LLMs except one in Appendix C. It would be desirable to test NGDiff on more modalities, and test more MTO methods for machine unlearning (see a list of methods in Appendix D).

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Zhiqi Bu and Shiyun Xu. Automatic gradient descent with generalized newton's method. *arXiv preprint arXiv:2407.02772*, 2024.

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

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2022.

Kongyang Chen, Zixin Wang, Bing Mi, Waixi Liu, Shaowei Wang, Xiaojun Ren, and Jiaxing Shen. Machine unlearning in large language models. *arXiv preprint arXiv:2404.16841*, 2024.

Shijie Chen, Yu Zhang, and Qiang Yang. Multi-task learning in natural language processing: An overview. *ACM Computing Surveys*, 2021.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pp. 794–803. PMLR, 2018a.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks, 2018b. URL `https://arxiv.org/abs/1711.02257`.

Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout, 2020. URL `https://arxiv.org/abs/2010.06808`.

Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by d-adaptation, 2023. URL `https://arxiv.org/abs/2301.07733`.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

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

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models, 2024. URL `https://arxiv.org/abs/2402.00838`.

Varun Gupta, Christopher Jung, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Chris Waites. Adaptive machine unlearning. *Advances in Neural Information Processing Systems*, 34:16319–16330, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Maor Ivgi, Oliver Hinder, and Yair Carmon. Dog is sgd's best friend: A parameter-free dynamic step size schedule, 2023. URL https://arxiv.org/abs/2302.12022.

Adrián Javaloy and Isabel Valera. Rotograd: Gradient homogenization in multitask learning, 2022. URL https://arxiv.org/abs/2103.02631.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Ahmed Khaled, Konstantin Mishchenko, and Chi Jin. Dowg unleashed: An efficient universal parameter-free gradient descent method, 2024. URL https://arxiv.org/abs/2305.16284.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023a.

Yucheng Li, Frank Geurin, and Chenghua Lin. Avoiding data contamination in language model evaluation: Dynamic test construction with latest materials. *arXiv preprint arXiv:2312.12343*, 2023b.

Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*.

Bo Liu, Qiang Liu, and Peter Stone. Continual learning and private unlearning. In *Conference on Lifelong Learning Agents*, pp. 243–254. PMLR, 2022.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning, 2024a. URL https://arxiv.org/abs/2110.14048.

Junxu Liu, Mingsheng Xue, Jian Lou, Xiaoyu Zhang, Li Xiong, and Zhan Qin. Muter: Machine unlearning on adversarially trained models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4892–4902, 2023.

Liyang Liu, Yi Li, Zhanghui Kuang, J Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. iclr, 2021.

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

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C Lipton, and J Zico Kolter. Tofu: A task of fictitious unlearning for llms. *arXiv preprint arXiv:2401.06121*, 2024.

Konstantin Mishchenko and Aaron Defazio. Prodigy: An expeditiously adaptive parameter-free learner, 2024. URL https://arxiv.org/abs/2306.06101.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only, 2023a. URL https://arxiv.org/abs/2306.01116.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023b. URL https://arxiv.org/abs/2306.01116.

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

Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah A Smith, and Chiyuan Zhang. Muse: Machine unlearning six-way evaluation for language models. *arXiv preprint arXiv:2407.06460*, 2024.

The New York Times. Exhibit j, 2023. URL https://nytco-assets.nytimes.com/2023/12/Lawsuit-Document-dkt-1-68-Ex-J.pdf.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Enayat Ullah, Tung Mai, Anup Rao, Ryan A Rossi, and Raman Arora. Machine unlearning via algorithmic stability. In *Conference on Learning Theory*, pp. 4126–4142. PMLR, 2021.

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models, 2020. URL https://arxiv.org/abs/2010.05874.

Derrick Xin, Behrooz Ghorbani, Justin Gilmer, Ankush Garg, and Orhan Firat. Do current multi-task optimization methods in deep learning even help? *Advances in neural information processing systems*, 35:13597–13609, 2022.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. *arXiv preprint arXiv:2404.05868*, 2024.

