Towards Certified Unlearning for Deep Neural Networks

Binchi Zhang [1] **Yushun Dong** [1] **Tianhao Wang** [1] **Jundong Li** [1]

Abstract

In the field of machine unlearning, certified unlearning has been extensively studied in convex machine learning models due to its high efficiency and strong theoretical guarantees. However, its application to deep neural networks (DNNs), known for their highly nonconvex nature, still poses challenges. To bridge the gap between certified unlearning and DNNs, we propose several simple techniques to extend certified unlearning methods to nonconvex objectives. To reduce the time complexity, we develop an efficient computation method by inverse Hessian approximation without compromising certification guarantees. In addition, we extend our discussion of certification to nonconvergence training and sequential unlearning, considering that real-world users can send unlearning requests at different time points. Extensive experiments on three real-world datasets demonstrate the efficacy of our method and the advantages of certified unlearning in DNNs.

1. Introduction

In modern machine learning applications, model training often requires access to vast amounts of user data. For example, computer vision models often rely on images sourced from platforms such as Flickr, shared by its users (Thomee et al., 2016). However, as concerns about user data privacy grow, recent legislation such as the General Data Protection Regulation (GDPR, 2016) and the California Consumer Privacy Act (CCPA, 2018) have underscored the importance of *the right to be forgotten*, which enables users to request the deletion of their data from entities that store it. The emergence of *the right to be forgotten* has prompted the development of a new research domain known as machine unlearning (Sommer et al., 2022; Mercuri et al., 2022; Nguyen et al., 2022; Xu et al., 2023; Zhang et al., 2023), i.e., deleting user data and its lineage from a machine learning model. Although the target data can be effectively unlearned by retraining the model from scratch, this method can incur a high computational cost, limiting its practical applications. To address this problem, certified unlearning was proposed to find an efficient approximation of the retrained model. In particular, the difference between the distributions of the unlearned model and the retrained model is bounded, making certified unlearning a stringent unlearning notion.

Despite the extensive study of certified unlearning in various machine learning models such as linear models (Guo et al., 2020; Izzo et al., 2021; Mahadevan & Mathioudakis, 2021), general convex models (Ullah et al., 2021; Sekhari et al., 2021; Neel et al., 2021), Bayesian models (Nguyen et al., 2020), sum-product networks (Becker & Liebig, 2022), and graph neural networks (Pan et al., 2023; Chien et al., 2023; Wu et al., 2023b;a), the understanding of certified unlearning for *deep neural networks* (DNNs) remains nascent. Most of the previous unlearning methods applied to DNNs (Golatkar et al., 2020; 2021; Tarun et al., 2023; Chundawat et al., 2023) only introduced empirical approximation of the retrained model and failed to obtain a theoretical certification. Although Mehta et al. tentatively analyzed the certification for DNNs, their results strongly rely on the convexity assumption, which is unrealistic and yields a significant gap between certified unlearning and DNNs.

To bridge the gap between certified unlearning and DNNs, we investigate the problem of certified unlearning without relying on convexity assumptions. First, we divide certified unlearning into two steps, estimating the retrained model with a bounded error and adding random noises corresponding to the approximation error bound. Existing works (Golatkar et al., 2020; 2021; Mehta et al., 2022; Warnecke et al., 2023) typically estimate the retrained model based on a single-step Newton update for convex objectives. In this paper, we demonstrate that by making simple modifications to the Newton update, we can establish an approximation error bound for nonconvex (more precisely, weakly convex (Davis & Grimmer, 2019; Davis & Drusvyatskiy, 2019)) functions such as DNNs. It is worth noting that our proposed techniques can be adapted to different approximate unlearning strategies for convex models to improve their soundness in DNNs. To improve the efficiency of the Newton update, we follow previous work to exploit a computationally

[1]University of Virginia, Charlottesville, VA, USA. Correspondence to: Jundong Li <jundong@virginia.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

efficient method to estimate the inverse Hessian in the Newton update and prove that the efficient method preserves a bounded approximation error. In addition, we explore the flexibility of our method to complex real-world scenarios in two specific cases where strategies such as early stopping can stop the training from convergence and users can send unlearning requests at different time points. Consequently, we adapt our certified unlearning approach to the nonconvergence training (the trained model is not a local optimum) and the sequential manner (the current unlearning process stems from a formerly unlearned model) without compromising certification guarantees. Finally, we conduct extensive experiments including ablation studies to verify the effectiveness of our certified unlearning for DNNs in practice. In particular, we use different metrics such as membership inference attacks (Chen et al., 2021) and relearn time (Golatkar et al., 2020) to evaluate the unlearning performance in preserving the privacy of unlearned samples, providing empirical evidence of the effectiveness of our approach. Further analysis explores the advantages of certified unlearning, e.g., stringency (certified unlearning outperforms other unlearning baselines with a relatively loose certification budget), efficiency, and robustness under sequential unlearning.

2. Certified Unlearning

Preliminary. Let $\mathcal{D}$ be a training dataset with n data samples derived from the sample space $\mathcal{Z}$, and let $\mathcal{H}$ be the parameter space of a hypothesis class. Let $\mathcal{A} : \mathcal{Z}^n \to \mathcal{H}$ be a randomized learning process that takes the training set $\mathcal{D}$ as input and outputs the optimal $\boldsymbol{w}^* \in \mathcal{H}$ that minimizes the empirical risk on $\mathcal{D}$ as

$$\boldsymbol{w}^* = \mathcal{A}(\mathcal{D}) = \operatorname{argmin}_{\boldsymbol{w}\in\mathcal{H}} \mathcal{L}(\boldsymbol{w}, \mathcal{D}). \tag{1}$$

$\mathcal{L}$ is the empirical risk function that measures the error of a model over each training sample as $\mathcal{L}(\boldsymbol{w}, \mathcal{D}) = \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{D}} l(\boldsymbol{w}, \boldsymbol{x})$, where $l(\boldsymbol{w}, \boldsymbol{x})$ is the loss function of the model $\boldsymbol{w}$ on the sample $\boldsymbol{x}$, e.g., cross-entropy loss and mean squared error. After the learning process, we obtain a trained model $\boldsymbol{w}^* = \mathcal{A}(\mathcal{D})$.

In machine unlearning, to remove some training samples $\mathcal{D}_u \subset \mathcal{D}$ from the trained model $\boldsymbol{w}^*$, a common way is to leverage a randomized unlearning process $\mathcal{U}$ to update $\boldsymbol{w}^*$ and obtain an unlearned model $\boldsymbol{w}^- = \mathcal{U}(\boldsymbol{w}^*, \mathcal{D}_u, \mathcal{D})$, consequently. We denote the number of unlearned samples as $n_u = |\mathcal{D}_u|$. A straightforward but exact unlearning process is to retrain the model from scratch. Let $\tilde{\boldsymbol{w}}^*$ be the retrained model and $\mathcal{D}_r = \mathcal{D}\backslash\mathcal{D}_u$, then we have $\tilde{\boldsymbol{w}}^* = \operatorname{argmin}_{\boldsymbol{w}\in\mathcal{H}} \mathcal{L}(\boldsymbol{w}, \mathcal{D}_r)$. Despite the rigor of retraining, the high computational cost makes it a trivial method that cannot be used in practice. Instead, certified unlearning processes find an unlearned model similar to the re-trained model, but with much less computation cost. Next, we introduce the definition of certified unlearning (Guo et al., 2020).

Definition 2.1. ($\varepsilon - \delta$ certified unlearning) Let $\mathcal{D}$ be a training set, $\mathcal{D}_u \subset \mathcal{D}$ be an unlearned set, $\mathcal{D}_r = \mathcal{D}\backslash\mathcal{D}_u$ be the retained set, $\mathcal{H}$ be the hypothesis space, and $\mathcal{A}$ be a learning process. $\mathcal{U}$ is an $\varepsilon - \delta$ certified unlearning process iff $\forall\, \mathcal{T} \subseteq \mathcal{H}$, we have

$$\begin{aligned} \Pr\left(\mathcal{U}\left(\mathcal{D}, \mathcal{D}_u, \mathcal{A}\left(\mathcal{D}\right)\right) \in \mathcal{T}\right) &\leq e^{\varepsilon} \Pr\left(\mathcal{A}\left(\mathcal{D}_r\right) \in \mathcal{T}\right) + \delta, \\ \Pr\left(\mathcal{A}\left(\mathcal{D}_r\right) \in \mathcal{T}\right) &\leq e^{\varepsilon} \Pr\left(\mathcal{U}\left(\mathcal{D}, \mathcal{D}_u, \mathcal{A}\left(\mathcal{D}\right)\right) \in \mathcal{T}\right) + \delta. \end{aligned} \tag{2}$$

Certified Unlearning and Differential Privacy. It is worth noting that Definition 2.1 is derived from the notion of differential privacy (Dwork, 2006). Specifically, a randomized algorithm $\mathcal{M} : \mathbb{N}^n \to \mathcal{R}$ gives $\varepsilon - \delta$ differential privacy iff $\forall\, \mathcal{T} \subseteq \mathcal{R}$ and $\forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{N}^n$ such that $\|\boldsymbol{x} - \boldsymbol{y}\|_1 = b$,

$$\Pr(\mathcal{M}(\boldsymbol{x}) \in \mathcal{T}) \leq e^{\varepsilon} \Pr(\mathcal{M}(\boldsymbol{y}) \in \mathcal{T}) + \delta. \tag{3}$$

Here, $\boldsymbol{x}$ and $\boldsymbol{y}$ can be seen as two adjacent datasets with b different data records. Noting the similarity of certified unlearning and differential privacy, we clarify their relationship as follows.

Proposition 2.2. *If learning algorithm $\mathcal{A}$ provides $\varepsilon - \delta$ differential privacy, $\mathcal{A}(\mathcal{D})$ is an $\varepsilon - \delta$ certified unlearned model.*

Proposition 2.2 indicates that differential privacy is a sufficient condition of certified unlearning. One of the advantages of differential privacy is its weak assumptions: differential privacy can be achieved without any requirements on convexity or continuity. Such property perfectly fits DNNs, which are usually nonconvex. Luckily, Proposition 2.2 enables achieving certified unlearning by borrowing the techniques in differential privacy. Consequently, we have the following theorem for certified unlearning.

Theorem 2.3. *Let $\tilde{\boldsymbol{w}}^*$ be the empirical minimizer over $\mathcal{D}_r$ and $\tilde{\boldsymbol{w}} = \mathcal{F}(\boldsymbol{w}^*, \mathcal{D}_u, \mathcal{D})$ be an approximation of $\tilde{\boldsymbol{w}}^*$. Define Δ as an upper bound of $\|\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^*\|_2$, then we have $\mathcal{U}(\boldsymbol{w}^*, \mathcal{D}_u, \mathcal{D}) = \tilde{\boldsymbol{w}} + Y$ is an $\varepsilon - \delta$ certified unlearning process, where $Y \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$ and $\sigma \geq \frac{\Delta}{\varepsilon}\sqrt{2\ln(1.25/\delta)}$.*

The proof of Theorem 2.3 is provided in Appendix A. With Theorem 2.3, we divide the problem of certified unlearning into two parts, *estimating the retrained model* and *adding noise corresponding to the approximation error*. We illustrate the framework of certified unlearning in Figure 1. Next, our primary goal is to find the approximation function $\mathcal{F}$ where the output $\tilde{\boldsymbol{w}} = \mathcal{F}(\boldsymbol{w}^*, \mathcal{D}_u, \mathcal{D})$ estimates the minimizer $\tilde{\boldsymbol{w}}^*$ with a *bounded approximation error*.

