Differentially Private Learning with Noisy Labels

by

Shubhanakar Mohapatra

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2020

Abstract

Supervised machine learning tasks require large labelled datasets. However, obtaining such datasets is a difficult task and often leads to noisy labels due to human errors or adversarial perturbation. Recent studies have shown multiple methods to tackle this problem in the non-private scenario yet this remains an unsolved problem when the dataset is private. In this work, we aim to train a model on a sensitive dataset that contains noisy labels such that (i) the model has high test accuracy and (ii) the training process satisfies (ϵ, δ)-differential privacy. Noisy labels, as studied in our work, are generated by flipping labels in the training set, from the true source label(s) to other target(s). Our approach, DIFFINDO, constructs a differentially private stochastic gradient descent algorithm which removes suspicious points based on their noisy gradients. We show experiments on datasets across multiple domains with different class balance properties. Our results show that the proposed algorithm can remove up to 100% of the points with noisy labels in the private scenario while restoring the precision of the targeted label and testing accuracy to its no-noise counterparts.

Acknowledgements

First of all, I would like to thank my supervisors for the immense support and time that they have given me throughout my research. Thank you, Prof. Helen Chen, for believing in me and for allowing me to be a research student with you. Thank you, Prof. Xi He, to introduce and help me learn this fascinating field of study and correct me whenever I went wrong. Both of your advice and support has led me to write this thesis today. I am incredibly fortunate to have been supervised by you and am excited about all our future endeavours. Thank you Prof. Florian Kerschbaum and Prof. Gautam Kamath for reading my thesis and for your comments on this work.

A huge thanks goes to Dr. Om Thakkar and Prof. Gautam Kamath(again) for guiding and collaborating towards the research for my thesis. I would also like to thank all the amazing researchers at the Data Systems group and WHISTL group at the University of Waterloo for all the lively and sometimes intricate discussions leading to the numerous ideas together. I'm fortunate to have met and have the chance to work with talented and hard-working researchers like you.

Thanks a million to all my amazing friends at UWaterloo who have spent hours with me at board games, badminton, Friday nights together at Uptown, planning get-togethers or have been to infinitely many coffee breaks and meanwhile always putting up with my atrocities and supporting me through the thick and thin. I also can't go without thanking my friends back at home in India with whom I've had long video chats talking about life and end up absolutely going nowhere. I wish with all my heart that all our dreams come true.

Last but not least, I would like to thank my parents, Lilima Mohapatra and Santosh Mohapatra and my brother Soumyankar for the never-ending love and support. I would not be half the person that I am today if it wouldn't be for you people. Your phone calls every morning motivated me to go through all the tough times. Love you Mamma, Papa and Chunnu!

Dedication

To my best friend, Rishav.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The last decade has observed immense growth in deep learning. With the popularization of deep learning, the need for clean datasets has become of utmost importance, as the accuracy of the model is highly dependent on the quality of the training data [57, 23]. One common data quality issue is noisy labels, which could arise due to several issues, including data entry errors [6], or data poisoning attacks [73], in which attackers inject fake points (e.g., creating fake accounts). If the model curator has direct access to the data as well as domain expertise, these noisy labels can be manually corrected or removed. However, cleaning these noisy labels is usually difficult, as many noisy datasets (e.g., medical data) are sensitive and private. One may be tempted to argue that the model curator could remove datapoints with manipulated labels from the training set by simply viewing the training point and manually adjusting the label. This is reasonable if the training points are interpretable, but is an impossible task if labelling requires significant domain expertise, or if there are a large number of outliers. The labelling of the points becomes an even harder task if the data is private and can not be directly accessed by the model curator. Motivated by these challenges, we investigate supervised learning with noisy labels in a private setting. There are numerous works in the literature focused on identifying and dealing with noisy labels in the non-private setting, but to the best of our knowledge, our paper is the first of its kind in the private setting.

Prior work in the non-private setting [60, 57, 61, 39] have tried to learn with noisy labels using three methods. The first approach is to design models which are inherently robust to model flips. Researchers have achieved such models by building robust loss functions, adding extra layers in a neural network to remove noisy points or using some auxiallry knowledge. A second approach is to relabel the points using unsupervised clustering techniques. The idea is to cluster the points without using the labels (e.g., k-nearest neighbours [61] or using validation from a separate set of trusted labels [28]), and then use the cluster centres to relabel points which are a minority in each cluster. The third approach instead is to identify and remove the label-flipped points from the training set [37, 13]. Diakonikolas, Kamath, Kane, Li, Steinhardt, and Stewart present a meta-algorithm called SEVER [13] which is used on top of any stochastic optimization technique to remove outlier points based on scores calculated from the top most singular vector of variance of the gradients. They show experiments on traditional machine learning algorithms like SVM and logistic regression. However, none of these works consider private settings. In our work, we show that SEVER can be extended to neural networks and performs well on common machine learning datasets in the private scenario.

For quantifying our privacy guarantees, we consider the standard privacy notion, differential privacy (DP) [16]. It is the current gold standard of privacy and is being used by the US Census Bureau [50] and by many leading software industries like Google, Uber, Facebook and Apple [21, 80]. To make our learning model private, we use differentially private stochastic gradient descent (DPSGD) [83, 72, 5, 1], which can be an optimizer for any deep learning model. DPSGD is based on a stochastic optimizer which iteratively updates the parameters in the model to optimize the objective function. At each iteration, the empirical gradients are computed from a batch of sampled points and noise is added to the (clipped) gradients to ensure differential privacy. Applying the meta-algorithm SEVER to remove points in DPSGD is non-trivial, as there are several new issues that arise under the additional constraint of privacy. We address these issues and propose a new algorithm which is both private and robust.

The main contributions of our work are as follows :

- We are the first study to show that the accuracy of a differentially private machine learning algorithm is prone to noisy labels in the training data.
- We provide a novel differentially private algorithm for this setting of learning on noisy labels, named DIFFINDO.
- Our results show that DIFFINDO can remove an up to a 100% of the outliers and increase testing accuracy for upto the no-noise accuracy for baseline algorithms, including private unsupervised learning and DPSGD.

This thesis is organized as follows. We start by first looking into some prior related work specifically in DP machine learning and learning with noisy labels in the non-private scenario in Chapter 2. Next in Chapter 3, we describe some preliminary notations and fundamentals which would later help us in sketching out our algorithm and the problem setup. As this problem has not been defined before in literature, we formally write down the problem setup and statement in Chapter 4. In Chapter 5, we introduce DIFFINDO which is a method to remove outliers while training on private data. We extensively evaluate the proposed algorithm on multiple inputs across different datasets, models and label flips in Chapter 6. Finally, in Chapter 7, we conclude this thesis while discussing some open questions and future work in this field.