A COMPARING NGDIFF WITH GRADNORM

Algorithm 2 NGDiff

1: **for** $t = 1, 2, \ldots$ **do**
2: Compute retaining loss $L_{\text{R}}(\boldsymbol{\theta}_t)$ by one forward pass
3: Compute retaining gradient $\boldsymbol{g}_{\text{R}}(\boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}} L_{\text{R}}$
4: Compute forgetting loss $L_{\text{F}}(\boldsymbol{\theta}_t)$ by one forward pass
5: Compute forgetting gradient $\boldsymbol{g}_{\text{F}}(\boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}} L_{\text{F}}$
6: Construct unlearning gradient $\boldsymbol{g}_{\text{NGDiff}} = \boldsymbol{g}_{\text{R}}/||\boldsymbol{g}_{\text{R}}|| - \boldsymbol{g}_{\text{F}}/||\boldsymbol{g}_{\text{F}}||$
7: Update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{g}_{\text{NGDiff}}$

Algorithm 3 GradNorm for two-task

1: Initialize the scalaring coefficients $w_{\text{R}}(\boldsymbol{\theta}_0) = 1$ and $w_{\text{F}}(\boldsymbol{\theta}_0) = 1$
2: Pick value for $\alpha > 0$ and pick the weights $\boldsymbol{\theta}_{\text{LS}}$ (the last shared layer of $\boldsymbol{\theta}_t$)
3: **for** $t = 1, 2, \ldots$ **do**
4: Compute retaining loss $L_{\text{R}}(\boldsymbol{\theta}_t)$ by one forward pass
5: Compute retaining gradient $\boldsymbol{g}_{\text{R}}(\boldsymbol{\theta}_{\text{LS}}) = \nabla_{\boldsymbol{\theta}_{\text{LS}}} L_{\text{R}}$
6: Compute forgetting loss $L_{\text{F}}(\boldsymbol{\theta}_t)$ by one forward pass
7: Compute forgetting gradient $\boldsymbol{g}_{\text{F}}(\boldsymbol{\theta}_{\text{LS}}) = \nabla_{\boldsymbol{\theta}_{\text{LS}}} L_{\text{F}}$
8: Compute loss $L(\boldsymbol{\theta}_t) = w_{\text{R}}(\boldsymbol{\theta}_t) L_{\text{R}}(\boldsymbol{\theta}_t) + w_{\text{F}}(\boldsymbol{\theta}_t) L_{\text{F}}(\boldsymbol{\theta}_t)$
9: Compute $\bar{\boldsymbol{g}}(\boldsymbol{\theta}_{\text{LS}})$ by averaging $\boldsymbol{g}_{\text{R}}$ and $\boldsymbol{g}_{\text{F}}$
10: Compute GradNorm loss

$$L_{GN}(\boldsymbol{\theta}_t) = |\boldsymbol{g}_{\text{R}} - \bar{\boldsymbol{g}} \times [r_{\text{R}}(t)]^{\alpha}|_1 + |\boldsymbol{g}_{\text{F}} - \bar{\boldsymbol{g}} \times [r_{\text{F}}(t)]^{\alpha}|_1$$

11: Compute GradNorm gradients $\nabla_{w_{\text{R}}} L_{GN}$ and $\nabla_{w_{\text{F}}} L_{GN} \in \text{R}$
12: Compute the full gradient $\nabla_{\boldsymbol{\theta}_t} L$
13: Update $w_{\text{R}}(\boldsymbol{\theta}_t) \to w_{\text{R}}(\boldsymbol{\theta}_{t+1})$ and $w_{\text{F}}(\boldsymbol{\theta}_t) \to w_{\text{F}}(\boldsymbol{\theta}_{t+1})$ using $\nabla_{w_{\text{R}}} L_{GN}$ and $\nabla_{w_{\text{F}}} L_{GN}$
14: Update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}_t} L$
15: Renormalize $w_{\text{R}}(\boldsymbol{\theta}_{t+1})$ and $w_{\text{F}}(\boldsymbol{\theta}_{t+1})$ so that $w_{\text{R}}(\boldsymbol{\theta}_{t+1}) + w_{\text{F}}(\boldsymbol{\theta}_{t+1}) = 2$