Additional Remarks on Privacy and Unlearning. The deduction of our certification budgets stems from results in differential privacy (Dwork et al., 2014). According to recent literature (Balle & Wang, 2018) in differential privacy, we

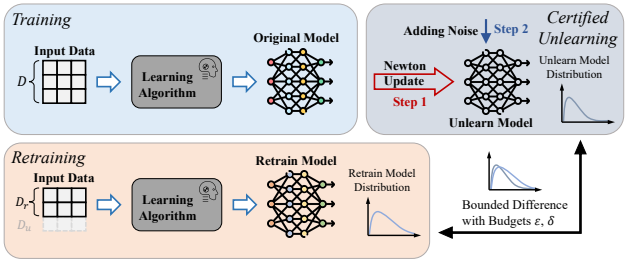


Figure 1. Illustration of certified unlearning, where the first step is to estimate the retrained model based on the original model, and the second step is to add noise to it. According to Theorem 2.3, we can guarantee the difference in distributions between the unlearned model and the retrained model is bounded by certification budgets.

notice that our certification budgets can be further tightened, i.e., we can achieve $\varepsilon - \delta$ certified unlearning by adding a noise with a smaller variance σ. In particular, we can find a smaller σ by satisfying $\Phi(\frac{\Delta}{2\sigma} - \frac{\varepsilon\sigma}{\Delta}) - e^{\varepsilon}\Phi(-\frac{\Delta}{2\sigma} - \frac{\varepsilon\sigma}{\Delta}) \leq \delta$, where Φ is the cumulative distribution function of the standard Gaussian. For simplicity, we still use the results in Theorem 2.3 in the following discussion.

In addition, in the classical differential privacy (Dwork et al., 2014), the value of privacy level ε is less than 1. However, this condition is sometimes difficult to satisfy in certified unlearning for DNNs. Luckily, according to the Theorem 4 in (Balle & Wang, 2018), the certification of unlearning can be preserved when $\varepsilon \geq 1$ by increasing the variance of the noise σ (scaled by a factor $O(\sqrt{\varepsilon})$).

3. Methodology

Existing Studies. In existing studies (Guo et al., 2020; Golatkar et al., 2020; 2021; Mehta et al., 2022), a single-step Newton update is widely used to estimate the empirical minimizer $\tilde{\boldsymbol{w}}^*$ and then add noise to satisfy unlearning. Assuming the second-order derivative of $\mathcal{L}$ with respect to $\boldsymbol{w}$ exists and is continuous, one can take the Taylor expansion of $\nabla\mathcal{L}$ at $\boldsymbol{w}^*$ as

$$\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r) \approx \nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) + \boldsymbol{H}_{\boldsymbol{w}^*}(\tilde{\boldsymbol{w}}^* - \boldsymbol{w}^*), \quad (4)$$

where $\boldsymbol{H}_{\boldsymbol{w}^*} = \nabla^2\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r)$ denotes the Hessian of $\mathcal{L}$ *over* $\mathcal{D}_r$ at $\boldsymbol{w}^*$. We also have $\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r) = \mathbf{0}$ since $\tilde{\boldsymbol{w}}^*$ is the minimizer. Move $\tilde{\boldsymbol{w}}^*$ term to the left-hand side and we have

$$\tilde{\boldsymbol{w}}^* \approx \tilde{\boldsymbol{w}} = \boldsymbol{w}^* - \boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r). \quad (5)$$

In Equation (5), we consider the right-hand side as an approximation of $\tilde{\boldsymbol{w}}^*$ and use $\tilde{\boldsymbol{w}}$ to denote the approximation value. Equation (5) cannot directly serve as the function $\mathcal{F}$ unless it satisfies two requirements simultaneously: **(1). bounded approximation error**; and **(2). computational efficiency**, which are extremely challenging due to the nature of DNNs. However, we found that we can make the single-step Newton update satisfy these requirements by making simple revisions to Equation (5). Next, we provide a detailed analysis.

Approximation Error. In Equation (4), we truncate the Taylor series and only consider the first-order term, hence Equation (5) has an error in estimating the value of $\tilde{\boldsymbol{w}}^*$. Prior to further discussion, we first make the following assumptions.

Assumption 3.1. The loss function $l(\boldsymbol{w}, \boldsymbol{x})$ has an L-Lipschitz gradient in terms of $\boldsymbol{w}$.

Assumption 3.2. The loss function $l(\boldsymbol{w}, \boldsymbol{x})$ has an M-Lipschitz Hessian in terms of $\boldsymbol{w}$.

Assumption 3.1 and Assumption 3.2 are widely used in existing works (Guo et al., 2020; Sekhari et al., 2021; Chien et al., 2023; Wu et al., 2023a;b). In contrast to most existing works, our assumption does not necessitate the convexity of the objective, enabling our discussion to be applicable to DNNs in practice. The following proposition provides a preliminary result on the approximation error bound.

Lemma 3.3. *Let* $\boldsymbol{w}^* = \operatorname{argmin}_{\boldsymbol{w}\in\mathcal{H}}\mathcal{L}(\boldsymbol{w}, \mathcal{D})$ *and* $\tilde{\boldsymbol{w}}^* = \operatorname{argmin}_{\boldsymbol{w}\in\mathcal{H}}\mathcal{L}(\boldsymbol{w}, \mathcal{D}_r)$. *Let* $\tilde{\boldsymbol{w}} = \boldsymbol{w}^* - \boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r)$ *be an approximation of* $\tilde{\boldsymbol{w}}^*$. *Consider Assumption 3.2, we have*

$$\|\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^*\|_2 \leq \frac{M}{2}\|\boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\|_2 \cdot \|\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*\|_2^2. \quad (6)$$

The proof of Lemma 3.3 can be found in Appendix A. From Lemma 3.3, we can find that the approximation error bound is determined by two factors: $\|\boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\|_2$ and $\|\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*\|_2^2$. Next, we aim to bound these two factors separately.

(1). The norm of the inverse Hessian $\|\boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\|_2$: For nonconvex objective $\mathcal{L}(\boldsymbol{w}, \mathcal{D}_r)$, $\|\boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\|_2$ can be arbitrarily large. To bound the value of $\|\boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\|_2$, we exploit the **local convex approximation** technique (Nocedal & Wright, 2006). Specifically, we add an ℓ-2 regularization term to the objective as $\mathcal{L}(\boldsymbol{w}, \mathcal{D}_r) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$ when computing the Hessian. Then, we have $\tilde{\boldsymbol{w}} = \boldsymbol{w}^* - (\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda\boldsymbol{I})^{-1}\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r)$ and the norm of the inverse Hessian changes to $\|(\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda\boldsymbol{I})^{-1}\|_2$, correspondingly. Intuitively, we use local convex approximation to make the nonconvex objective ($\lambda_{min} < 0$) strongly convex with parameter $\lambda + \lambda_{min}$ at $\boldsymbol{w}^*$.

(2). The squared ℓ-2 distance between $\boldsymbol{w}^*$ and $\tilde{\boldsymbol{w}}^*$: $\|\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*\|_2^2$: We find that $\|\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*\|_2^2$ is completely determined by the optimization problem in the learning process. To bound the value of $\|\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*\|_2^2$, we modify the optimization problem by **adding a constraint** $\|\boldsymbol{w}\|_2 \leq C$ **in the learning process**. Then we have $\boldsymbol{w}^* = \operatorname{argmin}_{\|\boldsymbol{w}\|_2\leq C}\mathcal{L}(\boldsymbol{w}, \mathcal{D})$ and $\tilde{\boldsymbol{w}}^* = \operatorname{argmin}_{\|\boldsymbol{w}\|_2\leq C}\mathcal{L}(\boldsymbol{w}, \mathcal{D}_r)$, correspondingly. It is worth noting that previous studies (Guo et al., 2020; Sekhari et al., 2021) assume the objective to be Lipschitz continuous and strongly convex simultaneously. However, these

assumptions are contradictory unless the norm of model parameters is bounded (see Appendix C). In this paper, we directly achieve the important requirements of bounded model parameters which is implicitly required in previous works. We resort to *projected gradient descent* (Bertsekas, 1997) to solve this constrained optimization problem in the learning process. By considering this constraint, we can obtain a worst-case upper bound of the term $\|\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*\|_2^2$ which is *independent of the size of the unlearned set*.

By leveraging the local convex approximation and the constraint on the norm of the parameters, we can further derive a tractable approximation error bound.

Theorem 3.4. *Let $\boldsymbol{w}^* = \operatorname{argmin}_{\|\boldsymbol{w}\|_2 \le C} \mathcal{L}(\boldsymbol{w}, \mathcal{D})$ and $\tilde{\boldsymbol{w}}^* = \operatorname{argmin}_{\|\boldsymbol{w}\|_2 \le C} \mathcal{L}(\boldsymbol{w}, \mathcal{D}_r)$. Denote λ_{min} as the smallest eigenvalue of $\boldsymbol{H}_{\boldsymbol{w}^*}$. Let $\tilde{\boldsymbol{w}} = \boldsymbol{w}^* - (\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I})^{-1} \nabla \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r)$ be an approximation of $\tilde{\boldsymbol{w}}^*$, where $\lambda > \|\boldsymbol{H}_{\boldsymbol{w}^*}\|_2$. Consider Assumption 3.2, we have*

$$\|\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^*\|_2 \le \frac{2C(MC + \lambda)}{\lambda + \lambda_{min}}. \tag{7}$$

The proof of Theorem 3.4 is provided in Appendix A. Theorem 3.4 provides a basic result for the approximation error bound. In contrast to previous studies, our result does not require the objective to be convex. We compare the approximation error bounds with and without the convexity assumption in Appendix B to show our progress.

Computational Efficiency. The main advantage of certified unlearning compared with retraining from scratch is computational efficiency. Noting the importance of efficiency for certified unlearning, we aim to reduce the computational cost of $\tilde{\boldsymbol{w}} = \mathcal{F}(\boldsymbol{w}^*, \mathcal{D}_u, \mathcal{D})$ *while keeping the approximation error bound*. The computation cost contains two parts: the inverse Hessian $(\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I})^{-1}$ (with local convex approximation) and the gradient $\nabla \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r)$. First, assuming the number of learnable parameters is p, the computation of the inverse Hessian requires a complexity of $O(np^2 + p^3)$. Since the scale of the DNNs can be extremely large, we adopt a computationally efficient algorithm, LiSSA (Agarwal et al., 2017), to estimate the inverse Hessian. In particular, we let $X \in \mathcal{D}_r$ be a random variable of the retained training samples. Given s independent and identically distributed (i.i.d.) retained training samples $\{X_1, \ldots, X_s\}$, we can obtain s i.i.d. samples $\{\boldsymbol{H}_{1,\lambda}, \ldots, \boldsymbol{H}_{s,\lambda}\}$ of the Hessian matrix $\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I}$, where $\boldsymbol{H}_{i,\lambda} = \nabla^2 \mathcal{L}(\boldsymbol{w}^*, X_i) + \lambda \boldsymbol{I}$. Then, we can construct an estimator of the inverse Hessian.

Proposition 3.5. *Given s i.i.d. samples $\{\boldsymbol{H}_{1,\lambda}, \ldots, \boldsymbol{H}_{s,\lambda}\}$ of the Hessian matrix $\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I}$, we let*

$$\tilde{\boldsymbol{H}}_{t,\lambda}^{-1} = \boldsymbol{I} + \left(\boldsymbol{I} - \frac{\boldsymbol{H}_{t,\lambda}}{H}\right) \tilde{\boldsymbol{H}}_{t-1,\lambda}^{-1}, \ \tilde{\boldsymbol{H}}_{0,\lambda}^{-1} = \boldsymbol{I}, \tag{8}$$

where $\|\nabla^2 l(\boldsymbol{w}^*, x) + \lambda \boldsymbol{I}\| \le H$, $\forall x \in \mathcal{D}_r$. Then, we have that $\frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}$ is an asymptotic unbiased estimator of the inverse Hessian $(\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I})^{-1}$.