Chapter 2

Related Work

In this section, we look into the related work. We first start with differentially private machine learning, then move to literature work where researchers have presented machine learning algorithms to learn with noisy labels. Finally, we present background in health data where differentially private learning with noisy labels can be used.

2.1 Differentially Private Machine Learning

Simply anonymizing the dataset and scrapping off demographic and personal information does not provide privacy. This has been proven by multiple researchers in history. For example, in 2006, Narayanan and Shmatikov [55] showed that the adversary is able to re-identify the members of the Netflix dataset, which consisted of anonymized individuals and their choice of movies by extrapolating rankings and timestamps in IMDB public data repository. The same phenomenon was observed in other kinds of data like public health records [75], social network graphs [4], search query logs [31] and many more. In 2009, Wang et al. showed that releasing computed statistics from high-dimensional sensitive genetic data can also face the same fate of de-anonymization [82].

Similar data leakage trends have also been observed while training a machine learning model. There has been much previous work that has demonstrated this by showing different attack techniques to machine learning models. One such attack is a *reconstruction attack*, where the adversary tries to decode the training dataset by asking the model multiple statistical queries and getting them answered with certain accuracies [14, 22]. Another line of attacks is called the *membership inference attack* where the adversary tries to guess whether a particular data point was used in the training of the machine learning model or not [29, 70, 7, 19, 71]. These attacks can even be made worse if the adversary has white box access to the machine learning model [56]. The third type of threat is focused on *memorization attacks*, where it has been seen that a neural network has tendencies of memorizing sensitive information from training dataset [8].

These attacks led to research that tried to bound the training data information fed into the model by providing differential privacy guarantees. The US Census Bureau was the first big organization to adopt differential privacy in 2008 for a product called OnTheMap [50] and subsequently Google, Apple, Microsoft, Facebook and Uber followed [80]. Google used differential privacy to monitor and analyze the Chrome browser properties of its user base to detect security vulnerabilities [21]. Although some of the underlying differential privacy mechanisms such Laplace mechanism [17] and Gaussian Mechanism [15] can be used to compose into iterative machine learning tasks but it has been seen that they provide loose guarantees and lower utility. In recent years, differentially private machine learning has been a focus of many researchers and it has led to much work in this field. Differentially private empirical risk minimization using *input*, *output* and *objective* perturbation was first proposed by Choudhuri et al. in 2011 [9]. Later, a private stochastic gradient descent algorithm was proposed by Song et al. [72] and optimal risk bounds were provided later by Bassily et al. [5]. There have also been some works in the non-convex optimization setting including deep learning by Abadi et al. [1]. A broad and more in-depth analysis of DP machine learning algorithms can be found here [76]. Our problem aims to learn with the same differential private guarantees as prior work, but the prior work only considers clean data for training, but we use noisy (more specifically, label flipped) datasets for training.

2.2 Learning with Noisy Labels

Machine learning is highly dependent on the quality of data, and that is the reason why ML practitioners spend most of their time cleaning and preprocessing the data to get the best out of the models [23, 57]. Some studies have also shown that label noise can even be more impactful on the training process than feature noise itself [86, 65]. Noisy labels in the training data is a common problem in the medical field because obtaining clean and accurate medical data is a challenging task [43, 35]. Additionally, it is even a harder task to label this kind of data as expert domain expertise or consensus from a group of experienced doctors is required [62]. Getting hold of medical data becomes a herculean task when these problems are combined with stringent health and government standards and policies. Some studies have been able to successfully gather health data by employing a large number of experts to annotate the datasets, but such efforts require massive financial and logistic resources [25]. Alternatively, a few studies have successfully used automated mining of medical image databases such as hospital picture archiving and communication systems (PACS) to build large training datasets [84]. However, this method is not always applicable, as historical data may not include all the desired labels or images. To mitigate the above issues, some studies have tried to collect data by crowd-sourcing them to get labels from non-experts. However, even for relatively simple segmentation tasks, computerized systems have been shown to generate significantly less accurate labels compared with human experts and crowd-sourced non-experts [27]. Given this scenario, it is becoming hard for deep learning practitioners to get hold noise-free datasets [10]. Hence, algorithmic solutions to get rid of noisy labels are highly desired in the medical field. In this section, we will discuss some prior work in learning with noisy labels with a focus on deep learning.

Researchers have tried to tackle noisy labels in multiple ways which can be broadly classified into the three following categories :

1. *Design based methods* - These methods aim at devising models that can inherently be robust to label noise. This can be achieved by altering the model, the loss function or the training procedure. Ghosh et al. studied the conditions for the robustness of a loss function to label noise for training deep learning models [24]. Two ways of improving the robustness of a loss function of deep learning models were proposed in Patrini et al. [60]. Several studies have proposed adding a “noise layer" to the end of deep learning models. The noise layer proposed by [74] is equivalent to multiplication with the transition matrix between noisy and true labels. The authors developed methods for learning this matrix in parallel with the network weights using error back-propagation. A similar noise layer was proposed by [77] for training a generative adversarial network (GAN) under label noise. Some papers have also tried a distillation approach based on the auxiliary knowledge learnt from some clean training data. The training starts by assigning a pseudo-label based on the convex combination of auxiliary knowledge and noisy label. They suggested that as the training proceeds, the model becomes more accurate, and its predictions can be weighed more strongly, thereby gradually forgetting the original incorrect labels.

2. *Clustering based methods* - Usually, the noisy labels are dominated by the correct labels, and there is a considerable correlation between the correct labels which outcast the noisy labels out. Authors have found high correlations in the latent space generated after some layers of a network. These correlations have shown clusters of the noisy labelled datapoints [44]. Another example is the work of [85], where the

authors proposed a method to leverage the multiplicity of data samples with the same (noisy) label in each training batch. All samples with the same label were fed into a light-weight neural network model that assigned a confidence weight to each sample based on the probability of it having the correct label. Some authors have tried *data re-weighting* techniques to down weight the datapoints, which have a high probability of having noisy labels [66]. A clean validation set is used to determine the weights assigned to the training data with noisy labels. The authors showed that this weighting scheme was equivalent to assigning larger weights to training data samples that were similar to the clean validation data in terms of both the learned features and optimization gradient directions.