We compare the GradNorm algorithm Chen et al. (2018b) with our proposed method, NGDiff. We highlight some steps of GradNorm in red to indicate the differences than NGDiff:

- NGDiff sets the scalaring coefficient as $1/||\boldsymbol{g}_{\text{R}}||$ and $1/||\boldsymbol{g}_{\text{F}}||$, while GradNorm uses gradient descent to learn these coefficients as w_{R} and w_{F}.
- NGDiff is model-agnostic while GradNorm contains specific designs for multi-task architecture. In unlearning, there are **2 data splits** (i.e. F and R) and each data split defines one task. Hence all model parameters are shared. However, in the original form of GradNorm, there is **1 data split** on which multiple tasks are defined (can be more than 2). Hence the model parameters are partitioned into [shared layers, task 1 specific layers, task 2 specific layers].
- NGDiff computes the full per-task gradients whereas GradNorm only computes the last shared layer's gradients.
- NGDiff requires 2 back-propagation at each iteration but GradNorm requires 3 (2 for per-task gradients, 1 for $\nabla_{\boldsymbol{\theta}} L$), which may translate to more training time for large models.
- GradNorm introduces additional hyperparameters that can be difficult and costly to tune, and may cause instability of training if not properly tuned. These hyperparameters include α and two learning rates to update w_{R} and w_{F} in Line 13 of Appendix A. In contrast, NGDiff is hyperparameter-free when equipped with GeN (AutoLR).
- NGDiff are theoretically supported by Theorem 5, while the choice of hyperparameters and the use of a heuristic $r_i(t)$ by GradNorm may require further justification. Here $r_i(t) = \widetilde{L}_i(\boldsymbol{\theta}_t)/\mathbb{E}_{\text{task}}[\widetilde{L}_i(\boldsymbol{\theta}_t)]$ is the "relative inverse training rate" of task i, where $\widetilde{L}_i(\boldsymbol{\theta}_t) = L_i(\boldsymbol{\theta}_t)/L_i(\boldsymbol{\theta}_0)$, $i \in \{\text{F}, \text{R}\}$.

In summary, NGDiff is remarkably simpler and more well-suited than GradNorm for unlearning, with stable performance and theoretical ground.

B DETAILS RELATED TO GEN

B.1 BRIEF INTRODUCTION OF GEN

GeN (Bu & Xu, 2024) is a method that sets the learning rate for any given gradient $\boldsymbol{d}$ as

$$\eta_{\text{GeN}} = \frac{\mathbf{G}^\top \boldsymbol{d}}{\boldsymbol{d}^\top \mathbf{H} \boldsymbol{d}}$$

where $\mathbf{G}$ is the gradient and $\mathbf{H}$ is the Hessian matrix of some loss L. One only needs to access the scalars $\mathbf{G}^\top \boldsymbol{d}$ and $\boldsymbol{d}^\top \mathbf{H}\boldsymbol{d}$, without computing the high-dimensional $\mathbf{G}$ and $\mathbf{H}$ (or Hessian-vector product). To do so, two additional forward passes are needed: given a constant (say $\xi = 0.001$), we compute $L(\boldsymbol{\theta} + \xi \boldsymbol{d})$ and $L(\boldsymbol{\theta} - \xi \boldsymbol{d})$. Then by curve fitting or finite difference as demonstrated below, we can estimate up to arbitrary precision controlled by ξ:

$$\mathbf{G}^\top \boldsymbol{d} \approx \frac{L(\boldsymbol{\theta} + \xi \boldsymbol{d}) - L(\boldsymbol{\theta} - \xi \boldsymbol{d})}{2\xi}$$

and

$$\boldsymbol{d}^\top \mathbf{H} \boldsymbol{d} \approx \frac{L(\boldsymbol{\theta} + \xi \boldsymbol{d}) - 2L(\boldsymbol{\theta}) + L(\boldsymbol{\theta} - \xi \boldsymbol{d})}{\xi^2}$$

Notice that the regular optimization requires 1 forward pass and 1 back-propagation; GeN requires in total 3 forward passes and 1 back-propagation. Given that back-propagation costs roughly twice the computation time than forward pass, the total time increases from 3 units of time to 5 units. Nevertheless, GeN needs not to be applied at each iteration: if we update the learning rate every 10 iterations as in Remark 3.2, the total time reduces to $3 + 2/10 = 3.2$ units, and the overhead is less than 10% compared to the regular optimization.

