RRInf: Efficient Influence Function Estimation via Ridge Regression for Large Language Models and Text-to-Image Diffusion Models

Zhuozhuo Tu[1], Cheng Chen[2,3*], Yuxuan Du[4]
[1] The University of Sydney, Australia
[2] CFAR, Agency for Science, Technology and Research, Singapore
[3] IHPC, Agency for Science, Technology and Research, Singapore
[4] College of Computing and Data Science, Nanyang Technological University, Singapore
tuzhuoz@gmail.com, chengchen.martin@gmail.com, yuxuan.du@ntu.edu.sg

Abstract

The quality of data plays a vital role in the development of Large-scale Generative Models. Understanding how important a data point is for a generative model is essential for explaining its behavior and improving the performance. The influence function provides a framework for quantifying the impact of individual training data on model predictions. However, the high computational cost has hindered their applicability in large-scale applications. In this work, we present RRInf, a novel and principled method for estimating influence function in large-scale generative AI models. We show that influence function estimation can be transformed into a ridge regression problem. Based on this insight, we develop an algorithm that is efficient and scalable to large models. Experiments on noisy data detection and influential data identification tasks demonstrate that RRInf outperforms existing methods in terms of both efficiency and effectiveness for commonly used large models: RoBERTa-large, Llama-2-13B-chat, Llama-3-8B and stable-diffusion-v1.5[1].

1 Introduction

Large Language Models (LLMs) and Text-to-Image models have achieved remarkable performance across a variety of tasks such as reading comprehension, natural language inference and image editing (Brown et al., 2020; Chowdhery et al., 2023; Dubey et al., 2024; Rombach et al., 2022a). The dominant approach to developing large generative models utilises unsupervised pre-training followed by supervised fine-tuning, both of which require collecting a large and diverse corpus of data. A major factor in the development process is the quality of data, which significantly impacts the performance of generative models on downstream tasks. Despite that, cleaning and filtering techniques could help address data quality issues, even state-of-the-art models can generate incorrect answers or biased outputs (Weidinger et al., 2021; Abid et al., 2021; Ferrara, 2023). This raises a question of whether a training point is beneficial or detrimental to model performance.

Influence function provides a method for answering this question by quantifying the impact of a training point on model predictions (Hampel, 1974; Cook and Weisberg, 1980; Martin and Yohai, 1986). Despite being well-grounded in robust statistics, calculating the influence function is expensive because of the intensive operation of inverting the Hessian matrix, which limits its applicability in modern machine learning problems. To tackle this difficulty, Koh and Liang (2017) proposes to use second-order optimisation techniques (Agarwal et al., 2017a) to approximate the influence function, which avoids explicitly computing the Hessian inverse. Schioppa et al. (2022) considers a diagonalised form of the Hessian based on Arnoldi iteration to simplify the matrix inversion in the influence function. Grosse et al. (2023) propose to use the Eigenvalue-corrected Kronecker-Factored Approximate Curvature to approximate Hessian matrix inversion by applying eigenvalue decomposition. These existing methods require additional complex operations and thus can not readily scale to very large models. Recently, Kwon et al. (2024) proposed a closed-form expression for approximating the influence function. While their algorithm can be applied to LLMs and diffusion models, the approximation error depends on the number of learnable model parameters and is thus only suitable for parameter-efficient fine-tuning like LoRA.

In this work, we propose a new method called RRInf for computing the influence function, which is computationally efficient and can easily be applied to large-scale machine learning models. In particular, we consider computing the influence

*Corresponding author.
[1]The code is available at https://github.com/tuzz0210/RRInf.

function of each training data simultaneously in the form of a vector-matrix-matrix product. Then, we formulate a *Ridge Regression* problem and show that the influence function is the solution to that regression problem. This way, the complex Hessian matrix inversion step is bypassed, and computing the influence function is transformed into solving an optimisation problem. A key advantage of this re-formulation is that *Ridge Regression* is a strongly convex optimisation problem which has been well-studied in the optimisation literature, with a variety of optimisation methods developed as well as their convergence guarantees. We propose a stochastic gradient-based algorithm to solve the *Ridge Regression* problem, which is easy to implement and very efficient in large-scale settings, making it attractive for use in large models like LLMs and diffusion models. Our main contributions are summarised as follows.

- We propose a new method for estimating the influence function. In particular, we formulate a ridge regression problem and prove its equivalence to the influence function.
- We develop a normalised stochastic gradient algorithm for solving the ridge regression, which is efficient in terms of both computational and memory complexity and can be applied to LLMs and diffusion models.
- RRInf achieves superior efficiency and performance over existing methods. For instance, improving mislabeled data detection by 4.9% on SST2 and reducing runtime from 1123.98s (DataInf) to 141.07s on text generation with Llama-2-13B-chat.

The remainder of this paper is structured as follows. Section 2 defines the influence function and notations. In Section 3, we present our *Ridge Regression* formulation and the corresponding algorithm. The experimental results are provided in Section 4. Section 5 describes the related work, and we conclude and discuss future directions in Section 6.

2 Preliminaries

We consider a standard statistical learning framework. Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input and label space, respectively. We assume that training data $S^{train} = \{(x_i^{train}, y_i^{train})\}_{i=1}^n$ is drawn *i.i.d.* from some unknown distribution P over $\mathcal{X} \times \mathcal{Y}$. Given a loss function $l : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ and a model space $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ where model parameters $\theta \in \Theta$, the empirical risk minimizer is defined as $\hat{\theta} := \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^n l(\theta, i)$ where we write $l(\theta, i) := l(f_\theta(x_i^{train}), y_i^{train})$ for notational convenience. The quality of the prediction made by a model f_θ can be measured on a set of test sample $S^{test} = \{(x_j^{test}, y_j^{test})\}_{j=1}^m$ by $L(\theta, S^{test}) = \frac{1}{m}\sum_{j=1}^m l(f_\theta(x_j^{test}), y_j^{test})$.

2.1 Influence Function

Our goal is to understand the impact of individual training points on a model's prediction. This can be formalized by influence function as follows (Hampel, 1974; Cook and Weisberg, 1980; Martin and Yohai, 1986; Koh and Liang, 2017; Kwon et al., 2024). For training point $(x_k^{train}, y_k^{train})$ and $\epsilon \in \mathbb{R}$, consider a ϵ-weighted risk minimization problem: $\hat{\theta}_{k,\epsilon} := \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^n l(\theta, i) + \epsilon \cdot l(\theta, k)$. If the loss function $l(f_\theta(x), y)$ is twice-differentiable and strongly convex in θ, the influence of the k-th training point on prediction loss is defined as the derivative of $L(\hat{\theta}_{k,\epsilon}, S^{test})$ at $\epsilon = 0$:

$$\begin{aligned}\mathcal{I}(k) &:= \left.\frac{dL(\hat{\theta}_{k,\epsilon}, S^{test})}{d\epsilon}\right|_{\epsilon=0} \\ &= \left.\frac{dL(\hat{\theta}_{k,\epsilon}, S^{test})}{d\hat{\theta}_{k,\epsilon}}\right|^T_{\hat{\theta}_{k,\epsilon}=\hat{\theta}} \left.\frac{d\hat{\theta}_{k,\epsilon}}{d\epsilon}\right|_{\epsilon=0}, \\ &= -\left(\frac{1}{m}\sum_{j=1}^m \nabla_\theta l(f_{\hat{\theta}}(x_j^{test}), y_j^{test})\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta l(\hat{\theta}, k)\end{aligned}$$

where ∇_θ denotes the gradient with respect to θ and $H_{\hat{\theta}} := \frac{1}{n}\sum_{i=1}^n \nabla_\theta^2 l(\hat{\theta}, i)$ is the Hessian matrix of empirical loss.