3. *Filtering methods* - The methods in this bracket either try removing the points with the noisy labels or try de-noising them back to its original label. The filtering process can be applied before the training process as a pre-processing step or whilst training. An example of this method is [79], where authors propose supervised and unsupervised methods to obtain the correctly labelled images from a collection of images. CleanNet, proposed by [45], extracts a feature vector from a query image with a noisy label and compares it with a feature vector that is representative of its class. The representative feature vector for each class is computed from a small clean dataset. The similarity between these feature vectors is used to decide whether the label is correct. In [47], a teacher-student based learning is proposed in which the student learns on a noisy dataset and the teacher who has some prior knowledge on a clean dataset helps the student in learning. The student model was encouraged to be consistent with the teacher model using a meta-objective in the form of the KL divergence between prediction probabilities. A similar area of research is the robust mean estimation problem, which is applied to more classical machine learning tasks that go back to Tukey and his students [78]. Recently, there has been much work in this direction for mean and co-variance estimation [12, 40, 63] leading to a meta-algorithm called SEVER which can be used on top of the gradient descent algorithm to remove data point with noisy labels(outliers) from the training dataset [13].

Here we have stated some of the previous work in this area and more related work can be found in [35]. Despite the numerous work in treating datasets with noisy labels for learning, there has been no advancement for doing the same in the private setting up to the best of our knowledge. This problem is well observed with medical data where access to health care data is plagued by vulnerability due to patient privacy considerations, which are protected by federal and local laws of protected health information such as the Health Insurance Portability and Accountability Act of 1996 (HIPAA). The fear of litigation and breach of privacy discourages providers from sharing patient health data, even when they are de-identified [2]. Noisy labels and privacy issues have made medical data an important use case for differentially private learning with noisy labels.

In this paper, we build on the Sever algorithm and use it in the private setting. One might be tempted to privatize any of the above algorithms, but the reason why we choose Sever as our best candidate is threefold. Firstly, it can be directly applied to the gradients, and as DPSGD already sanitizes the gradients, Sever is a comfortable fit. Secondly, Sever has previously shown not only to remove label flipped points but also adversarially generated points. We study further on adversarial data poisoning for our proposed algorithm in Appendix A. Thirdly, Sever provides theoretical guarantees for outlier point removal in higher dimensions while minimizing the loss function in the non-private scenario. This is beneficial while learning modern machine learning image datasets. We leverage these perks in our proposed algorithm.

Chapter 3

Preliminaries

In this section, we recall the definition of basic notions in machine learning, differential privacy, differentially private SGD, noisy labels, and the algorithm Sever, a non-private algorithm to combat data poisoning which our private method is based on.

3.1 Machine Learning

A supervised learning task takes in a set of n training points of the form $(x_1, y_1), .., (x_n, y_n)$, where $x_i \in \mathcal{X}$ is the feature vector of the ith point and $y_i \in \mathcal{Y}$ is its label (or class). The distribution of the random variables $(\mathcal{X}, \mathcal{Y}) \in \mathcal{X} \times \{1, 2, \ldots, C\}$ is defined as $\mathcal{D}$ where the feature space is $\mathcal{X} \subseteq \mathcal{R}^d$ and C is the number of classes. The goal is to train a parameterized function that takes in a feature vector $x \in \mathcal{X}$ and outputs a label $y \in \mathcal{Y}$ to fit the training data with respect to a loss function.

A loss function on parameters θ measures the difference between the labels predicted by θ-based function and the true labels in the training data. For example, the loss $\mathcal{L}(\theta)$ on θ can be the average of the loss over the training points, i.e., $\mathcal{L}(\theta) = \frac{1}{n} \sum_i \mathcal{L}(\theta, x_i)$. The learning task aims to minimize this loss function, but the loss function in complex models is usually non-convex and thus is difficult to minimize. Some common loss functions are the MSE (Mean Square Error) [46] which is mostly used for regression problems, Hinge Loss (used for SVMs) [69] and Cross Entropy loss [42] which is used for binary or multi class classification problems.

An optimizer like stochastic gradient descent (SGD) [67] is commonly used to reduce the loss function by iteratively updating the parameters θ in the negative gradient direction. At each iteration, t, a batch of points B from the training data is sampled and the gradient of the loss function g_t is estimated using these samples B. The model parameters are updated as $\theta_{t+1} = \theta_t - \eta \cdot g_t$, where η is the learning rate. SGD is ubiquitous in practice and seems to be effective at minimizing loss functions even in non-convex settings.

In our experiments, we use logistic regression and two common neural network architectures, a multi-layer perceptron and a convolutional neural network. Our logistic regression model is used on binary classification task and hence we use the *Binary Cross Entropy loss* and the neural networks are built for multi-label classification task and hence we use a variation of the cross entropy loss called the *negative log-likelihood loss*. However, the proposed algorithm in this thesis is not restricted to the aforementioned models and can be extended to any other models for supervised classification tasks.

3.2 Differential Privacy

Differential privacy (DP) [16, 17] is the gold standard privacy guarantee for learning patterns in the datasets to protect sensitive information of individuals. Formally, it is defined as follows.

Definition 1 (Differential Privacy)**.** *A randomized algorithm $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy (DP) if for any two adjacent inputs $D, D' \in \mathcal{D}$ that differ in an entry and for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ it holds that :*

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

The original implementation of Differential Privacy does not include the additive term δ. We use the variant used in [15], which allows for the possibility that the vanilla ϵ-Differential Privacy can be broken with probability δ (which is ideally set to $<< \frac{1}{|\mathcal{D}|}$).

Differential Privacy has many advantages like *post-processing* and *adaptive composition* which we define next.

Lemma 3.2.1 (Post-Processing [16])**.** *If a mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ is (ϵ, δ)-differentially private, then for any function $f : \mathcal{R} \to \mathcal{R}'$, we have that $f \circ \mathcal{M}$ is also (ϵ, δ)-differentially private.*

Lemma 3.2.2 (Basic Composition [16])**.** *For each $i \in [k]$, let algorithm $\mathcal{M}_i : \mathcal{D} \times \prod_{j=1}^{i-1} \mathcal{R}_j \to$*

$\mathcal{R}_i$ be (ϵ_i, δ_i)-DP in its first argument. If algorithm $\mathcal{M}_{[k]} : \mathcal{D} \times \prod_{j=1}^{k} \mathcal{R}_j$ is defined such that :

$$\mathcal{M}_{[k]} = (\mathcal{M}_1(\mathcal{D}), \mathcal{M}_2(\mathcal{D}, \mathcal{M}_1(\mathcal{D})), \cdots, \mathcal{M}_k(\mathcal{D}, \mathcal{M}_1(\mathcal{D})), \cdots, \mathcal{M}_{k-1}(\mathcal{D}))$$

then $\mathcal{M}_{[k]}$ is $\left(\sum_{i=1}^{k} \epsilon_i, \sum_{i=1}^{k} \delta_i\right)$-differentially private.