B.2 ADAPTING GEN TO UNLEARNING

Naively applying GeN to the unlearning will result in

$$\eta_{\text{GeN}} = \frac{\mathbf{G}^\top \boldsymbol{g}_{\text{UN}}}{\boldsymbol{g}_{\text{UN}}^\top \mathbf{H} \boldsymbol{g}_{\text{UN}}}$$

which minimizes the loss over all datapoints, in both F and R. This is against our goal to maximize the forgetting loss. We must consider the learning rate separately for F and R, as shown in Appendix F (Proof of Theorem 5). When both losses have a convex curvature in Figure 4, the optimal learning rate is only well-defined for L_{R} and we do not claim to maximize L_{F}. In other words, if we minimize L_{R}, we get to worsen L_{F} (though not maximally); if we choose to maximize L_{F}, we will use infinite learning rate that also maximizes L_{R}. Therefore, our learning rate in (6) only uses R instead of the whole dataset.

C COMPUTER VISION EXPERIMENTS

Table 4: Results of *Forget Acc* and *Retain Acc* using different unlearning methods on the CIFAR-10 dataset. Compared to other baseline methods, *NGDiff* has the best performance on the model utility.

Method	**CIFAR-10**		**CIFAR-100**	
	Forget Acc ↓	*Retain Acc* ↑	*Forget Acc* ↓	*Retain Acc* ↑
No-unlearn	0.926	0.956	0.745	0.750
GDiff c=0.9	0.000	0.817	0.000	0.664
GDiff c=0.5	0.000	0.830	0.000	0.609
GDiff c=0.1	0.000	0.825	0.000	0.667
LossNorm	0.000	0.753	0.000	0.432
NGDiff	0.000	**0.931**	0.000	**0.701**

To demonstrate the effectiveness of unlearning across other modalities, we also evaluate our method on the image classification task. Specifically, we choose the CIFAR-10 and CIFAR-100 dataset

Krizhevsky et al. (2009) and train a ResNet-50 He et al. (2015) model from scratch. For the CIFAR-10 dataset, we sample 500 images from the class dog as the forgetting data, and use images from the remaining 9 classes as the retaining data. For the CIFAR-100 dataset, we sample 500 images from the class bed as the forgetting data, and use images from the remaining 99 classes as the retaining data. After training, the initial forget data accuracy is 0.926, and the retain data accuracy is 0.956 on the CIFAR-10 dataset. The initial forget data accuracy is 0.745, and the retain data accuracy is 0.750 on the CIFAR-100 dataset. Then we apply different unlearning methods to the trained models. As shown in Table 4, all methods successfully reduce the forget accuracy to 0. However, the retaining accuracy of *NGDiff* remains the highest, which shows its effectiveness in preserving the model utility in image classification tasks.

D RELATED WORKS

Machine unlearning Machine unlearning is oftentimes viewed as a continual learning approach, that removes specific data points after a model has been trained to memorize them. Such removal is light-weighted in contrast to re-training, especially when the forgetting set is much smaller than the retaining. In addition to the methods already introduced in Section 2.2 (namely GA, GDiff and NPO), other methods include SISA Bourtoule et al. (2021), influence functions Ullah et al. (2021), differential privacy Gupta et al. (2021) and so on. However, these methods could be difficult to scale on large models and large datasets due to the algorithmic complexity. To our best knowledge, this is the first work that formulate the unlearning problem as a two-task problem, which can be solved by a number of well-known MTO methods.