The proof of Proposition 3.5 can be found in Appendix A. By incorporating the Hessian-vector product technique, we can compute $\tilde{\boldsymbol{H}}_{s,\lambda}^{-1} \nabla \mathcal{L}(\boldsymbol{w}^*, \boldsymbol{D}_r)$ with a complexity of $O(sp^2)$ which is independent of n. Second, to compute the gradient $\nabla \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r)$ efficiently, we leverage the property of the minimizer $\boldsymbol{w}^*$ that $\nabla \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}) = \frac{n_u}{n} \nabla \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_u) + \frac{n - n_u}{n} \nabla \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) = 0$. Noting that the number of unlearned samples is usually much smaller than the number of retained samples, we can replace the gradient $\nabla \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r)$ as $\nabla \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) = -\frac{n_u}{n - n_u} \nabla \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_u)$ and reduce the complexity from $O(np)$ to $O(n_u p)$ where $n_u \ll n$. After exploiting the two efficient techniques, we prove the approximation error is still bounded as follows.

Theorem 3.6. *Let $\boldsymbol{w}^* = \operatorname{argmin}_{\|\boldsymbol{w}\|_2 \le C} \mathcal{L}(\boldsymbol{w}, \mathcal{D})$ and $\tilde{\boldsymbol{w}}^* = \operatorname{argmin}_{\|\boldsymbol{w}\|_2 \le C} \mathcal{L}(\boldsymbol{w}, \mathcal{D}_r)$. Let λ_{min} be the smallest eigenvalue of $\boldsymbol{H}_{\boldsymbol{w}^*}$, and s be the recursion number for the inverse Hessian approximation. Let $\tilde{\boldsymbol{w}} = \boldsymbol{w}^* + \frac{n_u}{(n - n_u)H} \tilde{\boldsymbol{H}}_{s,\lambda}^{-1} \nabla \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_u)$ be an approximation of $\tilde{\boldsymbol{w}}^*$, where $\lambda > \|\boldsymbol{H}_{\boldsymbol{w}^*}\|_2$. Consider Assumption 3.1 and Assumption 3.2, the following inequality holds with a probability larger than $1 - \rho$.*

$$\|\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^*\|_2 \le \frac{2C(MC + \lambda)}{\lambda + \lambda_{min}} + \left(\frac{32\sqrt{\ln d/\rho}}{\lambda + \lambda_{min}} + \frac{1}{8}\right) LC. \tag{9}$$

The proof of Theorem 3.6 is provided in Appendix A. Finally, we obtain a computationally efficient $\mathcal{F}$ with bounded approximation error as

$$\mathcal{F}(\boldsymbol{w}^*, \mathcal{D}_u, \mathcal{D}) = \boldsymbol{w}^* + \frac{n_u}{(n - n_u)H} \tilde{\boldsymbol{H}}_{s,\lambda}^{-1} \nabla \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_u). \tag{10}$$

Based on our discussion, the complexity of computing $\mathcal{F}$ is $O(sp^2 + n_u p)$. Consequently, we can achieve certified unlearning based on $\mathcal{F}$ according to Theorem 2.3. We formulate the overall algorithm for certified unlearning in Algorithm 1.

4. Practical Consideration

Nonconvergence. In previous results, we let $\boldsymbol{w}^*$ and $\tilde{\boldsymbol{w}}^*$ be the exact minimum of $\mathcal{L}$ over $\mathcal{D}$ and $\mathcal{D}_r$, respectively. However, the training process of DNNs usually stops before reaching the exact minimum, e.g., early stopping, for better generalization. Hereby, we analyze the approximation error bound under the nonconvergence condition. In the nonconvergence case, $\boldsymbol{w}^*$ and $\tilde{\boldsymbol{w}}^*$ are not the exact minimum which is intractable. We further assume that the norm of the gradient $\|\nabla \mathcal{L}(\boldsymbol{w}^*, \mathcal{D})\|$ and $\|\nabla \mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\|$ are bounded by G. Consequently, we have the adjusted approximation error bound in nonconvergence cases.

Algorithm 1 Single-Batch Certified Unlearning for DNNs

Input: original trained model $\boldsymbol{w}^*$; certification budget ε and δ; local convex coefficient λ; norm upper bound C; Hessian norm bound H; number of recursion s; unlearned set $\mathcal{D}_u$.
Output: certified unlearning model $\boldsymbol{w}^-$.

Compute s i.i.d. Hessian samples $\{\boldsymbol{H}_{1,\lambda}, \ldots, \boldsymbol{H}_{s,\lambda}\}$.
$\boldsymbol{P}_{0,\lambda} \leftarrow \nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_u)$.
for $j = 1, \ldots, s$ **do**
  $\boldsymbol{P}_{j,\lambda} \leftarrow \nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_u) + (\boldsymbol{I} - \frac{\boldsymbol{H}_{j,\lambda}}{H})\boldsymbol{P}_{j-1,\lambda}$.
end for
$\tilde{\boldsymbol{w}} \leftarrow \boldsymbol{w}^* + \frac{n_u}{(n-n_u)H}\boldsymbol{P}_{s,\lambda}$.
Compute the error bound Δ based on Equation (11).
$\sigma \leftarrow \frac{\Delta}{\varepsilon}\sqrt{2\ln(1.25/\delta)}$.
$\boldsymbol{w}^- \leftarrow \tilde{\boldsymbol{w}} + Y$, where $Y \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$.

Proposition 4.1. *Let $\boldsymbol{w}^*$ and $\tilde{\boldsymbol{w}}^*$ be two learned model in practice such that $\|\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D})\|, \|\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\| \leq G$ and $\|\boldsymbol{w}^*\|, \|\tilde{\boldsymbol{w}}^*\| \leq C$. Let λ_{min} be the smallest eigenvalue of $\boldsymbol{H}_{\boldsymbol{w}^*}$, and s be the recursion number for the inverse Hessian approximation. Let $\tilde{\boldsymbol{w}} = \boldsymbol{w}^* + \frac{n_u}{(n-n_u)H}\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_u)$ be an approximation of $\tilde{\boldsymbol{w}}^*$, where $\lambda > \|\boldsymbol{H}_{\boldsymbol{w}^*}\|_2$. Consider Assumption 3.1 and Assumption 3.2, the following inequality holds with a probability larger than $1-\rho$.*

$$\|\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^*\|_2 \leq \frac{2C(MC+\lambda)+G}{\lambda+\lambda_{min}} + \left(\frac{16\sqrt{\ln d/\rho}}{\lambda+\lambda_{min}} + \frac{1}{16}\right)(2LC+G). \tag{11}$$

The proof of Proposition 4.1 is provided in Appendix A. With Proposition 4.1, we can derive a more precise approximation error bound in practice, knowing the tractable residual gradient of the learning algorithm $\mathcal{A}$: $\|\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D})\|$.

Sequential Unlearning. In practical scenarios, the user can send unlearning requests at different time points in a sequential order (Guo et al., 2020; Nguyen et al., 2022). For example, after unlearning one user's data, another user sends the unlearning request. In such a case, we should unlearn the second user's data based on the unlearned model of the first user's data. Consequently, certified unlearning should be able to work in a sequential setting to fit real-world scenarios. We next show that our certified unlearning algorithm can be easily implemented in a sequential manner. Let $\mathcal{D}_{u_k}$ be the unlearned set of the k-th unlearning request and $\mathcal{D}_{r_k} = \mathcal{D} \setminus \cup_{i=1}^{k} \mathcal{D}_{u_i}$ be the retained set after the k-th unlearning request. We estimate the retrained model sequentially as $\tilde{\boldsymbol{w}}_k = \tilde{\boldsymbol{w}}_{k-1} - \frac{1}{H}\tilde{\boldsymbol{H}}_{s,\lambda,k-1}^{-1}\nabla\mathcal{L}(\tilde{\boldsymbol{w}}_{k-1}, \mathcal{D}_{r_k})$ where $\tilde{\boldsymbol{H}}_{s,\lambda,k-1}^{-1}$ is the approximation of the inverse Hessian over $\mathcal{D}_{r_k}$ at $\tilde{\boldsymbol{w}}_{k-1}$. Note that in sequential approximation, we cannot use the efficient computation of $\nabla\mathcal{L}(\tilde{\boldsymbol{w}}_{k-1}, \mathcal{D}_{r_k})$ as $\tilde{\boldsymbol{w}}_{k-1}$ is not a minimum. After estimating the retrained model after the k-th unlearning request, we add noise to obtain the unlearned model according to Theorem 2.3. We can prove that our sequential approximation also has the same approximation error bound as single-step cases.

Algorithm 2 Sequential Unlearning for DNNs

Input: original trained model $\boldsymbol{w}^*$; certification budget ε and δ; local convex coefficient λ; norm upper bound C; number of recursion s; unlearned sets $\{\mathcal{D}_{u_1}, \ldots, \mathcal{D}_{u_k}\}$.
Output: certified unlearning model $\boldsymbol{w}^-$.

$\tilde{\boldsymbol{w}}_0 \leftarrow \boldsymbol{w}^*, \mathcal{D}_{r_0} \leftarrow \mathcal{D}$.
for $i = 1, \ldots, k$ **do**
  $\mathcal{D}_{r_i} \leftarrow \mathcal{D}_{r_{i-1}} \setminus \mathcal{D}_{u_i}$.
  Obtain $\nabla\mathcal{L}(\tilde{\boldsymbol{w}}_{k-1}, \mathcal{D}_{r_k})$ and $\{\boldsymbol{H}_{1,\lambda,i}, \ldots, \boldsymbol{H}_{s,\lambda,i}\}$.
  $\boldsymbol{P}_{0,\lambda,i} \leftarrow \nabla\mathcal{L}(\tilde{\boldsymbol{w}}_{k-1}, \mathcal{D}_{r_k})$.
  for $j = 1, \ldots, s$ **do**
    $\boldsymbol{P}_{j,\lambda,i} \leftarrow \nabla\mathcal{L}(\tilde{\boldsymbol{w}}_{k-1}, \mathcal{D}_{r_k}) + (\boldsymbol{I} - \frac{\boldsymbol{H}_{j,\lambda,i}}{H})\boldsymbol{P}_{j-1,\lambda,i}$.
  end for
  $\tilde{\boldsymbol{w}}_i \leftarrow \tilde{\boldsymbol{w}}_{i-1} - \frac{\boldsymbol{P}_{s,\lambda,i}}{H}$.
end for
Compute the error bound Δ based on Equation (11).
$\sigma \leftarrow \frac{\Delta}{\varepsilon}\sqrt{2\ln(1.25/\delta)}$.
$\boldsymbol{w}^- \leftarrow \tilde{\boldsymbol{w}}_k + Y$, where $Y \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$.

Proposition 4.2. *Let $\tilde{\boldsymbol{w}}_0$ be a model trained on $\mathcal{D}$, $\boldsymbol{w}_k^*$ be a model trained on $\mathcal{D}_{r_k}$, and $\tilde{\boldsymbol{w}}_k = \tilde{\boldsymbol{w}}_{k-1} - \tilde{\boldsymbol{H}}_{s,\lambda,k-1}^{-1}\nabla\mathcal{L}(\tilde{\boldsymbol{w}}_{k-1}, \mathcal{D}_{r_k})$ be an approximation of $\boldsymbol{w}_k^* = \operatorname{argmin}_{\|\boldsymbol{w}\|_2 \leq C}\mathcal{L}(\boldsymbol{w}, \mathcal{D}_{r_k})$ for $k = 1, 2, \ldots$, where λ_{min} is the smallest eigenvalue of $\nabla^2\mathcal{L}(\tilde{\boldsymbol{w}}_{k-1}, \mathcal{D}_{r_k})$ and $\lambda > \|\boldsymbol{H}_{\boldsymbol{w}^*}\|_2$. Consider Assumption 3.1 and Assumption 3.2 and assume $\|\nabla\mathcal{L}(\tilde{\boldsymbol{w}}_k, \mathcal{D}_{r_{k-1}})\|, \|\nabla\mathcal{L}(\boldsymbol{w}_k^*, \mathcal{D}_{r_k})\| \leq G$ for $k = 1, 2, \ldots$, the sequential approximation error $\|\tilde{\boldsymbol{w}}_k - \boldsymbol{w}_k^*\|_2$ has the same upper bound as Proposition 4.1.*

The proof of Proposition 4.2 is provided in Appendix A. Proposition 4.2 provides a theoretical guarantee on the certification for sequential unlearning. As mentioned before, our constraint on model parameters provides a worst-case upper bound of $\|\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*\|_2^2$ independent on $\mathcal{D}_u$ so that our error bound in Equation (11) remains unchanged as the number of unlearned data gradually increases. We illustrate the details of sequential unlearning in Algorithm 2. It is worth noting that we fix the number of unlearned samples in a single unlearning process to be b in Definition 2.1. During the sequential unlearning process, the number of unlearned samples keeps increasing. We can adapt the value of b after each unlearning request to fit the size of the current unlearned set in a post hoc manner. Another rigorous way to preserve the certification while fixing the unlearning granularity b is to incorporate the group privacy notion (Dwork et al., 2014) into the sequential unlearning process. Specifically, we can fix the unlearning granularity as the number of unlearning samples in a single unlearning request by linearly increasing the certification budget ε as $k\varepsilon$, where k is the number of

the unlearning requests.