Lemma 3.2.3 (Advanced Composition [34, 18])**.** *For each $i \in [k]$, let algorithm $\mathcal{M}_i : \mathcal{D} \times \prod_{j=1}^{i-1} \mathcal{R}_j \to \mathcal{R}_i$ be (ϵ_i, δ_i)-DP in its first argument. If algorithm $\mathcal{M}_{[k]} : \mathcal{D} \times \prod_{j=1}^{k} \mathcal{R}_j$ is defined such that :*

$$\mathcal{M}_{[k]} = (\mathcal{M}_1(\mathcal{D}), \mathcal{M}_2(\mathcal{D}, \mathcal{M}_1(\mathcal{D})), \cdots, \mathcal{M}_k(\mathcal{D}, \mathcal{M}_1(\mathcal{D})), \cdots, \mathcal{M}_{k-1}(\mathcal{D}))$$

then for every $\delta' > 0$, algorithm $\mathcal{M}_{[k]}$ is $(\epsilon', 1 - (1-\delta') \prod_{i=1}^{k} (1-\delta_i))$ - DP, where

$$\epsilon' = \min\left(\sum_{i=1}^{k} \epsilon_i, \sum_{i=1}^{k} \frac{\epsilon_i(e^{\epsilon_i} - 1)}{e^{\epsilon_i} + 1} + \sqrt{\min\left(\sum_{i=1}^{k} 2\epsilon_i^2 \log\left(e + \frac{\sqrt{\sum_{i=1}^{k} 2\epsilon_i^2}}{\delta'}\right), \sum_{i=1}^{k} 2\epsilon_i^2 \log\frac{1}{\delta'}\right)}\right)$$

One of the classic DP algorithms is the Gaussian mechanism. To define the Gaussian mechanism, we will first describe the global sensitivity of a function.

Definition 2 (L_2-sensitivity)**.** *A function $f : \mathcal{D} \to \mathcal{R}$ has L_2-sensitivity S_f if*

$$\max_{\substack{D, D' \in \mathcal{D} \\ s.t (D, D') neighbours}} ||f(D) - f(D')||_2 = S_f$$

Lemma 3.2.4 (Gaussian Mechanism [17])**.** *If a function $f : \mathcal{D} \to \mathcal{R}$ has L_2-sensitivity S_f, and a mechanism $\mathcal{M}$ has on input $D \in \mathcal{D}$ outputs $f(D) + b$, where $b \sim \mathcal{N}\Big(0, \sigma^2 I_{n \times n})$, then for $c^2 > 2\ln(1.25/\delta)$, the mechanism $\mathcal{M}$ with parameter $\sigma \geq cS_f/\epsilon$ is (ε, δ) -differentially private. Here, $\mathcal{N}(0, \sigma^2 I_{n\times n})$ denotes a vector of n i.i.d samples from the zero-mean Gaussian distribution having variance σ^2.*

The analysis of this mechanism can be done *post hoc* or during an iterative process to break if a certain threshold is met. Note that there are infinitely many (ϵ, δ) pairs satisfying this equation and to get a certain bound one of the values need to be fixed.

3.3 Differentially Private SGD

Complex DP algorithms can be built from the basic primitives based on post-processing and composition properties of DP. Usually, a differentially private additive noise mechanism consists of the following step: approximate the whole algorithm using the sequential composition of bounded-sensitivity functions; choosing appropriate noise parameters and finally performing the privacy analysis by a composition theorem to report the final privacy cost. One of such algorithms is the Differentially Private Stochastic Gradient Descent (DPSGD). It is an approach inspired by the vanilla SGD to control the influence of any data point during the training process. The gradients of SGD are the random variables to which the noise is added. However, as there is no apriori bound of this gradient, the sensitivity S_f is calculated by clipping the maximum ℓ_2-norm to a user-defined parameter called C. Algorithm 1 outlines our basic method for training a model with parameters θ by minimizing the empirical loss function $\mathcal{L}(\theta)$. At each step of the SGD, we compute the gradient $\nabla_\theta \mathcal{L}(\theta_t, x_i)$ for a random subset of examples, clip the ℓ_2 norm of each gradient to a maximum clipping factor of C, compute the average, add Gaussian noise with mean 0 and variance of bounded sensitivity C multiplied with the noise multiplier σ to protect privacy, and take a step in the opposite direction of this average noisy gradient. In the end, in addition to outputting the model, we will also need to compute the privacy loss of the mechanism based on the information maintained by the privacy accountant also known as the Moments Accountant.

We follow these steps while designing our algorithm in Chapter 5. We use the Moments Accountant described in [1] for privacy accounting. The details of the Moments Accountant is described below.

Moments Accountant

The moments accountant based on Rényi-DP [53] keeps track of a bound on the moments of the privacy loss random variable (In our case, gradients). It generalizes the standard approach of tracking (ϵ, δ) and using the strong composition theorem. While such an improvement was known previously for composing Gaussian mechanisms, it applies also for composing Gaussian mechanisms with random sampling and can provide a much tighter estimate of the privacy loss iterative processes like gradient descent. It is

Algorithm 1: DPSGD

```
Data: Training set $A : \{x_1, ...x_n\}$, Loss function $\mathcal{L}(\theta)$, Parameters: Learning rate
      $\eta$, Lot size $L$, Gradient norm bound $C$ Noise scale $\sigma$, Total number of
      iterations $T$
Result: Model with trained parameter $\theta$
1  Initialize model with $\theta_0$ randomly;
2  for $t \in [1, T]$ do
3      Sample a random subset $L_t \subseteq A$, by independently including each element of $A$
         with probability $L/n$ ;
4      for $i \in A_b$ do
5          Compute gradient $g_t(x_i) = \nabla_\theta \mathcal{L}(\theta_t, x_i)$;
6          Clip each gradient in $\ell_2$ norm to $C$: $\bar{g}_t(x_i) = g_t(x_i)/\max(1, \frac{\|g_t(x_i)\|_2}{C})$;
7      end
8      Add noise $\tilde{g}_t = \frac{1}{|L|}(\sum_i \bar{g}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 I))$;
9      Gradient descent $\theta_{t+1} = \theta_t - \eta \tilde{g}_t$;
10 end
11 return $\theta_T$ and compute the overall privacy cost $(\epsilon, \delta)$
```

difficult to use advanced composition for iterative mechanism like DPSGD as each application of an advanced composition theorem leads to a wide selection of possibilities for (ϵ, δ)-differentially private guarantees.