Multi-task optimization MTO is a paradigm where one model is trained to perform multiple tasks simultaneously, so as to significantly improve the efficiency in contrast to training multiple models, one for each task. The key challenge of MTO is the performance trade-off among tasks, where the multi-task model is worse than single-task model if trained on each task separately. Therefore, the core idea is to balance different tasks by modifying the per-task gradients, e.g. with normalization (LossNorm and NGDiff), PCGrad Yu et al. (2020), RLW Lin et al., IMTL Liu et al. (2021), MGDA Désidéri (2012), CAGrad Liu et al. (2024a), GradVaccine Wang et al. (2020), GradDrop Chen et al. (2020), RotoGrad Javaloy & Valera (2022), etc.

Learning-rate-free methods Parameter-free or learning-rate-free methods automatically set the learning rate scheduler without the hyperparameter tuning, which is computationally infeasible for LLMs, e.g. LLAMA2 pre-training uses 3 hyperparameters just for the learning rate: warmup steps, peak learning rate, and minimum learning rate. At high level, there are two approaches to learning-rate-free methods.

On one hand, GeN Bu & Xu (2024) leverages the Taylor expansion and convex-like landscape of deep learning, which is applicable for the general purpose, even if the gradient is modified like in the unlearning.

On the other hand, methods like D-adaptation Defazio & Mishchenko (2023), Prodigy Mishchenko & Defazio (2024), DoG Ivgi et al. (2023), DoWG Khaled et al. (2024) are based on the convex and G-Lipschitz conditions: $L(\bar{\theta}_T) - L(\boldsymbol{\theta}_*) \leq \frac{|\theta_0 - \theta_*|^2}{2\eta T} + \frac{\eta G^2}{2}$ where θ_* is the unknown minimizer of L and $\bar{\theta}_T$ is an averaging scheme of $\{\theta_0, ..., \theta_T\}$. With the same theoretical foundation, these methods propose different ways to approximate the initial-to-final distance $|\theta_0 - \theta_*|$. There are two main issues to apply these methods on the unlearning. Firstly, the assumption of G-Lipschitz is hard to verify and the minimizer θ_* is not well-defined in multi-objective (see our discussion on Pareto optimality under Lemma 2). Secondly, the optimal learning rate $\frac{|\theta_0 - \theta_*|}{G\sqrt{T}}$ is defined in a manner to minimize the loss, whereas MTO methods operate on the gradient level. Hence MTO is incompatible to such parameter-free methods given that we cannot derive a corresponding loss (e.g. there exists no L_{NGDiff} such that $\frac{\partial L_{\text{NGDiff}}}{\partial \theta} = \boldsymbol{g}_{\text{NGDiff}}$).

Table 5: Statistics of the *TOFU* and *MUSE-NEWS* datasets. For the *TOFU* dataset, we use *Full* split for training the target model, *Forget10* and *Retain90* as the forgetting and retaining split for unlearning experiments. For the *MUSE-NEWS* dataset, we utilize *Train* split for training, *Raw* split for unlearning. For evaluation, we use Verbmem_F and Knowmem_F splits from forgetting data, and Knowmem_R split from the retaining data.

Dataset	TOFU			MUSE-NEWS				
	Full	Forget10	Retain90	Train	Raw	Verbmem_F	Knowmem_F	Knowmem_R
# samples	$4,000$	400	$3,600$	$7,110$	$2,669$	100	100	100

E Experiments

E.1 Datasets

To evaluate the empirical performance of our proposed method, we experiment on the following datasets in Table 5.

- *Task of Fictitious Unlearning (TOFU)* Maini et al. (2024). This dataset consists of question-answer pairs based on fictitious author biographies generated by GPT-4 Achiam et al. (2023). Initially, predefined attributes, such as birthplace, gender, and writing genre, are assigned to 200 distinct authors. GPT-4 is then prompted to generate detailed information about each author. Following the synthesized data, 20 question-answer pairs are created for each fictitious author. The dataset is then divided into distinct datasets: the retaining set and the forgetting set. In our experiments, we use the *forget10* and *retain90* split, which excludes 10% of the original dataset.
- *MUSE-NEWS* Shi et al. (2024). This dataset consists of BBC news articles Li et al. (2023b) from August 2023. It includes seven subsets of news data: *raw*, *verbmem*, *knowmem*, *privleak*, *scal*, *sust*, and *train*. We utilize the *train* split to finetune a target model, and then the *raw* set, which includes both the forget and retain data, for the target model unlearning. Then, we use *verbmem*, *knowmem* split to evaluate the unlearned model's performance.