Reducing the Approximation Error Bound. With a lower approximation error bound, we can achieve the same certification budget by adding a *smaller* noise, leading to a smaller impact on the performance of the unlearned model. Therefore, we summarize our results and list several ways to reduce the approximation error bound in practice.

- **Increasing the value of λ.** According to Equation (11), we can easily obtain that the approximation error bound decreases after increasing the value of λ.
- **Decreasing the value of C.** Similarly, we can reduce the approximation error bound by decreasing the value of C. However, reducing C can impact the effectiveness of the original model. We should carefully select a proper C without compromising the model performance distinctly.
- **Increasing the regularization.** Adding a larger regularization would increase the value of λ_{min}. According to Equation (11), the approximation error bound decreases when λ_{min} increases, which is consistent with the empirical results in (Basu et al., 2021).

5. Experiments

5.1. Dataset Information

We conduct experiments based on three widely adopted real-world datasets for image classification, MNIST (LeCun et al., 1998), SVHN (Netzer et al., 2011), and CIFAR-10 (Krizhevsky et al., 2009) to evaluate certified unlearning for DNNs. The MNIST dataset consists of a collection of 60,000 handwritten digit images for training and 10,000 images for testing. The CIFAR-10 dataset contains 60,000 color images in 10 classes, with each class representing a specific object category. The dataset is split into 50,000 training images and 10,000 test images. The SVHN dataset consists of house numbers images captured from Google Street View. The dataset is divided into three subsets: 73,257 training images, 26,032 test images, and 531,131 extra images. We only use the training set and the test set.

5.2. Baseline Information

To evaluate unlearning methods, we choose three different types of original models for the image classification task. In particular, we train a three-layer MLP model, an All-CNN (Springenberg et al., 2015) model, and a ResNet18 (He et al., 2016) model for MNIST, CIFAR-10, and SVHN, respectively. Moreover, we compare the performance of our certified unlearning method with five unlearning baselines shown as follows. **Retrain from scratch**: retraining from scratch is an ideal unlearning baseline since it is usually considered as an exact unlearning method; **Fine tune** (Golatkar et al., 2020): fine-tuning the original model on the retained set $\mathcal{D}_r$ for one epoch after training the original model; **Negative gradient** (Golatkar et al., 2020): conducting the gradient ascent based on the gradient in terms of the unlearned set $\mathcal{D}_u$ for one epoch after training the original model; **Fisher forgetting** (Golatkar et al., 2020): after training the original model, the Fisher forgetting baseline exploits the Fisher information matrix to substitute the Hessian in the single-step Newton update; **L-CODEC** (Mehta et al., 2022): after training the original model, L-CODEC selects a subset of the model parameters to compute the Hessian based on the Fisher information matrix for efficiency.

5.3. Implementation

We implemented all experiments in the PyTorch (Paszke et al., 2019) package and exploited Adam (Kingma & Ba, 2015) as the optimizer. All experiments are implemented on an Nvidia RTX A6000 GPU. We reported the average value and the standard deviation of the numerical results under three random seeds. For the values of L and M in our theoretical results, since finding a practical upper bound of the Lipschitz constant can be intractable for real-world tasks, we follow most previous works (Koh & Liang, 2017; Wu et al., 2023a;b) to set them as hyperparameters which can be adjusted flexibly to adapt to different scenarios. The choice of these hyperparameters will not affect the soundness of our theoretical results, but render an imprecise value of the certification level, discussed in Appendix D. More detailed hyperparameter settings of certified unlearning and baselines can be found in Appendix D and Section 5.8. Our code is available at `https://github.com/zhangbinchi/certified-deep-unlearning`.

5.4. Unlearning Performance

To evaluate the performance of all unlearning baselines, we select different unlearning metrics according to existing studies. Table 1 exhibits the model utility, i.e., micro F1-score of the predictions over the unlearned set $\mathcal{D}_u$, retained set $\mathcal{D}_r$, and the test set $\mathcal{D}_t$. Based on the meaning of machine unlearning, we expect the results of a desirable unlearning method to be *close to the results of the retrained model*. Hence, we use the retrain-from-scratch baseline as a standard for evaluating the unlearning baselines. Regarding the performance of unlearning baselines, we have the following observations derived from the comparison of our method with unlearning baselines. (1). The micro F1-score of the certified unlearning method on the unlearned set $\mathcal{D}_u$ is closest to the retrained model for most cases. (2). The micro F1-score of the certified unlearning method on the retained set $\mathcal{D}_r$ and the test set $\mathcal{D}_t$ is closest to the retrained model on MNIST and CIFAR-10. (3). Compared with the original training, certified unlearning obtains an unlearned

Table 1. Comparison between the certified unlearning method and unlearning baselines over three popular DNNs across three real-world datasets. We record the micro F1-score of the predictions on the unlearned set $\mathcal{D}_u$, retained set $\mathcal{D}_r$, and test set $\mathcal{D}_t$.

Method	**MLP & MNIST**			**AllCNN & CIFAR-10**			**ResNet18 & SVHN**		
	F1 on $\mathcal{D}_u$	F1 on $\mathcal{D}_r$	F1 on $\mathcal{D}_t$	F1 on $\mathcal{D}_u$	F1 on $\mathcal{D}_r$	F1 on $\mathcal{D}_t$	F1 on $\mathcal{D}_u$	F1 on $\mathcal{D}_r$	F1 on $\mathcal{D}_t$
Original	98.30 ± 0.51	98.37 ± 0.06	97.50 ± 0.08	87.97 ± 3.01	90.71 ± 1.11	83.04 ± 0.20	94.53 ± 0.74	95.00 ± 0.47	93.26 ± 0.34
Retrain	97.20 ± 0.29	98.27 ± 0.09	97.19 ± 0.15	82.67 ± 0.57	90.10 ± 0.98	82.39 ± 0.98	93.13 ± 0.80	95.73 ± 0.60	93.34 ± 0.15
Fine Tune	97.67 ± 0.33	98.35 ± 0.15	97.22 ± 0.09	89.84 ± 1.99	92.25 ± 0.26	84.31 ± 0.54	93.73 ± 0.33	95.23 ± 0.47	93.75 ± 0.41
Neg Grad	97.83 ± 0.59	98.09 ± 0.28	97.22 ± 0.16	79.60 ± 1.92	85.98 ± 2.97	78.66 ± 1.94	92.13 ± 0.77	92.10 ± 0.60	92.10 ± 0.60
Fisher	97.70 ± 0.90	97.56 ± 0.14	96.69 ± 0.05	87.97 ± 3.69	90.71 ± 1.36	83.04 ± 0.24	94.23 ± 0.91	94.84 ± 0.49	93.06 ± 0.29
L-CODEC	98.27 ± 0.61	98.35 ± 0.07	97.46 ± 0.09	88.20 ± 3.70	90.98 ± 1.28	83.33 ± 0.23	95.00 ± 1.06	95.83 ± 0.35	93.53 ± 0.08
Certified	97.60 ± 0.96	98.28 ± 0.05	97.37 ± 0.11	87.83 ± 3.62	90.68 ± 1.32	83.04 ± 0.38	93.73 ± 0.76	94.61 ± 0.57	92.94 ± 0.49

Table 2. Comparison between the certified unlearning method and unlearning baselines over three popular DNNs across three real-world datasets. We record the relearn time, the accuracy of the membership inference attack, and the AUC score of the membership inference attack for measuring the unlearning performance.

Method	**MLP & MNIST**			**AllCNN & CIFAR-10**			**ResNet18 & SVHN**		
	Relearn T	Attack Acc	Attack AUC	Relearn T	Attack Acc	Attack AUC	Relearn T	Attack Acc	Attack AUC
Retrain	25	93.10 ± 0.33	95.16 ± 0.47	17	79.82 ± 0.35	88.71 ± 0.43	7	90.47 ± 0.14	93.07 ± 0.27
Fine Tune	17	93.65 ± 0.23	95.37 ± 0.46	14	79.42 ± 1.05	88.13 ± 0.66	7	90.63 ± 0.32	92.96 ± 0.31
Neg Grad	21	93.73 ± 0.45	95.42 ± 0.43	17	78.63 ± 1.23	87.58 ± 0.96	9	90.02 ± 0.13	92.89 ± 0.22
Fisher	21	93.85 ± 0.22	95.37 ± 0.51	14	79.70 ± 1.03	88.58 ± 0.76	9	90.47 ± 0.84	93.13 ± 0.19
L-CODEC	20	95.05 ± 0.05	95.31 ± 0.21	14	83.60 ± 0.62	92.18 ± 0.17	7	93.22 ± 0.35	93.75 ± 0.54
Certified	24	93.22 ± 0.46	95.28 ± 0.50	25	78.00 ± 1.18	87.22 ± 1.13	9	88.63 ± 1.58	92.18 ± 1.16

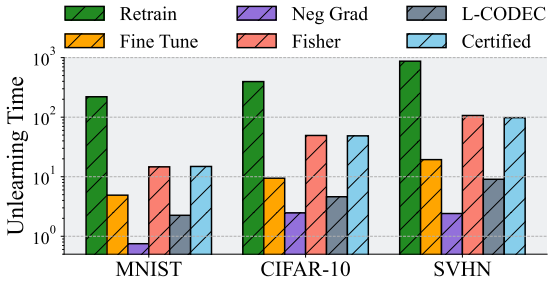


Figure 2. Comparison of unlearning time between the certified unlearning method and unlearning baselines over three popular DNNs across three datasets.

Table 3. The value of approximation error bound, approximation error, and approximation error without constraint under different values of the local convex coefficient λ over the AllCNN backbone on CIFAR-10.

λ	Err Bound	Approx Err	Approx Err (N)
0	35077.7545	26.5146	46.8936
10	390.3298	26.1841	46.7587
10^2	78.4548	26.1724	46.7240
10^3	46.9586	26.1692	46.7091
10^4	43.8059	26.1688	46.7035

model with a lower F1-score over $\mathcal{D}_u$, $\mathcal{D}_r$, and $\mathcal{D}_t$, but the utility over $\mathcal{D}_u$ always decreases the most, consistent with the goal of unlearning. In general, we can find that the negative gradient baseline leads to a large utility drop over all subsets and the L-CODEC baseline yields a distinct utility increase over all subsets. In contrast, our method has a lower prediction utility over the unlearned set $\mathcal{D}_u$ and maintains a desirable utility over the retained set $\mathcal{D}_r$ and the test set $\mathcal{D}_t$.