While the influence function $\mathcal{I}(k)$ provides an intuitive interpretation of how a training point affects the prediction loss, its computation in general is expensive due to the Hessian matrix. Luckily, for the negative log-likelihood loss function, the second-order Hessian enjoys a simplified form (Bartlett, 1953). To be more specific, suppose that $l(f_\theta(x), y) = -log\, p(y|f_\theta(x))$ where $p(y|f_\theta(x))$ is a probability density function of (x, y) at θ, Bartlett's second identity implies that $\mathbb{E}_{(x,y)\sim p}[\nabla_\theta^2 l(f_\theta(x), y)] = \mathbb{E}_{(x,y)\sim p}[\nabla_\theta l(f_\theta(x), y)\nabla_\theta l(f_\theta(x), y)^T]$. Thus, the Hessian matrix $H_{\hat{\theta}}$ can be replaced by the second moment of first-order gradient, i.e., $G_{\hat{\theta}} := \frac{1}{n}\sum_{i=1}^n \nabla_\theta l(\hat{\theta}, i)\nabla_\theta l(\hat{\theta}, i)^T$, yielding the following influence function (Kwon et al., 2024)

$$-\left(\frac{1}{m}\sum_{j=1}^m \nabla_\theta l(f_{\hat{\theta}}(x_j^{test}), y_j^{test})\right)^T G_{\hat{\theta}}^{-1} \nabla_\theta l(\hat{\theta}, k). \tag{1}$$

In the paper, we focus on the negative log-likelihood loss function, which is equivalent to cross-entropy loss and used for training many large language models. However, there are practical challenges in applying the influence function (1). When the model size is larger than the sample size, which is usually the case for large-scale machine learning models, the matrix $G_{\hat{\theta}}$ is not invertible as the rank is at most n and not full-rank. To address this issue, a damping Hessian approach is employed where a small positive constant is added to the diagonal entries of $G_{\hat{\theta}}$ to ensure positive definiteness, giving the following influence function:

$$-\left(\frac{1}{m}\sum_{j=1}^{m}\nabla_{\theta}l(f_{\hat{\theta}}(x_j^{test}),y_j^{test})\right)^T\left(G_{\hat{\theta}}+\lambda\mathbb{I}\right)^{-1}\nabla_{\theta}l(\hat{\theta},k) \tag{2}$$

where λ is some positive constant and $\mathbb{I}$ is the identity matrix. The influence function (2) stabilises the computation and has become the standard estimand in the literature (Grosse et al., 2023; Kwon et al., 2024). In the rest of the paper, we aim to calculate the influence function (2) efficiently for large-scale models like LLMs.

3 Methodology

In this section, we first show that the influence function (2) can be formulated as a *Ridge Regression* problem. Then we propose algorithms for solving the regression problem, which are efficient and easy to implement.

3.1 Ridge Regression Formulation

Before introducing the formulation, we define a few notations. Let $\theta \in \mathbb{R}^d$, i.e., d-dimensional model, where the ι-th component of model is given by θ_ι. We collect all training data gradient of loss $\nabla_\theta l(\hat{\theta},1), \nabla_\theta l(\hat{\theta},2), \cdots, \nabla_\theta l(\hat{\theta},n) \in \mathbb{R}^d$ as a matrix denoted by Φ where $\Phi \in \mathbb{R}^{d\times n}$. The ι-th row of the matrix Φ is denoted by $\nabla_\iota l(\hat{\theta},:) := (\nabla_{\theta_\iota} l(\hat{\theta},1), \nabla_{\theta_\iota} l(\hat{\theta},2), \cdots, \nabla_{\theta_\iota} l(\hat{\theta},n))^T \in \mathbb{R}^n$ where ∇_{θ_ι} represents the gradient with respect to θ_ι. We also define $\upsilon := -\frac{n}{m}\sum_{j=1}^{m}\nabla_\theta l(f_{\hat{\theta}}(x_j^{test}), y_j^{test}) \in \mathbb{R}^d$. Instead of calculating the influence function (2) for each training data one by one, we consider computing them simultaneously by concatenating all influence into a vector $\mathcal{I} := (\mathcal{I}(1), \mathcal{I}(2), \cdots, \mathcal{I}(n))^T \in \mathbb{R}^n$. Then, using the notations, the influence function (2) can be rewritten in a vector-matrix-matrix product form:

$$\mathcal{I}^T = \upsilon^T(\Phi\Phi^T + n\lambda\mathbb{I}_d)^{-1}\Phi, \tag{3}$$

where $\mathbb{I}_d \in \mathbb{R}^{d\times d}$ is the identity matrix of size d.

Computing the influence function in equation (3) is prohibitively expensive for large-scale machine learning models as it requires solving the inverse of the huge matrix $\Phi\Phi^T + n\lambda\mathbb{I} \in \mathbb{R}^{d\times d}$. To address this issue, we propose a *Ridge Regression* problem $J(\omega)$ defined as follows:

$$\begin{aligned}\min\nolimits_{\omega\in\mathbb{R}^n} J(\omega) &:= \frac{1}{d}\|\Phi\omega-\upsilon\|_2^2 + \tilde{\lambda}\|\omega\|_2^2 \\ &= \frac{1}{d}\sum\nolimits_{\iota=1}^{d}(\nabla_\iota l(\hat{\theta},:)^T\omega - \upsilon_\iota)^2 + \tilde{\lambda}\|\omega\|_2^2\end{aligned}, \tag{4}$$

where ω is the parameter to be optimized, $\tilde{\lambda} := n\lambda/d$ and $\|\cdot\|_2$ denotes the Euclidean norm, and show the equivalence between the influence function (3) and the *Ridge Regression* problem (4) in the following theorem.

Theorem 1. Let $\hat{\omega}$ be a solution to the *Ridge Regression* problem (4). Then, the influence function $\mathcal{I} = \hat{\omega}$.

Proof. First, by calculation, we have

$$\begin{aligned}J(\omega) &= \frac{1}{d}\|\Phi\omega-\upsilon\|_2^2 + \tilde{\lambda}\|\omega\|^2, \\ &= \frac{1}{d}(\omega^T(\Phi^T\Phi + n\lambda\mathbb{I}_n)\omega - 2\upsilon^T\Phi\omega + \upsilon^T\upsilon)\end{aligned}$$

where $\mathbb{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix of size n, and the Hessian matrix of $J(\omega)$

$$\nabla^2 J(\omega) = \frac{2}{d}(\Phi^T\Phi + n\lambda\mathbb{I}_n) \ .$$

It can be verified that $\nabla^2 J(\omega)$ is positive definite from the definition, and thus $J(\omega)$ is a strongly convex function. By the optimality condition, the minimiser of a differentiable and convex function should cancel the gradient (Boyd and Vandenberghe, 2004), i.e., $\hat{\omega}$ satisfies

$$\nabla J(\hat{\omega}) = \frac{2}{d}((\Phi^T\Phi + n\lambda\mathbb{I}_n)\hat{\omega} - \Phi^T\upsilon) = 0 \ ,$$

which is equivalent to

$$(\Phi^T\Phi + n\lambda\mathbb{I}_n)\hat{\omega} = \Phi^T\upsilon.$$

We now prove the above equality holds for the influence function $\mathcal{I}$. Taking transpose and multiplying by $(\Phi^T\Phi + n\lambda\mathbb{I}_n)$ on both sides of (3) yield the following:

$$\begin{aligned}&(\Phi^T\Phi + n\lambda\mathbb{I}_n)\mathcal{I} \\ &= (\Phi^T\Phi + n\lambda\mathbb{I}_n)(\Phi^T(\Phi\Phi^T + n\lambda\mathbb{I}_d)^{-1}\upsilon) \\ &= \Phi^T(\Phi\Phi^T + n\lambda\mathbb{I}_d)(\Phi\Phi^T + n\lambda\mathbb{I}_d)^{-1}\upsilon \\ &= \Phi^T\upsilon\end{aligned},$$

where the second equality uses $\mathbb{I}_n \Phi^T = \Phi^T \mathbb{I}_d$. Thus, the influence function (3) is also a solution to *Ridge Regression* problem (4). Furthermore, a strongly convex function is strictly convex, implying that the solution is unique (Boyd and Vandenberghe, 2004). Therefore $\mathcal{I}$ must be $\hat{\omega}$, which completes the proof. □