Privacy loss is a random variable dependent on the random noise added to algorithm 1. That a mechanism $\mathcal{M}$ is (ϵ, δ)-differentially private is equivalent to a certain tail bound on $\mathcal{M}$'s privacy loss random variable. While the tail bound is very useful information on a distribution, composing directly from it can result in quite loose bounds. Moments Accountant instead computes the log moments of the privacy loss random variable, which composes linearly. Then it uses the moments bound, together with the standard Markov inequality, to obtain the tail bound, that is the privacy loss in the sense of differential privacy. More specifically, for neighbouring databases $\mathcal{D}$ and $\mathcal{D}'$, a mechanism $\mathcal{M}$, auxiliary input aux, and an outcome $o \in \mathcal{R}$, define privacy loss as

$$c(o; \mathcal{M}, \mathsf{aux}, d, d') \triangleq \log \frac{Pr[\mathcal{M}(\mathsf{aux}, d) = 0]}{Pr[\mathcal{M}(\mathsf{aux}, d') = 0]} \tag{3.1}$$

A common design pattern, which is used extensively in the paper, is to update the state by sequentially applying differentially private mechanisms. This is an instance of adaptive composition, which we model by letting the auxiliary input of the k^{th} mechanism $\mathcal{M}_k$ be the output of all the previous mechanisms.

For a given mechanism $\mathcal{M}$, we define the λ^{th} moment $\alpha_{\mathcal{M}}(\lambda; \mathsf{aux}, d, d')$ as the log of *moment generating function* evaluated at the value of λ:

$$\alpha_{\mathcal{M}}(\lambda; \mathsf{aux}, d, d') \triangleq \log \mathbb{E}_{o \sim \mathcal{M}(\mathsf{aux}, D)}[e^{\lambda c(o; \mathcal{M}, \mathsf{aux}, d, d')}] \tag{3.2}$$

In order to prove privacy guarantees of a mechanism, it is useful to bound all possible $\alpha_{\mathcal{M}}(\lambda; \mathsf{aux}, d, d')$. We define

$$\alpha_{\mathcal{M}}(\lambda) \triangleq \max_{\mathsf{aux}, d, d'} \alpha_{\mathcal{M}}(\lambda; \mathsf{aux}, d, d')$$

where the maximum is taken over all the possible aux and all neighbouring datasets $\mathcal{D}$,$\mathcal{D}'$.

We state the properties of α that we use for Moments Accountant.

Theorem 3.3.1. *Let $\alpha_{\mathcal{M}}$ be defined as above. Then,*

1. ***[Composability]*** *Suppose that a mechanism $\mathcal{M}$ consists of a sequence of adaptive mechanisms $\mathcal{M}_1, \cdots, \mathcal{M}_k$ where $\mathcal{M}_i : \prod_{j=1}^{i-1} \mathcal{R}_j \times \mathcal{D} \to \mathcal{R}_i$. Then, for any λ*

$$\alpha_{\mathcal{M}}(\lambda) \leq \sum_{i=1}^{k} \alpha_{\mathcal{M}_i}(\lambda)$$

2. ***[Tail Bound]*** *For any $\epsilon > 0$, the mechanism $\mathcal{M}$ is (ϵ, δ)-differentially private for*

$$\delta = \min_{\lambda} e^{\alpha_{\mathcal{M}}(\lambda) - \lambda\epsilon}$$

The proof to the theorem is given in [1]. By Theorem 3.3.1 it suffices to compute, or bound, $\alpha_{\mathcal{M}}(\lambda)$ at each step and sum them to bound the moments of the mechanism overall. We can then use the tail bound to convert the moments bound to the (ϵ, δ)-differential privacy guarantee.

3.4 Noisy Labels

Training data collected from unreliable sources usually have noisy labels. This noise can range from being identically split into different target labels (non-adversarial and natural noise) to skewed label flips (adversarially perturbed noise). Deep neural networks have been found to be naturally robust to label flips when larger networks and batch sizes are used [68]. In their work, the dataset of size n is diluted by adding Δn points with flipped labels, where Δ is the noise level. This does not change the available number of original examples and the original distribution is preserved.

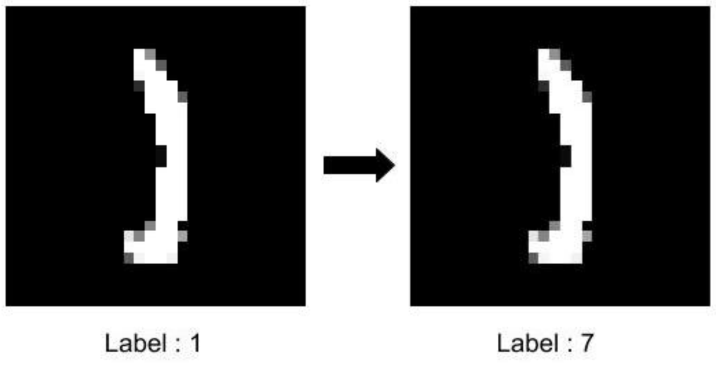

Figure 3.1: Example of a label flip from Class 1 to Class 7 in MNIST

In our work, we consider a stronger label flipping attack. Instead of adding noisy points, we use a *transition matrix* to change fraction of existing data points of one class to a target class [74]. We empirically show that this kind of attack not only degrades the test accuracy of the model but drastically reduces the precision of the target class. In other words, examples drawn from distribution $\mathcal{D}$ is unavailable and instead what we observe are noisy training samples $\{(x_1, \hat{y}_1), \ldots, (x_n, \hat{y}_n)\}$, where $x_i \in \mathcal{X}$ and $\hat{y}_i$ denotes the noisy labels. We call this new noisy distribution $\hat{\mathcal{D}}$ and it has the same feature and label space $(\mathcal{X}, \hat{\mathcal{Y}}) \in \mathcal{X} \times 1, 2, \ldots, C$.

Transition Matrix : The random variables $\hat{\mathcal{Y}}$ and $\mathcal{Y}$ are related through a *transition matrix* also known as a *corruption matrix* $T \in [0, 1]^{C \times C}$. Each element of this matrix $T_{i,j}$ stores the probability of elements from source class i to target class j.

$$T_{i,j} = p(\hat{y} = j | y = i)$$

Such matrices can help us in modelling single-targeted as well as composite label noises. When running our experiments, we implement this noisy flips by getting all the indices of the source class and randomly changing the labels of these source points to the target labels as given in the transition matrix. We also denote the total percentage of labelled flipped as Δ. In other words, this is the total sum of all the elements in the transition matrix.