In Table 2, we record several different unlearning metrics, i.e., relearn time, F1-score of membership inference attack, and AUC score of membership inference attack. By relearning the unlearned model over the unlearned set, relearn time is the number of epochs for the loss function over the unlearned set $\mathcal{D}_u$ to descend to a fixed threshold. We tried using the loss value of the original model over the forget set as the threshold but led to a very small relearn time that cannot discriminate any two baselines (most methods can recover the performance within 1 epoch). Finally, we artificially choose the threshold small enough to make the results of different baselines distinguishable. As an unlearning metric, a lower relearn time indicates the unlearned model retains more information of the unlearned samples. Moreover, we use the membership inference attack to evaluate the information leakage of the unlearning methods. Membership inference attack trains an adversary to infer whether a data sample is used to train the unlearned model. We choose the state-of-the-art membership inference attack method (Chen et al., 2021) for evaluating machine unlearning methods. Specifically, we use the accuracy and the AUC score to measure the utility of the membership inference attack. A higher attack success rate indicates the unlearned model contains more information of the unlearned samples. The experimental results demonstrate that our method is more effective

than other unlearning baselines in removing the information of the unlearned samples. It is worth noting that the retraining baseline is less desirable than other unlearning methods in some cases regarding the membership inference attack results. The reason is that the retraining baseline is directly used to train the shadow unlearned model in the membership inference attack framework (Chen et al., 2021) while the attack model is transferred to attack other unlearning baselines. In conclusion, our method can effectively remove the information of unlearned samples from the unlearned model and preserve the privacy of unlearned samples.

5.5. Efficiency

Efficiency is a crucial advantage of certified unlearning compared with retraining. We record the time cost in the unlearning stage for different unlearning baselines and exhibit the results in Figure 2. From the experimental results, we can observe that (1). Negative gradient has the shortest unlearning time and retraining has the longest unlearning time. (2). Certified unlearning has over 10 times speedup compared with exact unlearning (retrain) over DNNs. (3). Negative gradient, fine-tuning, and L-CODEC have shorter unlearning time but less desirable unlearning performance than certified unlearning. In addition, an important advantage of certified unlearning is its efficiency in adjusting hyperparameters. In practice, we should choose a proper group of hyperparameters considering the tradeoff between the utility of unlearning target data and the utility of predicting remaining data. Although the prediction utility can be evaluated efficiently by prevalent utility metrics such as accuracy and F1-score, evaluating unlearning utility is much more costly. For the unlearning methods without a certification, we can only obtain the unlearning utility by comparing the prediction utility with retrained models or launching membership inference attacks. In contrast, for certified unlearning methods, the certification budgets can be an effective clue of the unlearning utility and we can adjust the hyperparameters efficiently corresponding to the certification budgets.

5.6. Ablation Study

Knowing that certified unlearning for DNNs is practical in real-world experiments, we conduct ablation studies to verify the effectiveness of adopted techniques separately. Specifically, to verify the effectiveness of the local convex approximation, we remove the local convex approximation by setting $\lambda = 0$. We gradually increase the value of λ and record the approximation error and the approximation error bound computed by Equation (11). Moreover, to verify the effectiveness of the constraint $\|\boldsymbol{w}\|_2 \leq C$, we remove the constraint and record the approximation error under each λ. To compute the approximation error, we compute the Euclidean distance of the unlearned model parameters and the retrained model parameters. We present the experimental results in Table 3. We can observe from Table 3 that (1). Local convex approximation can distinctly reduce the approximation error bound and slightly reduce the real approximation error as well. (2). Certified unlearning with the constraint $\|\boldsymbol{w}\|_2 \leq C$ has a much lower approximation error (bound). (3). There exists a potential of the approximation error bound to be further tightened.

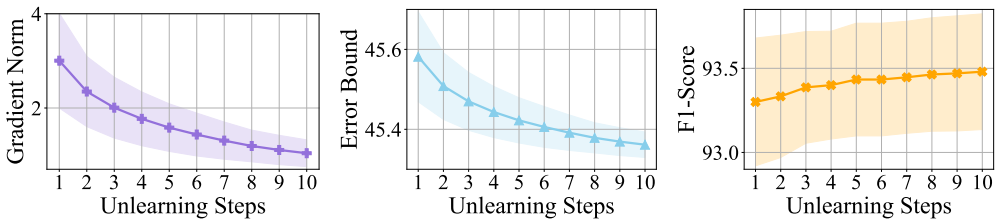
In addition, we also remove the inverse Hessian approximation to verify its effect in reducing the time complexity. In practice, we use *torch.linalg.solve* function to compute the exact inverse matrix of the full Hessian instead of using the inverse Hessian approximation. Due to the memory limitation, we only conduct the experiment with MLP on the MNIST dataset. The results show that the inverse Hessian approximation brings *470 times speedup* compared with computing the exact value of the inverse Hessian.

5.7. Sequential Unlearning

We conduct experiments on certified unlearning in a sequential setting to verify its feasibility in practice. In particular, we sequentially delete 10,000 training samples from a trained ResNet-18 model over the SVHN dataset in 10 iterations. Experimental results are shown in Figure 3. Note that each unlearning step can be seen as a single-step Newton update that reduces the loss value over the corresponding retain set. In the sequential setting, the estimation in each unlearning step reduces the loss value over the corresponding retain set. It is worth noting that the retained set in each step is a subset of the retained set in any previous step. Hence, the loss value over the retained set in the k-th unlearning step D_{r_k} is reduced in each previous unlearning step. Subsequently, the gradient norm $||\nabla\mathcal{L}(\tilde{w}_{k-1}, \mathcal{D}_{r_k})||$ also decreases as the model parameters approach a local optimum, so does the approximation error bound according to Equation (11). As a result, in a sequential setting, we can still launch the certified unlearning method, while robustly maintaining the utility of the unlearned model.

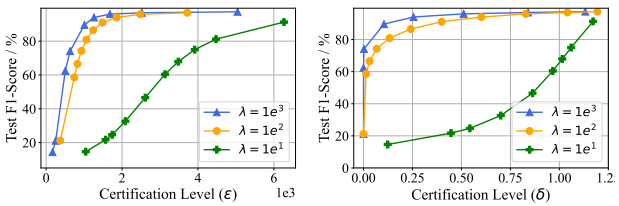
5.8. Parameter Study

In this experiment, we discuss the tradeoff between certification budgets (ε, δ) and the utility of the unlearned model. According to Theorem 2.3, to achieve a tighter certification budget, we have to add a larger noise, which can also lead to utility degradation and limit the practical application of certified unlearning. To demonstrate the effects of the certification budgets ε and δ on the unlearned model utility, we record the micro F1-score of the unlearned model over the test set when adding Gaussian noises with different standard deviations. We repeat the experiment with different values of the local convex coefficient λ and present the results in Figure 4. We fix the budget $\delta = 0.1$ for Figure 4(a)



(a) Grad Norm (b) Error Bound (c) Test F1-Score

Figure 3. Gradient norm, approximation error bound, and model utility after each unlearning step.



(a) Test F1-ε curve (b) Test F1-δ curve

Figure 4. The effect of local convex coefficient λ and certification budget ε and δ over the MLP backbone on MNIST.

and $\varepsilon = 1e^3$ for Figure 4(b). From the experimental results, we can obtain that (1). The unlearned model utility will decrease as the certification budget becomes tighter. (2). Under a fixed certification budget, the model utility increases when increasing the value of the local convex coefficient λ. (3). When increasing the value of the local convex coefficient λ, the model utility becomes more sensitive in terms of the certification level. It is worth noting that the utility increase indicates a smaller noise added in the unlearning stage. A smaller noise under a fixed certification level further indicates a lower approximation error. Therefore, Figure 4 also verifies the effectiveness of local convex approximation in mitigating the approximation error bound. Although the certification budgets in Figure 4 cannot decrease to a very small value when maintaining a desirable utility, our certified unlearning outperforms most unlearning baselines in various unlearning metrics even with a relatively loose budget according to Table 1 and Table 2, which verifies the stringency of certified unlearning. Meanwhile, the potential to decrease the certification budget also necessitates further studies on tightening the approximation error bound under nonconvex neural models.

6. Related Works

6.1. Exact Unlearning

Exact unlearning is a straightforward method for machine unlearning. The idea of exact unlearning is to find efficient ways to retrain the original model from scratch. From the beginning of machine unlearning, exact unlearning has been widely studied for statistical query learning (Cao & Yang, 2015), K-Means (Ginart et al., 2019), random forest (Brophy & Lowd, 2021), deep neural networks (Bourtoule et al., 2021; Aldaghri et al., 2021; Kim & Woo, 2022), and graph neural networks (Chen et al., 2022). Despite having desirable unlearning effectiveness, exact unlearning methods cannot scale to large stochastic models or tackle with batch unlearning settings (Nguyen et al., 2022).

6.2. Certified Unlearning

Certified unlearning was proposed based on the definition of differential privacy. The idea of certified unlearning is to find an approximation model similar to the retrained model in distribution by computationally efficient methods. Existing certified unlearning methods focus primarily on linear models (Guo et al., 2020; Izzo et al., 2021; Mahadevan & Mathioudakis, 2021), general convex models (Ullah et al., 2021; Sekhari et al., 2021; Neel et al., 2021), Bayesian models (Nguyen et al., 2020), sum-product networks (Becker & Liebig, 2022), and graph neural networks (Pan et al., 2023; Chien et al., 2023; Wu et al., 2023a;b). It is worth noting that most aforementioned certified unlearning methods are based on Newton update, i.e., influence function (Koh & Liang, 2017), to estimate the retrained model. Some existing works (Mehta et al., 2022) took a tentative step towards certified unlearning for DNNs, relying on strong assumptions on the convexity of the objective.

7. Conclusion

Despite the extensive study of certified unlearning in many convex machine learning models, the application of certified unlearning in DNNs still poses challenges. In this paper, we proposed two simple techniques to extend certified unlearning to nonconvex objectives and incorporated an inverse Hessian approximation approach to improve efficiency. Regarding the real-world scenarios, we also provide the theoretical analysis of the certification under nonconvergence training and sequential unlearning settings. We conducted extensive empirical experiments to verify the efficacy of our proposed methods and the superiority of certified unlearning in efficiently deleting the information and protecting the privacy of unlearned data.

Acknowledgements

This work is supported in part by the National Science Foundation under grants (IIS-2006844, IIS-2144209, IIS-2223769, CNS-2154962, CNS-2213700, and BCS-2228534), the Commonwealth Cyber Initiative Awards under grants (VV-1Q23-007, HV-2Q23-003, and VV-1Q24-011), the JP Morgan Chase Faculty Research Award, and the Cisco Faculty Research Award.

Impact Statement

This paper extends the certification of machine unlearning to nonconvex, nonconvergence, and sequential settings in real-world scenarios. The goal of this work is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Agarwal, N., Bullins, B., and Hazan, E. Second-order stochastic optimization for machine learning in linear time. *arXiv preprint arXiv:1602.03943*, 2016.

Agarwal, N., Bullins, B., and Hazan, E. Second-order stochastic optimization for machine learning in linear time. *The Journal of Machine Learning Research*, pp. 4148–4187, 2017.

Aldaghri, N., Mahdavifar, H., and Beirami, A. Coded machine unlearning. *IEEE Access*, pp. 88137–88150, 2021.

Balle, B. and Wang, Y.-X. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pp. 394–403, 2018.

Basu, S., Pope, P., and Feizi, S. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*, 2021.

Becker, A. and Liebig, T. Certified data removal in sum-product networks. *arXiv preprint arXiv:2210.01451*, 2022.

Bertsekas, D. P. Nonlinear programming. *Journal of the Operational Research Society*, pp. 334–334, 1997.

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., and Papernot, N. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 141–159, 2021.

Brophy, J. and Lowd, D. Machine unlearning for random forests. In *International Conference on Machine Learning*, pp. 1092–1104, 2021.

Cao, Y. and Yang, J. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pp. 463–480, 2015.

CCPA. https://oag.ca.gov/privacy/ccpa. 2018. URL https://oag.ca.gov/privacy/ccpa.

Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., and Zhang, Y. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 896–911, 2021.

Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., and Zhang, Y. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 499–513, 2022.

Chien, E., Pan, C., and Milenkovic, O. Efficient model updates for approximate unlearning of graph-structured data. In *International Conference on Learning Representations*, 2023.

Chundawat, V. S., Tarun, A. K., Mandal, M., and Kankanhalli, M. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7210–7217, 2023.