Remark 1. The *Ridge Regression* formulation (4) can be extended to multiple test sample case, e.g., estimating the influence of training data on C test samples $\{S^{test_c}\}_{c=1}^{C}$ where $S^{test_c} = \{(x_j^{test_c}, y_j^{test_c})\}_{j=1}^{m_c}$. Rather than replacing υ in (4) with $\upsilon_c := -\frac{n}{m_c}\sum_{j=1}^{m_c} \nabla_\theta l(f_{\hat{\theta}}(x_j^{test_c}), y_j^{test_c})$ and solving C different *Ridge Regression* problems, an equivalent and more efficient approach is to concatenate these υ_cs into a matrix $\Upsilon := [\upsilon_1, \cdots, \upsilon_C] \in \mathbb{R}^{d\times C}$ and solve the following problem:

$$\min_{\Omega\in\mathbb{R}^{n\times C}} J(\Omega) := \frac{1}{d}||\Phi\Omega - \Upsilon||_F^2 + \tilde{\lambda}||\Omega||_F^2 \ ,$$

where matrix $\Omega = [\omega_1, \omega_2, \cdots, \omega_C]$ is the optimisation variable, each column of which corresponds to the influence vector on one test sample, and $||\cdot||_F$ represents the Frobenius norm.

Theorem 1 shows that computing the influence function is equivalent to finding a solution to a ridge regression problem. In the following subsection, we propose algorithms for solving (4) which bypass the need for complex matrix inversion.

3.2 Influence Estimation

Ridge Regression is a popular method for solving the least squares problem by including a squared l_2-norm regularisation. The method is first introduced by Hoerl and Kennard (1970) as a means to mitigate the problem of collinearity in regression analysis and has become a widely used technique in machine learning to prevent overfitting and improve generalisation performance (Kuhn et al., 2013; Goodfellow et al., 2016). The main approaches for optimising ridge regression involve forming a system of linear equations (also known as normal equations) and solving these equations using techniques like Cholesky decomposition (Trefethen and Bau, 2022), QR factorisation (Golub and Van Loan, 2013) and Singular Value Decomposition (Golub and Reinsch, 1971). However, these methods require expensive matrix decomposition operations and become impractical when applied to large models like LLMs.

To overcome this challenge, we propose a stochastic gradient method for solving *Ridge Regression* problem (4) which makes use of approximate gradient information to refine a solution repeatedly and has been used for a variety of optimisation problems. The stochastic gradient method is simple and efficient, making it attractive for large-scale problems, and has become the dominant optimisation method for modern machine learning. In particular, the *Ridge Regression* problem $J(\omega)$ is strongly convex, and the stochastic gradient method is guaranteed to converge to the solution of the problem (Bottou et al., 2018; Bubeck, 2014).

Let us begin with the basic gradient descent method. To apply gradient descent to (4), we initiate at some point ω_0, e.g., $\omega_0 = 0$, then recursively updates:

$$\begin{aligned}
&\omega_{t+1} = \omega_t - \eta_t \nabla_\omega J(\omega_t) \\
&= \omega_t - \frac{\eta_t}{d}(2\Phi^T\Phi\omega_t - 2\Phi^T\upsilon + 2d\tilde{\lambda}\omega_t), \\
&= \omega_t - \frac{2\eta_t}{d}\sum_{\iota=1}^{d}(\nabla_\iota l(\hat{\theta},:)(\nabla_\iota l(\hat{\theta},:)^T\omega_t - \upsilon_\iota) + \tilde{\lambda}\omega_t),
\end{aligned}$$

where η_t is the learning rate parameter. With carefully chosen learning rate η_t, ω_t converges to $\hat{\omega}$ as t increases (Nesterov et al., 2018; Boyd and Vandenberghe, 2004). In practice, it is often assumed that gradient descent runs for a reasonable number of iterations. When the number of iterations is finite, the total computational complexity for gradient descent is $\mathcal{O}(dn)$[2]. However, for very large models, it can be very expensive to go through all model parameters every iteration to do $\nabla_\omega J(\omega_t)$ calculation. Moreover, deep neural networks are typically overparameterised and have a large number of duplicate neurons. This redundancy in neural networks suggests that using gradients of all model parameters in gradient descent is inefficient.

This motivates us to consider a stochastic gradient (SG) approach[3]. Instead of employing full gradient information, the stochastic gradient approach uses an unbiased estimator of $\nabla_\omega J(\omega_t)$ which samples only a small subset of model parameters per iteration, resulting in significant speedups. More precisely, at each iteration t, SG takes a sum over

[2]Each iteration requires one n-dimensional vector-vector product $\nabla_\iota l(\hat{\theta},:)^T\omega_t$ and one scalar multiplication $(\nabla_\iota l(\hat{\theta},:)^T\omega_t - \upsilon_\iota)\nabla_\iota l(\hat{\theta},:)$ for every model parameter θ_ι, which costs $\mathcal{O}(n)$ time. Since there are d parameters, the total time is $\mathcal{O}(dn)$ which is the same as (Kwon et al., 2024).

[3]Also referred to as mini-batch in the optimisation literature.

a small subset denoted by $\mathcal{D}_t$ of model parameter which are randomly chosen from $\{1, 2, \cdots, d\}$: $\omega_{t+1} = \omega_t - \frac{2\eta_t}{|\mathcal{D}_t|} \sum_{\iota \in \mathcal{D}_t} (\nabla_\iota l(\hat{\theta}, :)(\nabla_\iota l(\hat{\theta}, :)^T \omega_t - \upsilon_\iota) + \tilde{\lambda}\omega_t)$, where $|\mathcal{D}_t|$ is the size of $\mathcal{D}_t$. Despite that SG approach improves efficiency, it faces some challenges in practice. First, the learning rate η_t is hard to choose and often requires substantial tuning or carefully designed schedules for achieving good performance. Second, the model gradient $\nabla_\iota l(\hat{\theta}, :)$ varies significantly across the parameters, especially in large models, making the algorithm unstable. To address these issues, we propose a normalised stochastic gradient descent method called RRInf in which the gradient is normalised with the squared norm of $\nabla_\iota l(\hat{\theta}, :)$:

$$\omega_{t+1} = \omega_t - \frac{2\eta}{|\mathcal{D}_t|} \sum_{\iota \in \mathcal{D}_t} \left(\frac{\nabla_\iota l(\hat{\theta}, :)}{||\nabla_\iota l(\hat{\theta}, :)||^2} \left(\nabla_\iota l(\hat{\theta}, :)^T \omega_t - \upsilon_\iota \right) + \tilde{\lambda}\omega_t \right), \tag{5}$$

where the learning rate η now is fixed. We see that RRInf may be viewed as an adaptive SG method in which $\eta_t = \frac{\eta}{||\nabla_\iota l(\hat{\theta}, :)||^2}$. RRInf has important advantages over gradient descent. First, RRInf performs $\frac{d}{|\mathcal{D}_t|}$ updates for one sweep through all model parameters while gradient descent performs only one step, which is more efficient. Second, each iteration of RRInf is very cheap, involving only the gradient corresponding to a small subset of model parameters, which takes $\mathcal{O}(|\mathcal{D}_t|n)$ time. The value of $|\mathcal{D}_t|$ is typically many times smaller than d. As will be shown in Section 4, for most models, RRInf behaves well by using $|\mathcal{D}_t| = 1$. For this case, the time cost is $\mathcal{O}(n)$, which is independent of the dimension of a model. Therefore, RRInf is particularly well-suited for very large models like LLMs and diffusion models.