3.5 Sever

Sever [13] is a meta-algorithm to filter outlier points based on their gradients in the training process. Outliers, as defined by Diakonikolas et al. can be mislabelled points defined just like the section above or adversarially added fake points to the training dataset. In their paper, they show that even a small fraction of these outliers in the training set can drastically affect the testing error of traditional machine learning algorithms like SVM and logistic regression. This algorithm takes in a batch of points, projects their gradients onto a carefully chosen direction, and removes a fraction of points with large projected values. Sever possesses strong theoretical guarantees and it has been shown to achieve positive results on corrupted datasets and traditional machine learning models like ridge regression and support vector machines. Unlike many other filtering based noisy label removal algorithms, Sever has no dependence on the underlying dimension of the training set d and hence, have better robustness in higher dimensions. Our work is the first time that Sever is being used for neural network models.

Now, we show one of the important theorems from Sever which is used to build the proposed algorithm in this thesis. We start by defining γ-approximate critical point.

Definition 3 (γ-approximate critical point)**.** *Given a function $f : \mathcal{D} \rightarrow \mathcal{R}$, γ-approximate critical point of f, is a point $w \in \mathcal{D}$ so that for all the unit vectors v where $w + \delta v \in \mathcal{D}$ for arbitrarily small positive δ, we have that $v \cdot \nabla f(w) \geq -\gamma$.*

Essentially, the above definition means that the value of f cannot be decreased much by changing the input w locally while staying within the domain. The condition enforces that moving in any direction v either cause us to leave $\mathcal{D}$ or causes f to decrease at a rate at most γ.

This definition helps us to define a γ-approximate learner.

Definition 4 (γ-approximate learner)**.** *A learning algorithm $\mathcal{L}$ is called γ-approximate if, for any functions $f_1, \cdots, f_n : \mathcal{D} \rightarrow \mathcal{R}$ each bounded below on a closed domain $\mathcal{D}$, the output $w = \mathcal{L}(f_{1:n})$ of $\mathcal{L}$ is a γ-approximate critical point of $f(x) \coloneqq \frac{1}{n}\sum_{i=1}^{n} f_i(x)$.*

In other words, $\mathcal{L}$ always finds an approximate critical point of the empirical learning objective. We note that most common learning algorithms (such as stochastic gradient descent) satisfy the γ-approximate learner property. We are now ready to present Theorem 2.1 from [13] which shows that the algorithm Sever always finds a γ-critical point with high probability.

Theorem 3.5.1. *Suppose that functions $f_1, \cdots, f_n : \mathcal{D} \rightarrow \mathcal{R}$ are bounded below on a closed domain $\mathcal{D}$, and suppose that they satisfy the following deterministic regularity conditions: There exists a set $I_{good} \subseteq [n]$ with $|I_{good}| \geq (1 - \Delta)n$ and $\sigma > 0$ such that:*

1. *$Cov_{I_{good}}[\nabla f_i(w)] \preceq \sigma^2 I, w \in \mathcal{D}$,*
2. *$||\nabla \hat{f}(w) - \bar{f}(w)||_2 \leq \sigma\sqrt{\Delta}, w \in \mathcal{D}$, where $\hat{f} \overset{def}{=} (1/|I_{good}) \sum_{i \in I_{good}} f_i$.*

Then the algorithm Sever *applied to $f_1, \cdots, f_n, \Delta$ returns a point $w \in \mathcal{D}$ that, with probability at least 9/10 is a $(\gamma + O(\sigma\sqrt{\Delta}))$- critical point of $\hat{f}$.*

Note that there is no dependence of the dimension d and hence enabling Sever to perform well on higher dimensional data (e.g. Image data). We built our proposed algorithm on top of this Theorem making it suitable in the private scenario.

Chapter 4

Problem Setup

To the best of our knowledge, there have been no previous work in private learning with noisy labels and therefore, in this chapter, we formally define the problem statement and setup.

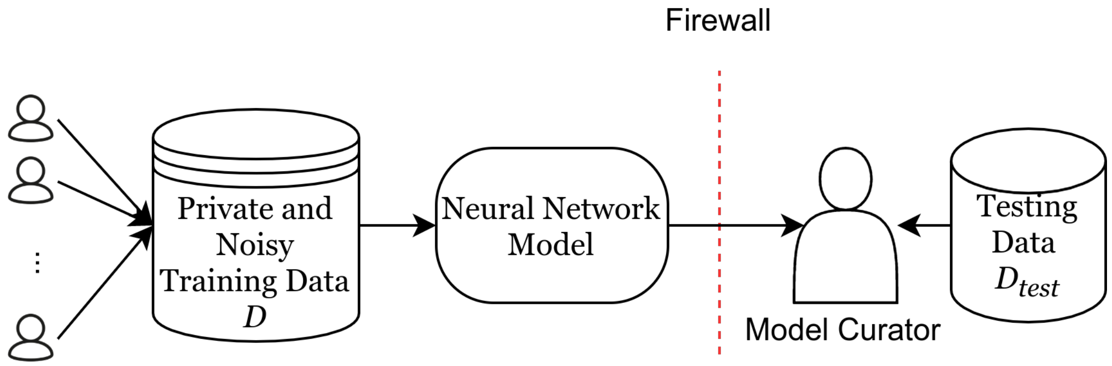

Figure 4.1: Problem Setup

Consider a set of sensitive and noisy training data D which is a set of n training points of the form $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i \in \mathcal{X}$ is the feature vector of the ith point and $y_i \in \mathcal{Y}$ is its label (or class) collected from users and sitting on a server behind a privacy firewall as shown in Figure 4.1. A model curator resides on the other side of the firewall and has no direct access to the dataset. However, a differentially private learning algorithm is allowed to run on the private dataset, and the learned model can be shared with the model curator. The curator is also not aware of any corruptions made to the data, but the curator has a set of clean testing data D_{test} with correct labels which can be used to test the final accuracy of the model. Also, note that the corruptions in the data (which in our case is generated by the transition matrix) can be targeted towards a specific class or set of classes. In such cases, we would also like to ensure high precision for the affected classes with the overall testing accuracy on D_{test}.

Problem Statement: Given a training dataset D with noisy labels, we would like to train a neural network model on D such that (i) the training process satisfies (ϵ, δ)-differential privacy, and (ii) the model predicts the labels of the points in the testing dataset D_{test} with high accuracy. In particular, if many training points are mislabeled to certain target class(es), the learned model may misclassify many testing examples to such class(es) resulting in low precision value(s). Hence, we would like to ensure high precision per class in the testing data.

Chapter 5

Our Approach

5.1 Overview

In this section, we outline the non-private method upon which our approach is based, highlight some new challenges which arise in our setting and summarize the modifications necessary to overcome these challenges.