Davis, D. and Drusvyatskiy, D. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

Davis, D. and Grimmer, B. Proximally guided stochastic subgradient method for nonsmooth, nonconvex problems. *SIAM Journal on Optimization*, 29(3):1908–1930, 2019.

Dwork, C. Differential privacy. In *Proceedings of the 33rd international conference on Automata, Languages and Programming-Volume Part II*, pp. 1–12, 2006.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, pp. 211–407, 2014.

GDPR. https://gdpr-info.eu/. 2016. URL https://gdpr-info.eu/.

Ginart, A., Guan, M., Valiant, G., and Zou, J. Y. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 2019.

Golatkar, A., Achille, A., and Soatto, S. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9304–9312, 2020.

Golatkar, A., Achille, A., Ravichandran, A., Polito, M., and Soatto, S. Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 792–801, 2021.

Guo, C., Goldstein, T., Hannun, A., and Van Der Maaten, L. Certified data removal from machine learning models. In *International Conference on Machine Learning*, pp. 3832–3842, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Izzo, Z., Smart, M. A., Chaudhuri, K., and Zou, J. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pp. 2008–2016, 2021.

Kim, J. and Woo, S. S. Efficient two-stage model retraining for machine unlearning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4361–4369, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894, 2017.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kurmanji, M., Triantafillou, P., Hayes, J., and Triantafillou, E. Towards unbounded machine unlearning. *Advances in Neural Information Processing Systems*, 36, 2023.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Liu, J., Ram, P., Yao, Y., Liu, G., Liu, Y., SHARMA, P., Liu, S., et al. Model sparsity can simplify machine unlearning. *Advances in Neural Information Processing Systems*, 36, 2023.

Mahadevan, A. and Mathioudakis, M. Certifiable machine unlearning for linear models. *arXiv preprint arXiv:2106.15093*, 2021.

Mehta, R., Pal, S., Singh, V., and Ravi, S. N. Deep unlearning via randomized conditionally independent hessians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10422–10431, 2022.

Mercuri, S., Khraishi, R., Okhrati, R., Batra, D., Hamill, C., Ghasempour, T., and Nowlan, A. An introduction to machine unlearning. *arXiv preprint arXiv:2209.00939*, 2022.

Neel, S., Roth, A., and Sharifi-Malvajerdi, S. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pp. 931–962, 2021.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Nguyen, Q. P., Low, B. K. H., and Jaillet, P. Variational bayesian unlearning. *Advances in Neural Information Processing Systems*, 33:16025–16036, 2020.

Nguyen, T. T., Huynh, T. T., Nguyen, P. L., Liew, A. W.-C., Yin, H., and Nguyen, Q. V. H. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*, 2022.

Nocedal, J. and Wright, S. J. *Numerical Optimization*. Springer, 2006.

Pan, C., Chien, E., and Milenkovic, O. Unlearning graph classifiers with limited data resources. In *Proceedings of the ACM Web Conference 2023*, pp. 716–726, 2023.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Sekhari, A., Acharya, J., Kamath, G., and Suresh, A. T. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, pp. 18075–18086, 2021.

Sommer, D. M., Song, L., Wagh, S., and Mittal, P. Athena: Probabilistic verification of machine unlearning. *Proc. Privacy Enhancing Technol*, 3:268–290, 2022.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations, Workshop Track Proceedings*, 2015.

Tarun, A. K., Chundawat, V. S., Mandal, M., and Kankanhalli, M. Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, pp. 64–73, 2016.

Ullah, E., Mai, T., Rao, A., Rossi, R. A., and Arora, R. Machine unlearning via algorithmic stability. In *Conference on Learning Theory*, pp. 4126–4142, 2021.

Warnecke, A., Pirch, L., Wressnegger, C., and Rieck, K. Machine unlearning of features and labels. In *Network and Distributed System Security Symposium, NDSS*, 2023.

Wu, J., Yang, Y., Qian, Y., Sui, Y., Wang, X., and He, X. Gif: A general graph unlearning strategy via influence function. In *Proceedings of the ACM Web Conference 2023*, pp. 651–661, 2023a.

Wu, K., Shen, J., Ning, Y., Wang, T., and Wang, W. H. Certified edge unlearning for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2606–2617, 2023b.

Xu, H., Zhu*, T., Zhang, L., Zhou, W., and Yu, P. S. Machine unlearning: A survey. *ACM Computing Surveys*, 2023.

Zhang, H., Nakamura, T., Isohara, T., and Sakurai, K. A review on machine unlearning. *SN Computer Science*, 2023.

A. Proofs

A.1. Proof of Proposition 2.2

Proof. Let the unlearning process be an identical map in terms of the model, i.e., $\mathcal{U}(\mathcal{D}, \mathcal{D}_u, \mathcal{A}(\mathcal{D})) = \mathcal{A}(\mathcal{D})$. Since $\mathcal{A}$ is a differentially private algorithm, we have $\forall\ \mathcal{T} \subseteq \mathcal{H}$,

$$\Pr(\mathcal{A}(\mathcal{D}) \in \mathcal{T}) \leq e^{\varepsilon}\Pr(\mathcal{A}(\mathcal{D}_u) \in \mathcal{T}) + \delta. \tag{12}$$

Similarly, we also have $\forall\ \mathcal{T} \subseteq \mathcal{H}$,

$$\Pr(\mathcal{A}(\mathcal{D}_u) \in \mathcal{T}) \leq e^{\varepsilon}\Pr(\mathcal{A}(\mathcal{D}) \in \mathcal{T}) + \delta. \tag{13}$$

According to Definition 2.1, $\mathcal{A}(\mathcal{D})$ is an $\varepsilon - \delta$ certified unlearned model under our defined unlearning process $\mathcal{U}$. □

A.2. Proof of Theorem 2.3

Proof. According to Lemma A.1, we obtain that $\mathcal{U}$ is $\varepsilon - \delta$ differentially private. According to Proposition 2.2, $\mathcal{U}(\boldsymbol{w}^*, \mathcal{D}_u, \mathcal{D}) = \tilde{\boldsymbol{w}} + Y$ is an $\varepsilon - \delta$ certified unlearning process. □

Lemma A.1. *Let f be an arbitrary d-dimensional function, define its ℓ_2 sensitivity as $\Delta_2 f = \max_{\text{adjacent } x, x'} \|f(x) - f(x')\|_2$. For $c^2 > 2\ln(1.25/\delta)$, the Gaussian mechanism, adding noise scaled to $\mathcal{N}(0, \sigma^2)$ to each of the d components of the output, with the parameter $\sigma > \frac{c\Delta_2 f}{\varepsilon}$, is (ε, δ) differentially private.*

Proof. Refer to the proof of Theorem A.1 (Dwork et al., 2014). □

A.3. Proof of Lemma 3.3

Proof. First, we have

$$\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^* = \boldsymbol{w}^* - \boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) - \tilde{\boldsymbol{w}}^*. \tag{14}$$

Considering that $\tilde{\boldsymbol{w}}^* = \operatorname{argmin}_{\boldsymbol{w}\in\mathcal{H}}\mathcal{L}(\boldsymbol{w}, \mathcal{D}_r)$, we have $\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r) = 0$. By substitute this equation into Equation (14) we have

$$\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^* = \boldsymbol{w}^* - \tilde{\boldsymbol{w}}^* - \boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\left(\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) - \nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\right). \tag{15}$$

According to Assumption 3.2, the loss function $\mathcal{L}$ has the second-order derivative. Consequently, according to the fundamental theorem of calculus, we have

$$\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) - \nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r) = \int_0^1 \boldsymbol{H}_{\boldsymbol{w}^* + t(\tilde{\boldsymbol{w}}^* - \boldsymbol{w}^*)}(\tilde{\boldsymbol{w}}^* - \boldsymbol{w}^*)dt. \tag{16}$$

By incorporate Equation (16) into Equation (15), we have

$$\begin{aligned}\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^* &= \boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\left(\boldsymbol{H}_{\boldsymbol{w}^*}(\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*) + \int_0^1 \boldsymbol{H}_{\boldsymbol{w}^* + t(\tilde{\boldsymbol{w}}^* - \boldsymbol{w}^*)}(\tilde{\boldsymbol{w}}^* - \boldsymbol{w}^*)dt\right) \\ &= \boldsymbol{H}_{\boldsymbol{w}^*}^{-1} \cdot \int_0^1 (\boldsymbol{H}_{\boldsymbol{w}^*} - \boldsymbol{H}_{\boldsymbol{w}^* + t(\tilde{\boldsymbol{w}}^* - \boldsymbol{w}^*)})(\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*)dt.\end{aligned} \tag{17}$$

We compute the norm of both sides and obtain the right-hand side as

$$\begin{aligned}&\left\|\boldsymbol{H}_{\boldsymbol{w}^*}^{-1} \cdot \int_0^1 (\boldsymbol{H}_{\boldsymbol{w}^*} - \boldsymbol{H}_{\boldsymbol{w}^* + t(\tilde{\boldsymbol{w}}^* - \boldsymbol{w}^*)})(\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*)dt\right\| \\ \leq &\|\boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\| \cdot \int_0^1 \|\boldsymbol{H}_{\boldsymbol{w}^*} - \boldsymbol{H}_{\boldsymbol{w}^* + t(\tilde{\boldsymbol{w}}^* - \boldsymbol{w}^*)}\| \cdot \|\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*\|dt.\end{aligned} \tag{18}$$

According to Assumption 3.2, the loss function l has an M-Lipschitz Hessian. Hence, the loss function $\mathcal{L}$ also has an M-Lipschitz Hessian. Consequently, we have

$$\|\boldsymbol{H}_{\boldsymbol{w}^*} - \boldsymbol{H}_{\boldsymbol{w}^* + t(\tilde{\boldsymbol{w}}^* - \boldsymbol{w}^*)}\| \leq Mt\|\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*\|. \tag{19}$$

Incorporating Equation (19) into Equation (18) leads to

$$\begin{aligned}&\left\|\boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\cdot\int_0^1(\boldsymbol{H}_{\boldsymbol{w}^*}-\boldsymbol{H}_{\boldsymbol{w}^*+t(\tilde{\boldsymbol{w}}^*-\boldsymbol{w}^*)})(\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*)dt\right\|\\ \leq&\|\boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\|\cdot\int_0^1 Mt\|\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*\|^2dt\\ =&\frac{M}{2}\|\boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\|\cdot\|\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*\|^2.\end{aligned}\tag{20}$$

To finish the proof, we can incorporate Equation (20) into Equation (17) and let the norm be the ℓ-2 norm.