E.2 Evaluation Metrics

Following the existing work Shi et al. (2024), we evaluate the unlearning performance based on the quality of outputs from the model after unlearning. We expect a good performance should satisfy the following requirements:

No verbatim memorization We evaluate this metric by prompting the model with the first l tokens of the news data in the forget set and compare the model's continuation outputs with the ground truth continuation. Specifically, for each input $x \in F$, we choose $x_{[:l]}$ as input, and compare the output $f(x_{[:l]})$ with the ground truth continuation $x_{[l+1:]}$ with the ROUGE-L recall score:

$$\textit{Verbmem}(f, F) = \frac{1}{||F||} \Sigma_x \text{ ROUGE-L}(f(x_{[:l]}), x_{[l+1:]}) \tag{8}$$

No knowledge memorization To evaluate this metric, we use the generated question-answer pair based on each example $x \in F$. We prompt the model with the question part q and compare the output answer $f(q)$ to the ground truth answer a using ROUGE-L recall scores:

$$\textit{Knowmem}(f, F) = \frac{1}{||F||} \Sigma_x \text{ ROUGE-L}(f(q), a) \tag{9}$$

Maintained model utility An effective unlearning method should also maintain the model's performance on the retain data. For the *MUSE-NEWS* dataset, we use the $\textit{Knowmem}_r$ split, which consists of the generated question-answer pairs based on the retain data. For the *TOFU* dataset, we prompt the model with the question from the retain set and compare the generated answer with the ground truth. We use ROUGE-L recall scores for evaluation:

$$Utility(f, R) = \frac{1}{||R||} \Sigma_x \text{ ROUGE-L}(f(q), a) \tag{10}$$

Additionally, we evaluate the model using the *Retain10-perturbed* split from the *TOFU* dataset. It consists of five perturbed answers for each original answer, keeping original template but modifying the facts. We compute the Truth Ratio metric, which compares the likelihood of the model generating a correct answer versus an incorrect one for each question in the retain set. A higher Truth Ratio indicates better model utility that effectively remembers knowledge from the retain data.

E.3 Hyper-parameter Settings

To finetune a targeted model with the full dataset, we use the optimizer Adam with a learning rate of $\eta = \{10^{-5}, 2 * 10^{-5}\}$, a training batch size of $\{16, 32\}$, and train 25 epochs for all language models. For the unlearning process, we use the optimizer Adam with a learning rate $\eta = \{10^{-5}, 5 * 10^{-5}, 10^{-4}\}$, and train 15 epochs for all unlearning methods.

E.4 Other unlearning results

Table 6: Results of *Verbatim memorization*, *Model utility*, and *TruthRatio* on *MUSE-NEWS* dataset with different unlearning methods on Phi-1.5, and Falcon-1B models. Lower *Verbmem* along with higher *Utility* and *TruthRatio* indicate a more superior unlearning strategy.

Base Model	**Metric**	**Method**						
		No-unlearn	*GDiff-0.9*	*GDiff-0.5*	*GDiff-0.1*	*NPO*	*LossNorm*	*NGDiff*
Phi-1.5	Verbmem ↓	0.018	0.000	0.012	0.000	0.000	0.012	**0.004**
	Utility ↑	0.372	0.277	0.061	0.000	0.000	0.061	**0.001**
	Knowmem ↓	0.030	0.000	0.002	0.000	0.000	0.002	**0.023**
Falcon-1B	Verbmem ↓	0.204	0.132	0.000	0.000	0.000	0.126	**0.000**
	Utility ↑	0.386	0.214	0.000	0.000	0.000	0.142	**0.025**
	Knowmem ↓	0.232	0.078	0.000	0.000	0.000	0.130	**0.087**