Remark 2. The RRInf algorithm can be easily extended to the multiple test sample case in Remark 1 as follows:

$$\Omega_{t+1} = \Omega_t - \frac{2\eta}{|\mathcal{D}_t|} \sum_{\iota \in \mathcal{D}_t} \left(\frac{\nabla_\iota l(\hat{\theta}, :) \otimes \left(\Omega_t^T \nabla_\iota l(\hat{\theta}, :) - \Upsilon_\iota \right)}{||\nabla_\iota l(\hat{\theta}, :)||^2} + \tilde{\lambda}\Omega_t \right),$$

where Ω_t is the influence matrix at iteration t, $\otimes$ denotes outer product and $\Upsilon_\iota \in \mathbb{R}^C$ corresponds to the ι-th row of Υ.

4 Experiments

We validate our proposed method, **RRInf**, on two key tasks: **Mislabeled Data Detection** and **Influential Data Identification**. In addition, we conduct an ablation study to examine the impact of different batch size (stochastic or full gradient) and sampling strategies (sampling a layer or a subset of neurons) on the performance of RRInf. Overall, RRInf consistently demonstrates superior performance and robustness across all datasets for both tasks.

Experimental set-up. In this section, we present our experimental findings and key observations on the performance of the proposed **RRInf** algorithm relative to the following baselines: Hessian-free which calculates $v^T \nabla_\theta l(\hat{\theta}, k)/n$ (Pruthi et al., 2020), DataInf (Kwon et al., 2024), and LiSSA (Agarwal et al., 2017b). We consider publicly available and widely used LLMs and diffusion models: RoBERTa (Liu et al., 2019), Llama-2-13B-chat (Touvron et al., 2023), Llama-3-8B (Grattafiori et al., 2024) and stable-diffusion-v1.5 (Rombach et al., 2022b) where the first is used for mislabeled data detection task and the other three are for the influential data identification. In all experiments, we first fine-tune a model and then compute the influence function. For LLMs, we simply sample a single neuron at each iteration in RRInf, i.e., $|\mathcal{D}_t| = 1$, which is sufficient to achieve good performance. For diffusion models, we propose to sample an entire layer per iteration from the total set of layers which is shown to be more efficient than randomly choosing a subset of parameters in the ablation study. See more implementation details in Appendix A. As summarised in Table 1 and 2, RRInf consistently outperforms every baseline on all models.

4.1 Mislabeled Data Detection

Mislabeled samples are known to degrade model performance, and including such samples during training often increases the loss. As a result, their influence values tend to be higher than those of correctly labeled samples. In this experiment, we evaluate the performance of RRInf alongside baseline influence computation methods, including DataInf, Hessian-Free, and LiSSA, on the task of mislabeled data detection using five public GLUE benchmark datasets: QNLI (Wang et al., 2018), MRPC (Dolan and Brockett, 2005), QQP (Iyer et al., 2017), SST-2 (Socher et al., 2013), and RTE (Dagan et al., 2006), all based on the *RoBERTa* backbone (Liu et al., 2019). Following prior work, we synthetically generate mislabeled training data by flipping the binary label for 20% randomly chosen training data points and use the ground truth annotations of

Dataset	Hessian-free	DataInf	LiSSA	RRInf
SST2	$45.60\% \pm 5.59\%$	$83.62\% \pm 8.15\%$	$45.38\% \pm 5.52\%$	**$88.52\% \pm 4.64\%$**
QQP	$63.85\% \pm 0.89\%$	$70.43\% \pm 2.35\%$	$63.86\% \pm 0.90\%$	**$73.39\% \pm 2.28\%$**
MRPC	$67.60\% \pm 0.98\%$	$70.02\% \pm 2.61\%$	$67.57\% \pm 0.96\%$	**$75.67\% \pm 0.74\%$**
RTE	$50.78\% \pm 0.68\%$	$49.74\% \pm 1.17\%$	$50.45\% \pm 0.85\%$	**$52.01\% \pm 0.59\%$**
QNLI	$50.06\% \pm 0.78\%$	$53.55\% \pm 1.94\%$	$50.04\% \pm 0.76\%$	**$58.36\% \pm 1.79\%$**

Table 1: AUC comparison of Hessian-free, DataInf, LiSSA and RRInf(Our) on Mislabeled Data Detection tasks. Mislabeled Data Detection Performance Comparison Across Tasks and Methods (Mean AUC ± Standard Deviation), where higher values indicate better performance.

Text Generation	**Method**	**Class detection (AUC) ↑**	**Class detection (Recall) ↑**
`LLaMA-2-13B-chat`	Hessian-free	$60.0\% \pm 7.6\%$	$33.0\% \pm 8.3\%$
`LLaMA-2-13B-chat`	DataInf	$65.7\% \pm 9.1\%$	$39.0\% \pm 9.9\%$
`LLaMA-2-13B-chat`	RRInf	**$67.5\% \pm 8.4\%$**	**$41.9\% \pm 9.1\%$**
`LLaMA-3-8B`	Hessian-free	$51.6\% \pm 9.3\%$	$27.1\% \pm 9.8\%$
`LLaMA-3-8B`	DataInf	$59.3\% \pm 11.0\%$	$35.1\% \pm 11.0\%$
`LLaMA-3-8B`	RRInf	**$66.3\% \pm 9.1\%$**	**$40.4\% \pm 11.0\%$**
Text-to-Image Style Generation	**Method**	**Class detection (AUC) ↑**	**Class detection (Recall) ↑**
`Stable-Diffusion-v1.5`	Hessian-free	$58.8\% \pm 7.5\%$	$43.7\% \pm 6.8\%$
`Stable-Diffusion-v1.5`	DataInf	$62.6\% \pm 11.5\%$	$49.0\% \pm 10.4\%$
`Stable-Diffusion-v1.5`	RRInf	**$64.3\% \pm 11.1\%$**	**$50.6\% \pm 9.8\%$**
Text-to-Image Subject Generation	**Method**	**Class detection (AUC) ↑**	**Class detection (Recall) ↑**
`Stable-Diffusion-v1.5`	Hessian-free	$62.4\% \pm 18.4\%$	$14.1\% \pm 19.2\%$
`Stable-Diffusion-v1.5`	DataInf	$61.4\% \pm 21.6\%$	$23.6\% \pm 23.4\%$
`Stable-Diffusion-v1.5`	RRInf	**$63.9\% \pm 20.9\%$**	**$28.3\% \pm 24.9\%$**

Table 2: AUC and Recall Comparison of Hessian-Free and DataInf on Influential Data Identification Tasks. LiSSA is omitted from the experiments due to poor model stability. The results report for AUC and recall across the test dataset, shown as "average ± standard deviation", where higher values indicate better performance.

mislabeled samples (one for mislabeled data and zero for clean data) for assessing the quality of the influence estimates. We adopt the Area Under the ROC Curve (AUC) as the evaluation metric, which measures the probability that a randomly selected mislabeled sample receives a higher influence score than a clean sample. Thus, a well-performing influence function is expected to assign higher scores to mislabeled instances, resulting in higher AUC values.