□

A.4. Proof of Theorem 3.4

Proof. Following the proof of Lemma 3.3, we have

$$\tilde{\boldsymbol{w}}-\tilde{\boldsymbol{w}}^*=\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*-(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})^{-1}(\nabla\mathcal{L}(\boldsymbol{w}^*,\mathcal{D}_r)-\nabla\mathcal{L}\left(\tilde{\boldsymbol{w}}^*,\mathcal{D}_r\right)).\tag{21}$$

Based on Assumption 3.2, Equation (16) still holds. Incorporating Equation (16) into Equation (21) leads to

$$\tilde{\boldsymbol{w}}-\tilde{\boldsymbol{w}}^*=(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})^{-1}\cdot\int_0^1(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I}-\boldsymbol{H}_{\boldsymbol{w}^*+t(\tilde{\boldsymbol{w}}^*-\boldsymbol{w}^*)})(\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*)dt.\tag{22}$$

Then, we compute the norm of both sides and obtain the right-hand side as

$$\begin{aligned}&\left\|(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})^{-1}\cdot\int_0^1(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I}-\boldsymbol{H}_{\boldsymbol{w}^*+t(\tilde{\boldsymbol{w}}^*-\boldsymbol{w}^*)})(\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*)dt\right\|\\ \leq&\|(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})^{-1}\|\cdot\int_0^1\left(\lambda+\|\boldsymbol{H}_{\boldsymbol{w}^*}-\boldsymbol{H}_{\boldsymbol{w}^*+t(\tilde{\boldsymbol{w}}^*-\boldsymbol{w}^*)}\|\right)\cdot\|\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*\|dt.\end{aligned}\tag{23}$$

According to Assumption 3.2, Equation (19) still holds. By incorporating Equation (19) into Equation (23), we have

$$\begin{aligned}&\left\|(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})^{-1}\cdot\int_0^1(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I}-\boldsymbol{H}_{\boldsymbol{w}^*+t(\tilde{\boldsymbol{w}}^*-\boldsymbol{w}^*)})(\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*)dt\right\|\\ \leq&\|(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})^{-1}\|\cdot\int_0^1 Mt\|\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*\|^2+\lambda\|\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*\|dt\\ =&\left(\frac{M}{2}\|\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*\|+\lambda\right)\|(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})^{-1}\|\cdot\|\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*\|.\end{aligned}\tag{24}$$

For the inverse Hessian term, We then have

$$\begin{aligned}\|(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})^{-1}\|_2&=\sqrt{\lambda_{max}[((\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})^{-1})^\top(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})^{-1}]}\\ &=\sqrt{\lambda_{max}[((\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})^\top)^{-1}]}\\ &=\frac{1}{\sqrt{\lambda_{min}[(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})(\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I})^\top]}}\\ &=\frac{1}{\lambda_{min}[\boldsymbol{H}_{\boldsymbol{w}^*}+\lambda\boldsymbol{I}]}\\ &=\frac{1}{\lambda+\lambda_{min}[\boldsymbol{H}_{\boldsymbol{w}^*}]},\end{aligned}\tag{25}$$

where $\lambda_{min}[\boldsymbol{H}_{\boldsymbol{w}^*}]$ and $\lambda_{max}[\boldsymbol{H}_{\boldsymbol{w}^*}]$ denote the minimum and maximum eigenvalue of the Hessian $\boldsymbol{H}_{\boldsymbol{w}^*}$. In addition, we also have $\|\boldsymbol{w}^*\|_2\leq C$ and $\|\tilde{\boldsymbol{w}}^*\|_2\leq C$. Consequently, we have

$$\|\boldsymbol{w}^*-\tilde{\boldsymbol{w}}^*\|_2\leq 2C.\tag{26}$$

To finish the proof, we can incorporate Equation (25), Equation (26), and Equation (24) into Equation (22) and let the norm be the ℓ-2 norm.

□

A.5. Proof of Proposition 3.5

Proof. We compute the expectation value for both sides of Equation (8). Considering $\{\boldsymbol{H}_{1,\lambda}, \ldots, \boldsymbol{H}_{s,\lambda}\}$ are i.i.d. samples from the Hessian $\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I}$, we have

$$\begin{aligned}\mathbb{E}[\tilde{\boldsymbol{H}}_{t,\lambda}^{-1}] =& \mathbb{E}\left[\boldsymbol{I} + \left(\boldsymbol{I} - \frac{\boldsymbol{H}_{t,\lambda}}{H}\right)\tilde{\boldsymbol{H}}_{t-1,\lambda}^{-1}\right] \\ =& \boldsymbol{I} + \mathbb{E}[\tilde{\boldsymbol{H}}_{t-1,\lambda}^{-1}] - \frac{1}{H}\mathbb{E}[\boldsymbol{H}_{t,\lambda}\tilde{\boldsymbol{H}}_{t-1,\lambda}^{-1}] \\ =& \boldsymbol{I} + \mathbb{E}[\tilde{\boldsymbol{H}}_{t-1,\lambda}^{-1}] - \frac{1}{H}\mathbb{E}[\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I}]\mathbb{E}[\tilde{\boldsymbol{H}}_{t-1,\lambda}^{-1}].\end{aligned} \tag{27}$$

Let $s \to \infty$, the limit $\mathbb{E}[\tilde{\boldsymbol{H}}_{\infty,\lambda}^{-1}] = \lim_{t\to\infty} \mathbb{E}[\tilde{\boldsymbol{H}}_{t,\lambda}^{-1}]$ exists as $\|\frac{\mathbb{E}[\boldsymbol{H}_{t,\lambda}]}{H}\| \leq 1$. We then compute the limit for both sides of Equation (27) and have

$$\mathbb{E}[\tilde{\boldsymbol{H}}_{\infty,\lambda}^{-1}] = \boldsymbol{I} + \mathbb{E}[\tilde{\boldsymbol{H}}_{\infty,\lambda}^{-1}] - \frac{1}{H}\mathbb{E}[\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I}]\mathbb{E}[\tilde{\boldsymbol{H}}_{\infty,\lambda}^{-1}]. \tag{28}$$

Consequently, we have

$$\mathbb{E}\left[\frac{\tilde{\boldsymbol{H}}_{\infty,\lambda}^{-1}}{H}\right] = \mathbb{E}[(\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I})^{-1}]. \tag{29}$$

Hence, we have proved that $\frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}$ is an asymptotic unbiased estimator of the inverse Hessian $(\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I})^{-1}$.

□

A.6. Proof of Theorem 3.6

Proof. Following the proof of Lemma 3.3, we have

$$\begin{aligned}\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^* &= \boldsymbol{w}^* - \tilde{\boldsymbol{w}}^* + \frac{n_u}{(n-n_u)H}\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_u) \\ &= \boldsymbol{w}^* - \tilde{\boldsymbol{w}}^* - \frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) \\ &= \boldsymbol{w}^* - \tilde{\boldsymbol{w}}^* - \frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}(\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) - \nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)) \\ &= \boldsymbol{w}^* - \tilde{\boldsymbol{w}}^* - \left((\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I})^{-1} + \frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H} - (\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I})^{-1}\right) \cdot (\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) - \nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)).\end{aligned} \tag{30}$$

The right-hand side of Equation (30) can be divided into two parts, (1). $\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^* - (\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I})^{-1}(\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) - \nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r))$ and (2). $((\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I})^{-1} - \tilde{\boldsymbol{H}}_{s,\lambda}^{-1}/H)(\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) - \nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r))$. We compute the norm for both sides of Eq. (30) and the norm of the right-hand side is smaller than the summation of the norm of the part (1) and the norm of the part (2). To find an upper bound of $\|\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^*\|$, we can find upper bounds for the norm of part (1) and the norm of part (2). According to Theorem 3.4, the norm of part (1) is bounded by $\frac{2C(MC+\lambda)}{\lambda+\lambda_{min}}$. Next, our goal is to find the upper bound of the norm of part (2). In particular, the norm of part (2) is smaller than the product $\|(\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I})^{-1} - \tilde{\boldsymbol{H}}_{s,\lambda}^{-1}/H\| \cdot \|\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) - \nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\|$. Refer to Lemma 3.6 (Agarwal et al., 2016), we have

$$\left\|(\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda \boldsymbol{I})^{-1} - \frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}\right\| > \left(\frac{16\sqrt{\ln d/\rho}}{\lambda + \lambda_{min}} + \frac{1}{16}\right), \tag{31}$$

with a probability smaller than ρ. According to Assumption 3.1, we also have

$$\|\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) - \nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\| \leq L\|\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^*\| \leq 2LC. \tag{32}$$

Then, we incorporate Equation (31) and Equation (32) and obtain the upper bound of the norm of part (2): $\left(\frac{32\sqrt{\ln d/\rho}}{\lambda+\lambda_{min}} + \frac{1}{8}\right) LC$. To finish the proof, we can combine the upper bounds for the norm of part (1) and the norm of part (2) and then obtain Equation (9).

□

A.7. Proof of Proposition 4.1

Proof. Following the proof of Lemma 3.3, we have

$$\begin{aligned}\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^* &= \boldsymbol{w}^* - \tilde{\boldsymbol{w}}^* + \frac{n_u}{(n-n_u)H}\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_u) \\ &= \boldsymbol{w}^* - \tilde{\boldsymbol{w}}^* - \frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) \\ &= \boldsymbol{w}^* - \tilde{\boldsymbol{w}}^* - \frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}\left(\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) - \nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\right) - \frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r).\end{aligned} \tag{33}$$

We compute the norm for both sides of Equation (33) and have

$$\|\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^*\| \leq \left\|\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^* - \frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}(\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) - \nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r))\right\| + \left\|\frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\right\|. \tag{34}$$

To find an upper bound of $\|\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^*\|$, we can find an upper bound for each norm value on the right-hand side. According to the proof of Theorem 3.6, we have $\left\|\boldsymbol{w}^* - \tilde{\boldsymbol{w}}^* - \frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}(\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r) - \nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r))\right\|$ is upper bounded by $\frac{2C(MC+\lambda)}{\lambda+\lambda_{min}} + (\frac{32\sqrt{\ln d/\rho}}{\lambda+\lambda_{min}} + \frac{1}{8})LC$ with a probability larger than $1-\rho$. Next, our goal is to find the upper bound for $\left\|\frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\right\|$. In particular, we have

$$\begin{aligned}\left\|\frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\right\| &\leq \|(\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda\boldsymbol{I})^{-1}\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\| + \left\|(\frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H} - (\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda\boldsymbol{I})^{-1})\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\right\| \\ &\leq \|(\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda\boldsymbol{I})^{-1}\| \cdot \|\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\| + \left\|\frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H} - (\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda\boldsymbol{I})^{-1}\right\| \cdot \|\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\|.\end{aligned} \tag{35}$$

According to Equation (25), we have $\|(\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda\boldsymbol{I})^{-1}\| \leq \frac{1}{\lambda - \|\boldsymbol{H}_{\boldsymbol{w}^*}\|}$. According to Equation (31), we have $\|(\boldsymbol{H}_{\boldsymbol{w}^*} + \lambda\boldsymbol{I})^{-1} - \tilde{\boldsymbol{H}}_{s,\lambda}^{-1}/H\| > \left(\frac{16\sqrt{\ln d/\rho}}{\lambda+\lambda_{min}} + \frac{1}{16}\right)$ with a probability smaller than ρ. We also have $\|\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D})\|, \|\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\| \leq G$. By incorporating these results into Equation (35) we have

$$\left\|\frac{\tilde{\boldsymbol{H}}_{s,\lambda}^{-1}}{H}\nabla\mathcal{L}(\tilde{\boldsymbol{w}}^*, \mathcal{D}_r)\right\| \leq \left(\frac{1}{\lambda - \|\boldsymbol{H}_{\boldsymbol{w}^*}\|} + \frac{16\sqrt{\ln d/\rho}}{\lambda+\lambda_{min}} + \frac{1}{16}\right)G. \tag{36}$$

To finish the proof, we can incorporate Equation (36) and results of Theorem 3.6 into Equation (34).

□

A.8. Proof of Proposition 4.2

Proof. Follow the proof of Proposition 4.1. □

B. Comparison Between Convex and Nonconvex Objectives

In this section, we focus on the results shown in Theorem 3.4. Our results proposed in this paper do not require the objective to be convex. However, if we further assume the objective to be convex, we can obtain a tighter approximation upper bound as follows.

Proposition B.1. *Let* $\boldsymbol{w}^* = \operatorname{argmin}_{\|\boldsymbol{w}\|_2 \leq C}\mathcal{L}(\boldsymbol{w}, \mathcal{D})$ *and* $\tilde{\boldsymbol{w}}^* = \operatorname{argmin}_{\|\boldsymbol{w}\|_2 \leq C}\mathcal{L}(\boldsymbol{w}, \mathcal{D}_r)$. *Let* $\tilde{\boldsymbol{w}} = \boldsymbol{w}^* - \boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\nabla\mathcal{L}(\boldsymbol{w}^*, \mathcal{D}_r)$ *be an approximation of* $\tilde{\boldsymbol{w}}^*$. *Consider Assumption 3.2 and assume* $l(\boldsymbol{w}, \boldsymbol{x})$ *to be* K*-strongly convex with respect to* $\boldsymbol{w}$*, then we have*

$$\|\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^*\|_2 \leq \frac{2MC^2}{K}. \tag{37}$$

Table 4. The hyperparameter settings of original models on the corresponding datasets.