Our approach is based on the non-private algorithm Sever [13], which can be briefly described as follows. Given some learning task, a model is first trained (ignoring any corruptions present in the data) to a local optimum. For instance, this could simply involve optimizing the parameters of the model (in order to minimize the loss function) using SGD, but any other appropriate optimization method could also be used as a black box. Given this optimum, the following "filtering" algorithm is run: we compute the empirical gradients of the training set, take the covariance matrix of these gradients, find the top eigenvector direction, and remove points whose gradient when projected onto this direction is too large. We repeat the entire procedure on the pruned dataset until convergence. An appropriate instantiation of this framework has strong provable guarantees, and [13] also demonstrates that it is effective in practice for simple models, including logistic regression and support vector machines (SVMs).

However, in the setting we are concerned with, there are a number of new considerations that arise. Most obvious is the constraint of differential privacy, but the *scale* of the datasets and models we investigate also have their own associated problems.

To elaborate on the last point, the largest dataset and model considered in [13] is the Enron dataset [51] trained on a SVM model, and thus the problem as a whole has roughly

4,000 training points and 5,000 parameters. In comparison, since one of our datasets and models is the MNIST dataset [41] on a multilayer perceptron, the problem has 60,000 training points and nearly 800,000 parameters – overall, several orders of magnitude larger! As it would be impractical to store data of this scale in memory (e.g., the covariance matrix alone would have $800000^2 \approx 7 \times 10^{11}$ entries), we are forced to turn to other options. Our method instead performs filtering using the gradients with respect to only the input pixels (rather than the parameters of the model), reducing the dimensionality from $800,000$ to roughly 800.

Turning to issues that arise due to privacy: in a differentially private setting, it is not cheap to train a model to optimality using DPSGD, as SEVER would prescribe. In particular, each privatized step of SGD will expend a fraction of our privacy budget, and we must be judicious with our choice of how many steps to take before running the filtering procedure.

More broadly speaking, there are many steps in the framework which must be privatized. The choice of how to split up our privacy budget (as well as the associated hyperparameters) gives rise to many important design decisions. In addition to the standard noise injection in DPSGD, we must also pay in our privatization of SEVER, as we must privately compute the covariance of the gradients (requiring another private mean computation as well). Also, while the version of SEVER used in experiments in [13] removes a percentile of points with the highest scores, it is not clear how to do this same removal privately, and thus we use a fixed threshold.

Finally, since our method involves removal of points from the dataset at various intervals during the process of DPSGD, a new privacy analysis is needed to account for the total privacy loss of the dataset. Since sensitive information of individual points is used in the removal step, it can be tricky to obtain utility along with a strong privacy guarantee. By carefully taking advantage of post-processing and adaptive composition, our method achieves this while ensuring that it doesn't worsen the privacy guarantees of the underlying DPSGD procedure as well, even though the effective dataset size can potentially reduce during the process. For further technical details, refer to section 5.3.

5.2 Diffindo

Algorithm 2: Diffindo

```
Data: Training set $\{x_1, ...x_n\}$, Loss function $\mathcal{L}(\theta)$, Parameters: Learning rate $\eta$,
      Lot size $L$, Gradient norm bound $C$, Average gradient norm bound $C_2$,
      Noise scale $\sigma$, Removal multiplier $p$, Total number of SGD iterations $T$,
      Total number of removal iterations $T_f$
Result: Model with trained parameter $\theta$
1  Initialize model with $\theta_0$;
2  Initialize all datapoints as active indices in an array of length $n$ as $A = [1, 2, ...n]$;
3  for $t \in [1, T]$ do
4      Sample a random set $A_b \subseteq A$, by independently including each element of $A$
        with probability $L/n$ ;
5      for $i \in A_b$ do
6          Compute gradient $g_t(x_i) = \nabla_\theta \mathcal{L}(\theta_t, x_i)$;
7          Clip each gradient in $\ell_2$ norm to $C$: $\bar{g}_t(x_i) = g_t(x_i)/\max(1, \frac{\|g_t(x_i)\|_2}{C})$;
8      end
9      Add noise $\tilde{g}_t = \frac{1}{|L|}(\sum_i \bar{g}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 I))$;
10     Gradient descent $\theta_{t+1} = \theta_t - \eta \tilde{g}_t$;
11     if $t \mod \lceil (T/T_f) \rceil = 0$ then
12         Compute gradient $G$ by passing whole active dataset with indices $\in A$:
            $G = \nabla_x \mathcal{L}(\theta_t, x_{i \in A})$;
13         for $i \in A$ do
14             Clip each gradient in $\ell_2$ norm to $C_2$: $\bar{G}_i = G_i/\max(1, \frac{\|G_i\|_2}{C_2})$;
15         end
16         Add noise: $\tilde{G}_{avg} = \frac{1}{n}(\sum_i \bar{G}_i + \mathcal{N}(0, \sigma^2 C_2^2 I))$;
17         $\widehat{A}$ = FILTER($G$,$\tilde{G}_{avg}$,$A$,$p$,$C_2$,$\sigma$,$p$);
18         update active indices $A = A \setminus \widehat{A}$;
19 end
20 return $\theta_T$ and compute the overall privacy cost $(\epsilon, \delta)$
```

In this section, we describe our proposed algorithm called Diffindo[1] [54]. Given a supervised learning task, Diffindo trains on a sensitive dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ privately. This algorithm considers noisy labels in the dataset and hence applies a filter

[1]The name is inspired from the severing charm incantation in the famous fantasy novel series Harry Potter by J. K. Rowling.

function during the training process to remove points that are likely mislabeled.

DIFFINDO is described in Algorithm 2. A privatized version of the core filtration step of SEVER is described in Algorithm 3, and is a key primitive in DIFFINDO. In the sequel, we describe the steps of DIFFINDO, highlighting some important design decisions along the way. Lines 3 through 10 of Algorithm 2 are almost identical to DPSGD: we subsample the

```
Algorithm 3: FILTER($G$,$G_{avg}$,$A$,$p$,$C_2$,$\sigma$,$p$)
  Data: Gradients $G$, Average batch gradients $G_{avg}$, indices A, Gradient norm
        bound $C_2$, Noise scale $\sigma$, Removal multiplier $p$
  Result: Bad indices $\widehat{A}$
1  Initialize $S \leftarrow \{1, \ldots, |G|\}$;
2  $\widehat{G} = [G - \tilde{G}_{avg}]$ be the $|G|$ x $d$ matrix of centered gradients;
3  Clip each gradient in $\ell_2$ norm to $C_2$ $\widehat{G} = \widehat{G} / \max(1, \frac{\|\widehat{G}\|_2}{C_2}) \forall i \in S$;
4  Let $E \in \mathcal{R}_{S*S}$ be a symmetric matrix where the upper triangle (including the
     diagonal) is i.i.d. samples from $\mathcal{N}(0, \sigma^2 C_2^4 I)$, and each lower triangle entry is
     copied from its upper triangle counterpart;
5  Covariance matrix $\tilde{\Sigma} = \widehat{G}^T \widehat{G} + E$;
6  Let $\tilde{v}$ be the top eigenvector of $\tilde{\Sigma}$;
7  Compute the vector of *outlier scores* $\tau_{i \in S} = ((\widehat{G}_i - \tilde{G}_{avg}) \cdot \tilde{v})^2$ ;
8  Compute threshold score $\tau_{thresh} = p \cdot (\tilde{G}_{avg} \cdot \tilde{v})^2$;
9  $\widehat{A}$ = Indices with scores more than $\tau_{thresh}$ ;
10 Return $\widehat{A}$ ;
```