Results RRInf consistently outperforms all other methods across all datasets, demonstrating significantly better detection performance—for example, +4.9% on SST2, +2.96% on QQP, +5.65% on MRPC, +2.27% on RTE, and +4.81% on QNLI compared to DataInf. This indicates that RRInf not only surpasses the baselines on relatively easier tasks such as SST2, QQP, and MRPC, but also on more challenging tasks like RTE, on which DataInf is worse than Hessian-Free and LiSSA. In terms of runtime efficiency, RRInf demonstrates performance comparable to DataInf, further highlighting that it is both effective and efficient when applied to large language models such as RoBERTa. For example, on SST2 dataset, RRInf takes 10.2 seconds while DataInf and LiSSA take 10.7 and 57.1 seconds respectively. For a detailed runtime comparison of RRInf and the baselines, please refer to Table 4 in the Appendix.

4.2 Influential Data Identification

To further demonstrate the effectiveness of RRInf, we evaluate its performance in identifying influential data points across two tasks: text generation and text-to-image generation. We use Llama-2-13B-chat (Touvron et al., 2023) and Llama-3-8B model (Grattafiori et al., 2024) for the text generation and stable-diffusion-v1.5 model (Rombach et al., 2022b) for text-to-image generation tasks. All models are open-source and widely adopted within the research community.

Text Generation We use the SVAMP dataset (Patel et al., 2021) which consists of English math word problems with grade level up to 4. SVAMP comprises four classes: Addition, Subtraction, Multiplication and Division. For each class, we select 75 examples for training and 25 examples for testing.

Text-to-Image Generation We consider two subtasks: style generation and subject generation. The style generation task includes three publicly available image-text pair datasets, each representing a distinct style: Cartoons (Adler, 2023), Pixel-art (Jain, 2023) and Line sketches (Chowdhury et al., 2022). For each style, we use 200 training image-text pairs and 50 test image-text pairs, totalling 600 training and 150 test data points. For subject generation, we use the DreamBooth dataset (Ruiz et al., 2022), which includes 30 different subjects. Following Kwon et al. (2024), we choose 3 data points from each subject for training, and the remaining as validation. We add a unique random string to each subject in the prompt. For example, we use "a AXNkV dog" and "a DxE3K dog" to differentiate two different dogs.

To evaluate the influence estimate, we use two metrics: AUC and Recall, following Kwon et al. (2024). For each test data, a pseudo label is assigned to every training data which is one if its class is the same as the test data's class, and zero otherwise. Then, the AUC between the negative influence function values and the pseudo labels is calculated. We also compute the Recall for each test data which is the percentage of training data with the same class as the test data among the s smallest influential training point where s is the number of training example per class. Intuitively, if a training point has the same class as the test data, adding it in the training will decrease the loss, meaning its influence function should be negative, and these two metrics intend to assess whether an influence estimate can identify training data having the same class as a test data to be more helpful than one that belongs to a different class. We report average AUC and average Recall across all test points as class detection AUC and Recall respectively. Note that this is the multiple test sample case discussed in Remark 1 where we compute the influence function of training data on each test point.

Results RRInf outperforms both DataInf and Hessian-free methods on text and text-to-image generation. For text generation task, RRInf achieves best identification across all metrics on both Llama-2-13B-chat and Llama-3-8B models. In particular, when changing the model from Llama-2-13B-chat to Llama-3-8B, RRInf still performs well, while the performance of Hessian-free and DataInf drops significantly. On the style generation task, RRInf achieves the highest AUC (64.3%) and recall (50.6%), outperforming DataInf by +1.7% (AUC) and +1.6% (Recall), and Hessian-free by +5.5% (AUC) and +6.9% (Recall). It also exhibits lower standard deviation, indicating greater stability and further demonstrating the effectiveness of our normalised algorithm. It is worth noting that on the subject generation task RRInf performs best on both AUC and Recall, while DataInf achieves a higher Recall at a cost of lower AUC than Hessian-free. In terms of runtime efficiency, we highlight that RRInf computes the influence function for all test data simultaneously, as noted in Remark 1. Consequently, on the style generation task using the stable-diffusion-v1.5 model, our method completes the task in 74.94 seconds, compared to 1950.18 seconds for DataInf—making our approach approximately **26×** faster. Similarly, for the text generation task trained on Llama-2-13B-chat, our method achieves a **8×** improvement in computational efficiency, requiring only 141.07 seconds. In contrast, DataInf takes 1123.98 seconds to complete.

4.3 Ablation Studies

In this section, we conduct a thorough ablation study on batch size and sampling strategies on the effectiveness of RRInf.

4.3.1 Stochastic vs. Full Gradient

In this subsection, we empirically compare RRInf with its full gradient version denoted by $\text{RRInf}_{\text{full}}$ on the style generation task (see Table 7). Specifically, $\text{RRInf}_{\text{full}}$ uses all model parameters, i.e., $|\mathcal{D}_t| = d$, whereas RRInf samples an entire layer from the total set of layers at each iteration. Although $\text{RRInf}_{\text{full}}$ achieves moderately higher AUC (+0.015) and recall (+0.013), it does so at the expense of significantly greater computational resources due to full-batch updates across entire layers. In contrast, RRInf uses a single layer out of the entire set of layers. While the stochastic version incurs only a slight drop in performance, it dramatically reduce computational cost, making it more practical for large-scale or resource-constrained scenarios, such as in large language models and diffusion models. These empirical results are consistent with the theoretical advantages of the SG method for high-dimensional models, which shows that SG is more computationally efficient than gradient descent under limited resources (Bottou et al., 2018). This supports our motivation for proposing RRInf as a more feasible alternative for large-scale models like LLMs to gradient descent, which applies full-batch gradient update with entire model parameters.

4.3.2 Sampling Strategy

In this subsection, we conduct an ablation study on different sampling strategies on the performance of RRInf. We consider a variant of RRInf denoted by RRInf_{mb}, which randomly picks a small subset of neurons instead of a layer on the text-to-image style generation task. We also use various sample sizes ranging from 1 to 2048. The results are shown in Figure 1. AUC increases from 60.3% to 63.4% (+3.1%) when increasing batch size from 1 to 64. The improvement plateaus at batch size between 64-128. Recall gains from 45% to 50.1 %(+5.1%) over the same batch size range. The results indicate that small batches (1-32) are more prone to unstable updates that undermine performance. Later increases (up to 2048) show less than 0.3% changes in AUC score and recall score. From the results, we can see that RRInf_{mb} even with the optimal batch size is slightly inferior to RRInf, suggesting that a layer-level sampling may be more suitable than neuron-level sampling in diffusion models.

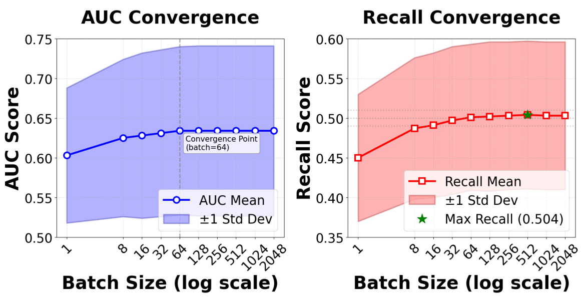

Figure 1: Convergence of the performance of $\textbf{RRInf}_{\text{mb}}$ with respect to sampling batch size. The performance converges at a batch size of 64. For further details, refer to Table 8 in the Appendix.