Hyperparameter	MLP	AllCNN	ResNet
learning rate	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$
weight decay	$5e^{-4}$	$5e^{-4}$	$5e^{-4}$
epochs	50	50	50
dropout	0.5	0.5	0.5
batch size	128	128	128
param bound C	10	20	20

Proof. As the loss function l is K-strongly convex, the loss function $\mathcal{L}$ is also K-strongly convex. Consequently, we have $\|\boldsymbol{H}_{\boldsymbol{w}^*}\|_2 \geq K$ and $\|\boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\|_2 \leq \frac{1}{K}$. In addition, Equation (26) still holds. By incorporating the upper bound $\|\boldsymbol{H}_{\boldsymbol{w}^*}^{-1}\|_2 \leq \frac{1}{K}$ and Equation (26) into Lemma 3.3, we can obtain Equation (37). □

Considering the similarity between $\lambda + \lambda_{min}$ in Equation (7) and K in Equation (37) (both measuring the objective's convexity), we use $\lambda + \lambda_{min}$ to replace K in Equation (37) and derive the approximation error bound for convex models as $\frac{2MC^2}{\lambda+\lambda_{min}}$. Comparing the result with Equation (37), we find that the Newton update method has a lower approximation error bound for convex models versus nonconvex models. This result highlights the fact that our method works under *both convex and nonconvex objectives*. In particular, our approximation obtains a tighter upper bound for strongly convex objectives.

C. Bounded Model Parameters is also Necessary for Convex Models

In this section, we demonstrate our observation that previous works (Sekhari et al., 2021) on certified unlearning for convex models implicitly rely on the requirement of bounded model parameters $\|\boldsymbol{w}\|_2 \leq C$, which means our proposed constraint on model parameters is also suitable for convex models.

Previous works (Sekhari et al., 2021) studied the certified unlearning and its generalization for convex models. In particular, they jointly assume the objective to be L-Lipschitz continuous and M-strongly convex in the proof of certification. Next, we demonstrate that these two assumptions jointly indicate $\|\boldsymbol{w}\|_2 \leq C$. Let $\boldsymbol{w}_1, \boldsymbol{w}_2$ be two models in the hypothesis space, and $\mathcal{L}$ be the loss function. Based on the strong convexity, we have

$$\|\nabla\mathcal{L}(\boldsymbol{w}_1) - \nabla\mathcal{L}(\boldsymbol{w}_2)\| \geq M\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|. \tag{38}$$

In addition, according to the Lipschitz continuity, we have

$$\|\nabla\mathcal{L}(\boldsymbol{w})\| \leq L, \tag{39}$$

for any $\boldsymbol{w}$ in the hypothesis space. Incorporate Eq. (39) into Eq. (38) and we have

$$2L \geq \|\nabla\mathcal{L}(\boldsymbol{w}_1) - \nabla\mathcal{L}(\boldsymbol{w}_2)\| \geq M\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|. \tag{40}$$

Consequently, we have $\|\boldsymbol{w}_1 - \boldsymbol{w}_2\| \leq \frac{2L}{M}$ for any $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ in the hypothesis space. The certification requires any two models in the hypothesis space to have a bounded distance, which is a non-trivial condition. Luckily, by letting $C = \frac{L}{M}$, our constraint on model parameters $\|\boldsymbol{w}\| \leq C$ satisfy the condition for certification. Hence, we find that the assumptions in previous works naturally necessitate a constraint of bounded norm to the model parameters. In this paper, we exploit projected gradient descent to actively restrict the norm of model parameters for bounding the approximation error. We argue that this technique is also necessary for fulfilling the assumptions made in previous works to derive the certification for convex models.

D. Implementation

We implemented all experiments in the PyTorch (Paszke et al., 2019) package and exploited Adam (Kingma & Ba, 2015) as the optimizer for training. For the training of original models, we exploited Adam as the optimizer. We set the learning rate as $1e^{-3}$, the weight decay parameter as $5e^{-4}$, and the training epochs number as 50. We ran all experiments on an Nvidia RTX A6000 GPU. All experiments are conducted based on three real-world datasets: MNIST (LeCun et al., 1998),

CIFAR-10 (Krizhevsky et al., 2009), and SVHN (Netzer et al., 2011). All datasets are publicly accessible (MNIST with GNU General Public License, CIFAR-10 with MIT License, and SVHN with CC BY-NC License). We reported the average value and the standard deviation of the numerical results under three different random seeds. For the relearn time in Table 2, we directly report the rounded mean value without the standard deviation as the value of the epoch number is supposed to be an integer. The unlearned data is selected randomly from the training set. Detailed hyperparameter settings of the original models are presented in Table 4. The hyperparameter settings of unlearning baselines are shown as follows.

- **Retrain from scratch**: size of unlearned set: 1,000.
- **Fine tune**: size of unlearned set: 1,000; learning rate: $1e^{-3}$; epochs: 1.
- **Negative gradient**: size of unlearned set: 1,000; learning rate: $1e^{-4}$; epochs: 1.
- **Fisher forgetting**: size of unlearned set: 1,000; α: $1e^{-6}$ for MLP, $1e^{-8}$ for AllCNN and ResNet.
- **L-CODEC**: size of unlearned set: 1,000; number of perturbations: 25; Hessian type: Sekhari; ε: 100; δ: 0.1; ℓ-2 regularization: $5e^{-4}$.
- **Certified unlearning**: size of unlearned set: 1,000; number of recursion s: 1,000; standard deviation σ: $1e^{-2}$ for MLP, $1e^{-3}$ for AllCNN and ResNet; continuity coefficients L: 1, M: 1; minimal eigenvalue of Hessian λ_{min}: 0; convex coefficient λ: 1 for MLP, 200 for AllCNN, 2,000 for ResNet; Hessian scale H: 10 for MLP, 20,000 for AllCNN, 50,000 for ResNet.

The detailed meanings of hyperparameters in unlearning baselines can be found in the original papers. For λ, since finding a practical upper bound of the norm of Hessian can be intractable for real-world tasks, we follow most previous works (Koh & Liang, 2017; Wu et al., 2023a;b) to set it as a hyperparameter which can be chosen flexibly to adapt to different scenarios. In addition, we compute the norm of the Hessian in the case of MLP over MNIST. To reduce the time complexity, we use the Hessian with respect to a single random mini-batch of D_r as an unbiased estimation of the full Hessian. The results of the mean value and the variance of the norm of the stochastic Hessian (under 10 different random seeds) is 12.11 ± 0.63, which falls into the range of λ in Table 3. When $\lambda < 12.11$, the computed error bound is still valid (from Table 3, as the value of λ decreases, the computed error bound increases correspondingly, indicating that a smaller choice of λ can still lead to a valid but larger error bound). In this case, the certification requires adding a larger noise to hide the (overestimated) remaining information of the unlearned data. Different from λ, the values of L, M, and λ_{min} only affect the value of the approximation error bound Δ. In practice, following previous works (Guo et al., 2020), we first determine the variance of noise σ (to preserve the model utility after adding noise) and then obtain the certification level ε and δ which can be achieved. Hence, the certification holds for any choices of L, M, and λ_{min}, but the calculated certification level can be imprecise in some cases. To weaken the dependency of the approximation error bound on the Lipschitz constants, we leave more general theoretical results as future works. It is also worth noting that the values of certification budgets ε and δ are flexible in our experiments. In particular, we fixed the noise and the budgets ε and δ can be tuned in a range for specific needs (we have $\varepsilon \in [1e^2, 5e^3]$ when $\delta \in [0.1, 1]$, and the decrease of one can lead to the increase of the other). Detailed results regarding the certification budgets are shown in the parameter study in Section 5.8. In addition, we list some key packages in Python required for implementing certified unlearning as follows.

- python == 3.9.16
- torch == 1.12.1
- torchvision == 0.13.1
- numpy == 1.24.3
- scikit-learn == 1.2.2
- scipy == 1.10.1

Table 5. The micro F1-score of the original model and the retrained model without using PGD.

	MLP & MNIST	CNN & CIFAR-10	ResNet & SVHN
Original	97.03 ± 0.25	83.16 ± 0.62	93.92 ± 0.19
Retrain	96.80 ± 0.20	83.39 ± 0.59	93.80 ± 0.31

E. Additional Experiments

E.1. Effect of Bounded Model Parameters

In our experiments, we adopt PGD during the training of the original model and the retrained model. PGD is conducted solely for ensuring the strictness of our theoretical results while trying not to affect the model utility. To achieve this, we deliberately choose the value of C in the constraint $|w|_2 < C$ to make PGD have little impact on the model utility. After verifying with experiments, we find that $C = 10$ for MLP over MNIST, $C = 20$ for CNN over CIFAR-10, and $C = 20$ for ResNet over SVHN can be a desirable choice. In this experiment, we record and report the utility metrics (micro F1-score over the test set) of the original model and retrained model without conducting PGD in Table 5. Compared with the corresponding results in Table 1, we can verify that our adopted PGD modification with a carefully chosen constraint does not affect the model utility distinctly.

E.2. Comparison with Advanced Baselines

Table 6. Comparison of accuracy among three advanced unlearning baselines over three popular DNNs across three real-world datasets. We record the micro F1-score of the predictions on the unlearned set $\mathcal{D}_u$, retained set $\mathcal{D}_r$, and test set $\mathcal{D}_t$.

Method	**MLP & MNIST**			**AllCNN & CIFAR-10**			**ResNet18 & SVHN**		
	F1 on $\mathcal{D}_u$	F1 on $\mathcal{D}_r$	F1 on $\mathcal{D}_t$	F1 on $\mathcal{D}_u$	F1 on $\mathcal{D}_r$	F1 on $\mathcal{D}_t$	F1 on $\mathcal{D}_u$	F1 on $\mathcal{D}_r$	F1 on $\mathcal{D}_t$
SCRUB	98.10 ± 0.60	98.12 ± 0.05	97.25 ± 0.04	90.73 ± 2.63	96.83 ± 0.82	86.31 ± 0.31	94.07 ± 0.38	96.09 ± 1.25	94.02 ± 0.38
NegGrad+	97.43 ± 0.60	98.92 ± 0.01	97.68 ± 0.07	80.70 ± 8.79	96.58 ± 0.47	86.05 ± 1.52	91.03 ± 1.32	97.58 ± 0.45	95.07 ± 0.12
ℓ-1 Sparse	97.83 ± 0.47	98.40 ± 0.19	97.24 ± 0.12	87.17 ± 2.80	92.26 ± 0.37	83.99 ± 0.32	94.33 ± 0.61	95.55 ± 0.33	93.67 ± 0.16

To strengthen the soundness of certified unlearning, we supply three more advanced baselines for approximate unlearning: SCRUB (Kurmanji et al., 2023), NegGrad+ (Liu et al., 2023), and ℓ-1 Sparse (Liu et al., 2023). For SCRUB, we adopt the provided hyperparameter settings in the code base (small scale for MLP and large scale for CNN and ResNet). For NegGrad+, we tune the hyperparameter manually and find the optimal setting α=0.95, epoch=1, and learning rate=1e-4. For ℓ-1 Sparse (Liu et al., 2023), the proposed framework consists of two parts, pruning and unlearning. The pruning step is orthogonal to all other unlearning baselines. Hence, we drop the pruning step for all baselines to ensure a fair comparison (similar to the case that we use PGD during training for all baselines). It is worth noting that other than pruning, ℓ-1 sparse unlearning proposes a novel unlearning technique as well (adding an ℓ-1 penalty to the objective). As a result, we include ℓ-1 sparse unlearning without pruning in the additional results shown in Table 6. Comparing the additional results with Table 1, we can observe that our proposed certified method still has the most desirable and robust performance. Compared with the negative gradient method, NegGrad+ improves the utility over the retain and test sets but still has a lower performance on the forget set. With the ℓ-1 penalty, the ℓ-1 sparse unlearning has a distinct improvement compared with vanilla fine-tuning.