F Proofs

Proof of Lemma 4. We firstly show $\boldsymbol{g}_{\text{R}}^\top \boldsymbol{g}_{\text{UN}} \geq 0$ for $\boldsymbol{g}_{\text{UN}} = \boldsymbol{g}_{\text{NGDiff}}$. We write

$$\boldsymbol{g}_{\text{R}}^\top \boldsymbol{g}_{\text{NGDiff}} = \boldsymbol{g}_{\text{R}}^\top \left(\frac{\boldsymbol{g}_{\text{R}}}{||\boldsymbol{g}_{\text{R}}||} - \frac{\boldsymbol{g}_{\text{F}}}{||\boldsymbol{g}_{\text{F}}||} \right) = ||\boldsymbol{g}_{\text{R}}|| - \frac{\boldsymbol{g}_{\text{R}}^\top \boldsymbol{g}_{\text{F}}}{||\boldsymbol{g}_{\text{F}}||} \geq ||\boldsymbol{g}_{\text{R}}|| - \frac{||\boldsymbol{g}_{\text{R}}|| ||\boldsymbol{g}_{\text{F}}||}{||\boldsymbol{g}_{\text{F}}||} = 0$$

where the inequality is the Cauchy-Schwarz inequality. Similarly, $\boldsymbol{g}_{\text{F}}^\top \boldsymbol{g}_{\text{UN}} \leq 0$ easily follows. □

Proof of Theorem 5. Applying (4) with $\boldsymbol{g}_{\text{NGDiff}}$ gives

$$L_{\text{R}}(\boldsymbol{\theta}_{t+1}) - L_{\text{R}}(\boldsymbol{\theta}_t) = -\eta \boldsymbol{g}_{\text{R}}^\top \boldsymbol{g}_{\text{NGDiff}} + \frac{\eta^2}{2} \boldsymbol{g}_{\text{NGDiff}}^\top \mathbf{H}_{\text{R}} \boldsymbol{g}_{\text{NGDiff}} + o(\eta^2) \tag{11}$$

For part (1), note that Lemma 4 gives $\boldsymbol{g}_{\text{R}}^\top \boldsymbol{g}_{\text{NGDiff}} > 0$ unless $\boldsymbol{g}_{\text{F}} \parallel \boldsymbol{g}_{\text{R}}$. Hence for any $\eta > 0$, we have

$$L_{\text{R}}(\boldsymbol{\theta}_{t+1}) - L_{\text{R}}(\boldsymbol{\theta}_t) = -\eta \boldsymbol{g}_{\text{R}}^\top \boldsymbol{g}_{\text{NGDiff}} + o(\eta) < o(\eta)$$

and similarly for L_{F}.

For part (2), now that $\boldsymbol{g}_{\text{NGDiff}}^\top \mathbf{H}_{\text{R}} \boldsymbol{g}_{\text{NGDiff}} > 0$, we have

$$-\eta \boldsymbol{g}_{\text{R}}^\top \boldsymbol{g}_{\text{NGDiff}} + \frac{\eta^2}{2} \boldsymbol{g}_{\text{NGDiff}}^\top \mathbf{H}_{\text{R}} \boldsymbol{g}_{\text{NGDiff}} < 0 \iff 0 < \eta < \frac{2 \boldsymbol{g}_{\text{R}}^\top \boldsymbol{g}_{\text{NGDiff}}}{\boldsymbol{g}_{\text{NGDiff}}^\top \mathbf{H}_{\text{R}} \boldsymbol{g}_{\text{NGDiff}}}$$

and similarly

$$-\eta \boldsymbol{g}_{\text{F}}^\top \boldsymbol{g}_{\text{NGDiff}} + \frac{\eta^2}{2} \boldsymbol{g}_{\text{NGDiff}}^\top \mathbf{H}_{\text{F}} \boldsymbol{g}_{\text{NGDiff}} > 0 \Longleftarrow 0 < \eta$$

We complete the proof by substituting the inequalities into (11). □