5 Related Work

Data valuation has emerged as an active research area in machine learning that quantifies the value and importance of individual data points on a model's prediction (Jia et al., 2019; Sim et al., 2022; Hammoudeh and Lowd, 2024). One widely used class of data valuation methods is based on retraining models with and without specific samples and computing the influence of a data point as model outputs, including Leave-One-Out (Cook, 1977), downsampling (Feldman and Zhang, 2020), Shapley value methods (Ghorbani and Zou, 2019; Ghorbani et al., 2020; Kwon and Zou, 2021; Garrido Lucero et al., 2024) and Out-of-bag (Kwon and Zou, 2023). Gradient-Based Approaches is another family of methods for data valuation which use training gradients to calculate an importance score. The influence function studied in this work falls into this category. Compared to retraining-based methods, influence function eliminates the need for repeated retraining, making it more applicable to very large models. There are other gradient-based approaches which exploit gradient information during the training of the model, including tracing gradient descent (Pruthi et al., 2020; Park et al., 2023), measuring gradient similarity(Evans et al., 2024) and dynamically self-weighting (Wibiral et al., 2024). Finally, some model-agnostic data valuation approaches have been proposed that do not rely on any specific machine learning model and are based solely on the data (Xu et al., 2021; Just et al., 2023). For a comprehensive review of these methods, we refer the reader to Wibiral et al. (2024).

The early notion of influence emerges out of robust statistics, with a focus on analysing the effects of perturbations on linear regression models (Cook, 1977; Cook and Weisberg, 1980; Cook et al., 1982). Despite its rich history, the influence function has not been widely used in modern machine learning until the work of Koh and Liang (2017). Since then, there have been substantial efforts in developing different influence estimation methods, and they have shown promising results on many real-world applications (Han et al., 2020; Han and Tsvetkov, 2020; Joaquin et al., 2024; Lin et al., 2024b,a). Calculating influence functions is computationally expensive as it involves inverting the Hessian matrix. To improve scalability, numerous speed-ups have been proposed, such as applying influence function only to the model's last layer (Barshan et al., 2020), using parallelizability (Guo et al., 2021) and projecting gradients onto a low-dimensional space (Choe et al., 2024). However, these methods either are too slow to be used in very large models or cause accuracy degradation. More recently, Kwon et al. (2024) proposes a closed-form influence approximation method and applies it to LLMs and diffusion models. But their method incurs an approximation error which increases with model dimension. While this paper focuses on the influence function (2), there are other variants of influence function in the literature. For example, Mlodozeniec et al. (2025) formulates an influence function based on a generalised Gauss-Newton matrix specifically tailored to diffusion models and estimates the influence function using Kronecker-Factored Approximate Curvature method akin to Grosse et al. (2023).

6 Conclusion

In this paper, we propose RRInf, a new method for estimating the influence function. We introduce a ridge regression formulation and show an equivalence between the ridge regression problem and the influence function. We then develop a normalised stochastic gradient algorithm for solving that problem, which is easy to implement and well-suited for large-scale generative AI models like LLMs and diffusion models. Experimental results show that RRInf can gain significant benefits in terms of both efficiency and effectiveness. There remain many avenues for future direction. Our algorithm is a modification of the SG method. It is well known that the SG method enjoys theoretical guarantees for strongly convex optimisation problems. Extending such a result to our algorithm might be an interesting topic for future work.

Limitations

This paper proposes a method for estimating the influence function and demonstrates its advantages through various experiments. Nevertheless, several limitations must be considered. First, our experiments focus primarily on fine-tuning task-specific datasets, despite the fact that our method also applies to pre-training. Assessing the impact of each data point on the pre-training stage is critical for improving the model's general capabilities, which are left for future research.

Evaluating the influence is hard given the lack of a ground truth. In the experiments, we use a metric assuming that mislabeled data would have a large influence value and data in the same class would have a negative influence score. Despite intuition, an objective and quantitative evaluation criterion is necessary for real-world applications.

In addition, we use a simple arithmetic math word problem dataset for text generation. While our method achieves the best performance, more complex tasks like commonsense and symbolic reasoning can be considered in future work.

Ethical and Broader Impact

This study proposes an algorithm for improving the efficiency of influence function estimation in large-scale generative models, potentially contributing to greater accessibility and energy savings in AI systems. It highlights the importance of training large language models (LLMs) more efficiently by selecting informative and representative corpus samples. This targeted approach not only enhances computational efficiency during both pretraining and fine-tuning but also contributes to better model interpretability and transparency. Efficient models can be more easily deployed, which raises concerns about their potential misuse (e.g., generating harmful or misleading content). Ethical deployment must be coupled with safeguards and responsible usage policies.

References

Abubakar Abid, Maheen Farooqi, and James Zou. 2021. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 298–306.

Doron Adler. 2023. Cartoon blip captions dataset. Accessed: 2025-05-18.

Naman Agarwal, Brian Bullins, and Elad Hazan. 2017a. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116):1–40.

Naman Agarwal, Brian Bullins, and Elad Hazan. 2017b. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116):1–40.

Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. 2020. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*, pages 1899–1909. PMLR.

MS Bartlett. 1953. Approximate confidence intervals. *Biometrika*, 40(1/2):12–19.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. 2018. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda

Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck. 2014. Theory of convex optimization for machine learning. *arXiv preprint arXiv:1405.4980*, 15.

Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, et al. 2024. What is your data worth to gpt? llm-scale data valuation with influence functions. *arXiv preprint arXiv:2405.13954*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Pinaki Nath Chowdhury, Aneeshan Sain, Ayan Kumar Bhunia, Tao Xiang, Yulia Gryaditskaya, and Yi-Zhe Song. 2022. Fs-coco: Towards understanding of freehand sketches of common objects in context. In *ECCV*.

R Dennis Cook. 1977. Detection of influential observation in linear regression. *Technometrics*, 19(1):15–18.

R Dennis Cook and Sanford Weisberg. 1980. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508.

RDWS Cook et al. 1982. Residuals and influence in regression.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges, Lecture Notes in Computer Science*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*.

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

Nathaniel J Evans, Gordon B Mills, Guanming Wu, Xubo Song, and Shannon McWeeney. 2024. Data valuation with gradient similarity. *ArXiv*, pages arXiv–2405.

Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891.

Emilio Ferrara. 2023. Should chatgpt be biased? challenges and risks of bias in large language models. *arXiv preprint arXiv:2304.03738*.

Felipe Garrido Lucero, Benjamin Heymann, Maxime Vono, Patrick Loiseau, and Vianney Perchet. 2024. Du-shapley: A shapley value proxy for efficient dataset valuation. *Advances in Neural Information Processing Systems*, 37:1973–2000.

Amirata Ghorbani, Michael Kim, and James Zou. 2020. A distributional framework for data valuation. In *International Conference on Machine Learning*, pages 3535–3544. PMLR.

Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pages 2242–2251. PMLR.

Gene H Golub and Christian Reinsch. 1971. Singular value decomposition and least squares solutions. In *Handbook for Automatic Computation: Volume II: Linear Algebra*, pages 134–151. Springer.

Gene H Golub and Charles F Van Loan. 2013. *Matrix computations*. JHU press.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT press Cambridge.

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

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. 2023. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*.

Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. 2021. Fastif: Scalable influence functions for efficient model interpretation and debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10333–10350.

Zayd Hammoudeh and Daniel Lowd. 2024. Training data influence analysis and estimation: A survey. *Machine Learning*, 113(5):2351–2403.

Frank R Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393.

Xiaochuang Han and Yulia Tsvetkov. 2020. Fortifying toxic speech detectors against veiled toxicity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7732–7739.

Xiaochuang Han, Byron C Wallace, and Yulia Tsvetkov. 2020. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563.

Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs. `https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs`.

Rahul Jain. 2023. Diffusiondb-pixelart dataset. Accessed: 2025-05-18.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas Spanos, and Dawn Song. 2019. Efficient task-specific data valuation for nearest neighbor algorithms. *Proceedings of the VLDB Endowment*, 12(11):1610–1623.

Ayrton Joaquin, Bin Wang, Zhengyuan Liu, Philippe Muller, Nicholas Asher, Brian Lim, and Nancy Chen. 2024. In2core: Leveraging influence functions for coreset selection in instruction finetuning of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10324–10335.

Hoang Anh Just, Feiyang Kang, Tianhao Wang, Yi Zeng, Myeongseob Ko, Ming Jin, and Ruoxi Jia. 2023. Lava: Data valuation without pre-specified learning algorithms. In *The Eleventh International Conference on Learning Representations*. OpenReview.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

Max Kuhn, Kjell Johnson, et al. 2013. *Applied predictive modeling*, volume 26. Springer.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. 2024. Datainf: Efficiently estimating data influence in loRA-tuned LLMs and diffusion models. In *The Twelfth International Conference on Learning Representations*.

Yongchan Kwon and James Zou. 2021. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. *arXiv preprint arXiv:2110.14049*.

Yongchan Kwon and James Zou. 2023. Data-oob: Out-of-bag estimate as a simple and efficient data value. In *International conference on machine learning*, pages 18135–18152. PMLR.

Huawei Lin, Yingjie Lao, and Weijie Zhao. 2024a. Dmin: Scalable training data influence estimation for diffusion models. *arXiv preprint arXiv:2412.08637*.

Huawei Lin, Jikai Long, Zhaozhuo Xu, and Weijie Zhao. 2024b. Token-wise influential training data retrieval for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 841–860, Bangkok, Thailand. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, and et al. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

R Douglas Martin and Victor J Yohai. 1986. Influence functionals for time series. *The annals of Statistics*, pages 781–818.

Bruno Kacper Mlodozeniec, Runa Eschenhagen, Juhan Bae, Alexander Immer, David Krueger, and Richard E. Turner. 2025. Influence functions for scalable data attribution in diffusion models. In *The Thirteenth International Conference on Learning Representations*.

Yurii Nesterov et al. 2018. *Lectures on convex optimization*, volume 137. Springer.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022a. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022b. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2022. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *arXiv preprint arXiv:2208.12242*.

Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. 2022. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8179–8186.

Rachael Hwee Ling Sim, Xinyi Xu, and Bryan Kian Hsiang Low. 2022. Data valuation in machine learning:" ingredients", strategies, and open challenges. In *IJCAI*, pages 5607–5614.

Richard Socher, Alex Perelygin, Jean Wu, and et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

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

Lloyd N Trefethen and David Bau. 2022. *Numerical linear algebra*. SIAM.

Alex Wang, Amanpreet Singh, Julian Michael, and et al. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the EMNLP Workshop on BlackboxNLP*.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

Tim Wibiral, Mohamed Karim Belaid, Maximilian Rabus, and Ansgar Scherp. 2024. Lossval: Efficient data valuation for neural networks. *arXiv preprint arXiv:2412.04158*.

Xinyi Xu, Zhaoxuan Wu, Chuan Sheng Foo, and Bryan Kian Hsiang Low. 2021. Validation free and replication robust volume-based data valuation. *Advances in Neural Information Processing Systems*, 34:10837–10848.

Xinyu Zhou, Simin Fan, and Martin Jaggi. 2024. Hyperinf: Unleashing the hyperpower of the schulz's method for data influence estimation. *arXiv preprint arXiv:2410.05090*.

A Experimental Details

Datasets The five GLUE benchmark datasets for Mislabeled Data Detection are available at HuggingFace Datasets library, and we use the training split for fine-tuning the model and validation split for computing the prediction loss. Following DataInf, we randomly choose 4500 and 500 samples from the original training and validation splits respectively for QNLI, QQP and SST-2. The SVAMP dataset for text generation task consists of 4 different types of math word problems, with a total of 700 training data and 300 test data. The number of sample in each type varies, we take 75 (resp., 25) examples from each type as the training (resp., test) set. The text-to-image style generation dataset contains three different styles of images. Each style has 200 training image-text pairs and 150 validation image-text pairs, and the text prompt follows the structure: "Generate an image in a specific {custom} style. {A text sequence of the original dataset which describes an image}", where {custom} is substituted with either "cartoon", "pixelart", or "black and white line sketch". The DreamBooth dataset for subject generation includes 30 different subjects. Each subject has 4 to 6 examples, and 3 data points from each subject are used for training with the remaining as validation. We add a unique random string to each subject in the prompt. For example, we use "a AXNkV dog" and "a DxE3K dog" to differentiate two different dogs.

Fine-tuning Setup We use Low-Rank Adaptation (LoRA) for fine-tuning. For RoBERTa model, the training is performed for 10 epochs with a batch size of 32 and a learning rate of 3×10^{-4}. The LoRA rank is $r = 4$ and α is set to be r. For Llama-2-13B-chat and Llama-3-8B, we use a learning rate of 3×10^{-4}, LoRA hyperparameters $r = 8$ and $\alpha = 32$, in 4-bit quantization, with a batch size of 64 across 10 training epochs. We fine-tune the stable-diffusion-v1.5 model with a batch size of 4 for 10000 training steps, using a learning rate of 1×10^{-4} and LoRA hyper-parameters $r = 8$ and $\alpha = 8$.

Influence Estimation We report the hyperparameters for RRInf in Table 5. For LLMs including RoBERTa, Llama-2-13B-chat and Llama-3-8B, we sample a single neuron at each iteration. For stable-diffusion-v1.5 model, we sample an entire layer from the total set of layers. The learning rate is set to 0.01 across all tasks.

B Additional Experimental Results

Mislabeled Data Detection We compare the time efficiency of different influence estimation methods on Mislabeled Data Detection task in Section 4. The average computation time (in seconds) along with the standard deviation is presented in Table 4. While Hessian-free is the fastest in all datasets (1.19 seconds to 2.42 seconds), it trades off accuracy. RRInf achieves identical (slightly better) runtime to DataInf and is significantly faster than LiSSA which is the slowest and takes 45–57

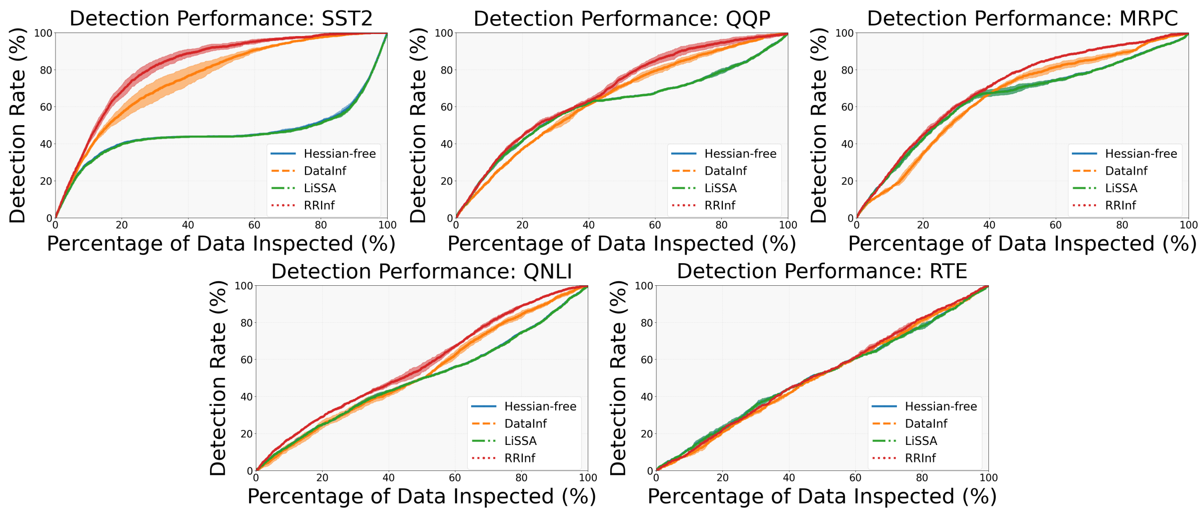

Figure 2: Detection rate and data inspected across *sst2*, *qqp*, *mrpc*, *qnli*, and *rte* (RRInf is shown in red. Hessian-free is shown in blue. DataInf is shown in orange. LiSSA is shown in green).

Method	**RTE**	**QNLI**	**QQP**	**MRPC**	**SST2**
Hessian-free	$48.3\% \pm 2.8\%$	$68.4\% \pm 4.2\%$	$71.3\% \pm 1.7\%$	$71.9\% \pm 1.3\%$	$82.0\% \pm 4.2\%$
DataInf	$48.8\% \pm 4.4\%$	$73.4\% \pm 4.4\%$	$75.0\% \pm 4.4\%$	$73.0\% \pm 0.1\%$	$91.1\% \pm 0.2\%$
LiSSA	$49.1\% \pm 3.4\%$	$67.4\% \pm 4.0\%$	$69.9\% \pm 2.0\%$	$71.4\% \pm 1.0\%$	$81.9\% \pm 6.9\%$
HyperINF	$49.1\% \pm 4.6\%$	$76.7\% \pm 3.4\%$	$77.2\% \pm 2.3\%$	$72.7\% \pm 1.0\%$	$92.8\% \pm 0.4\%$
RRInf	$56.1\% \pm 2.8\%$	$81.6\% \pm 2.8\%$	$77.7\% \pm 1.4\%$	$75.9\% \pm 0.9\%$	$93.5\% \pm 0.8\%$

Table 3: Accuracy Comparison on Data Selection Task (%).

Dataset	**Hessian-free**	**DataInf**	**LiSSA**	**RRInf**
SST-2	2.1406 ± 0.2955	10.7444 ± 0.4890	57.0686 ± 1.4048	10.2276 ± 0.0064
QQP	2.0230 ± 0.0360	10.9016 ± 0.5941	57.8501 ± 0.8990	10.2285 ± 0.0086
QNLI	2.4201 ± 0.8299	10.7237 ± 0.4375	56.7970 ± 1.9614	10.2294 ± 0.0073
MRPC	1.7183 ± 0.1267	8.6888 ± 0.8272	45.5788 ± 1.0792	8.4108 ± 0.0093
RTE	1.1905 ± 0.1450	5.6884 ± 0.2116	30.6206 ± 0.7600	6.2483 ± 0.9192

Table 4: Comparison of Algorithmic Computation Time (Run Time in Seconds) on Mislabeled Data Detection Task

Task	**Batch size**	**#Iter**	**LR**
MRPC/RTE/QNLI/QQP/SST2	neuron	1000	0.01
LLAMA 2 Text Generation	neuron	1000	0.01
LLAMA 3 Text Generation	neuron	1000	0.01
SD Style Generation	layer	2000	0.01
SD Subject Generation	layer	2000	0.01

Table 5: Hyperparameter Settings for RRInf.

LoRA Rank	**AUC (Mean ± Std)**	**Recall (Mean ± Std)**
2	$62.0\% \pm 9.6\%$	$46.9\% \pm 8.8\%$
8	$64.3\% \pm 11.1\%$	$50.6\% \pm 9.8\%$
12	$62.9\% \pm 10.5\%$	$49.4\% \pm 9.3\%$
24	$63.4\% \pm 10.5\%$	$50.3\% \pm 9.1\%$

Table 6: Performance Metrics of RRInf Across Ranks. AUC and Recall metrics in % (mean ± std) for different ranks. The rank $r = 8$ is used in the main results.

seconds depending on the dataset. This aligns with our analysis in the paper that the complexity of our normalized stochastic gradient algorithm does not depend on the model size, making it particularly well-suited for large scale models. We also report results on the Mislabeled Data Detection task, with a different evaluation metric which measures the fraction of mislabeled data detected (i.e., detection rate) when using influence values to pick data points to inspect. Figure 2 shows that RRInf identifies a larger fraction of mislabelled training data (y-axis) regardless of the fraction of the training data set that is examined (x-axis), demonstrating its superior ability to detect harmful data points across all benchmark datasets.

Style Generation Task Additional experiments on the style generation task using varying ranks $r = 2, 8, 12, 24$ are presented in Table 6, illustrating the impact of LoRA rank on the final performance of RRInf. The results suggest that while the rank can influence performance to some degree, it does not lead to any significant differences in overall performance.

We provide the additional results on Style Generation Task used in the Ablation Studies. Table 7 shows that the full gradient variant $\text{RRInf}_{\text{full}}$ achieves slightly higher AUC and Recall than

Metric / RRInf$_{\text{full}}$	**Mean ± Std**
AUC	$65.8\% \pm 13.3\%$
Recall	$51.9\% \pm 11.8\%$
Metric / RRInf	**Mean ± Std**
AUC	$64.3\% \pm 11.1\%$
Recall	$50.6\% \pm 9.8\%$

Table 7: Performance comparison between RRInf variants using full gradient descent (RRInf$_{\text{full}}$) and layer-wise sampling (RRInf). Reported results are mean ± standard deviation.

Batch Size	**AUC**	**Recall**
1	$60.3\% \pm 8.5\%$	$45.0\% \pm 8.0\%$
8	$62.5\% \pm 9.9\%$	$48.7\% \pm 8.9\%$
16	$62.8\% \pm 10.4\%$	$49.1\% \pm 9.1\%$
32	$63.1\% \pm 10.5\%$	$49.7\% \pm 9.3\%$
64	$63.4\% \pm 10.6\%$	$50.1\% \pm 9.2\%$
128	$63.4\% \pm 10.7\%$	$50.2\% \pm 9.4\%$
256	$63.4\% \pm 10.7\%$	$50.3\% \pm 9.3\%$
512	$63.4\% \pm 10.7\%$	$50.4\% \pm 9.3\%$
1024	$63.4\% \pm 10.7\%$	$50.3\% \pm 9.3\%$
2048	$63.4\% \pm 10.7\%$	$50.3\% \pm 9.3\%$

Table 8: Performance of RRInf$_{\text{mb}}$ by varying Sampling Batch Size (Mean $\pm$ Std).

RRInf which samples a layer at each iteration. Table 8 report the performance of RRInf$_{\text{mb}}$ which randomly samples a subset of neurons. The results show that RRInf$_{\text{mb}}$ is slightly inferior to RRInf, suggesting that a layer-level sampling may be more suitable than neuron-level sampling in diffusion models.

C Data Selection Task

We add experiments for Data Selection Task on the same noisy GLUE datasets used in Mislabeled Data Detection. The experimental setting is the same as the one in Table 1. After computing the influence function, the top 70% most beneficial data points are selected. We then retrain a model from scratch with the selected subset and evaluate the model's classification accuracy on the holdout test dataset. In addition to DataInf, LiSSA and Hessian-free, we include a recent baseline method HyperINF (Zhou et al., 2024). As shown in Table 3, RRInf consistently outperforms all baseline influence estimation methods and achieves the best accuracy across all five datasets, demonstrating the effectiveness of our method in data selection tasks.