dataset, compute and clip the gradients, and add noise. The main difference is that, instead of subsampling from the entire dataset, we have a set A of "active indices" from which we subsample. This set is initialized to be the entire set of points but will be pruned as points are removed by the filtering procedure. An important point to note is the scaling factor in Line 9, which is $1/L$. Ideally, we would scale this sum by the number of summands, $|A_b|$. Since using this value exactly as the normalization factor would lose some of the privacy gained due to subsampling, we could instead rescale by its expected value, $L|A|/n$. However, since exactly revealing the size of $|A|$ would also lose privacy, we simply divide by the fixed value L, which is accurate at the initial step, but can potentially be too large at later steps, acting as a decay in the learning rate. Decaying the learning over time has been demonstrated to increase accuracy in certain situations, though one could also consider gradually increasing the learning rate to counteract this effect.

Since there are T total iterations of this procedure, and we have T_f total removal iterations, we satisfy the condition of the if statement on Line 11 every T/T_f steps. This initiates the filtering algorithm, which is a privatized analog of [13]. The gradients of the entire (active) dataset are taken. It is important that we perform a "mega-batch" computation involving all the data (rather than just a mini-batch A_b) since this will result in the best possible concentration for the covariance matrix of the gradients (Line 5 in Algorithm 3), crucially with respect to the input rather than the model parameters (for memory reasons, as discussed above). We clip and (privately) compute the average gradient. Once again, in Line 16, note that we average by n rather than $|A|$ to save privacy budget. In Line 17, the privatized filtering algorithm (Algorithm 3) is actually invoked. We centre and clip the gradients as before, and then form a noised and private version of their covariance matrix [20]. We calculate scores for each point based on the magnitude of their projection onto the top eigenvector of the (privatized) covariance (Line 7). Finally, any point with a score greater than some prescribed threshold is marked for removal from the set of active indices A.

This process is repeated until a pre-set number of epochs by the moment accountant [1] or until the privacy budget is spent.

In DIFFINDO, we carefully choose where to apply SEVER filters. SEVER is usually run after the learning algorithm has already made one pass on the entire training set. If we apply SEVER on every batch of DPSGD, we may remove too many good points as some batches only have a few outliers. The same points also may appear multiple times in the same batch due to sampling with replacement in DPSGD. Hence, we keep the list of active points in our algorithm and run SEVER on a mega-batch of points. In the evaluation section, we also study the optimal fraction of points to be removed when applying SEVER.

5.3 Privacy Analysis

Theorem 5.3.1. *There exist constants c_1 and c_2 so that given the sampling probability $q = L/n$, the number of SGD steps T, and the number of removal steps T_f, for any $\epsilon < c_1(q^2T + T_f)$, DIFFINDO (Algorithm 2) is (ϵ, δ)-differentially private for any $\delta > 0$ if we choose*

$$\sigma \geq c_2 \frac{\sqrt{(q^2T + T_f)\log(1/\delta)}}{\epsilon}.$$

Proof. We view Algorithm 2 as a sequence of adaptive mechanisms $M_1, M_2, \ldots, M_k$, where $M_i : \prod_{j=1}^{i-1} \mathcal{R}_j \times \mathcal{D} \to \mathcal{R}_i$. There are three types of mechanisms involved: (i) computing noisy gradients ($\tilde{g}_t$) with respect to model weights for a batch in Line 16 of Algorithm 2, (ii) computing noisy gradients with respect to input features ($\tilde{G}_{avg}$) in Line 9 of Algorithm 2 , (iii) computing the top eigenvector of noisy covariance ($\tilde{\Sigma}$) in Algorithm 3. We denote these three types of mechanisms by M^g, M^G, M^c. In Algorithm 2, M^g runs T times, and both M^G and M^c run T_f times each.

We will analyze the privacy loss of this sequence of adaptive mechanisms similarly as the privacy analysis of DPSGD using the moments accountant [1]. First, for a given mechanism M, the λ^{th} log moment of M is defined as the maximum log of the moment generating function evaluated at λ for all possible adjacent database pairs D, D' and for all possible auxiliary input aux, i.e.,

$$\begin{aligned}\alpha_M(\lambda) &\triangleq \max_{aux,D,D'} \log \mathbb{E}_{o\sim M(aux,D)}[e^{\lambda \log \frac{M(aux,D)=o}{M(aux,D')=o}}] \\ &= \max_{aux,D,D'} \log \mathbb{E}_{o\sim M(aux,D)}[(\tfrac{M(aux,D)=o}{M(aux,D')=o})^{\lambda}].\end{aligned}$$

The log moments of the three types of mechanisms are shown next. The first type of mechanism has a log moment of $\alpha_{M^g}(\lambda) \leq \frac{q^2\lambda^2}{\sigma^2}$ for any positive integer $\lambda \leq \sigma^2 \ln(\frac{1}{q\sigma})$, if we select $q < \frac{1}{16\sigma}$ (Lemma 3 in DPSGD [1]). Note that Algorithm 2 samples from a set of active points instead of from the full dataset like DPSGD. For the privacy proof, we can consider an alternate algorithm that sets the gradients of the inactive points' gradients as zero and then samples them with the same fixed probability q as the active points. This results in the same sum as the current algorithm. The other two types of mechanisms M^G and M^c are simply a single round of Gaussian mechanism and hence each of them has a log moment of $\frac{\lambda(\lambda-1)}{2\sigma^2}$ [53].

By the composition of moments [1]), the log moment of the entire sequence of adaptive mechanisms can be bounded as follows:

$$\begin{aligned}\alpha_{M_1,\ldots,M_k}(\lambda) &\leq T \cdot \frac{q^2\lambda^2}{\sigma^2} + T_f \cdot \frac{\lambda(\lambda-1)}{2\sigma^2} \cdot 2 \\ &\leq \frac{\lambda^2}{\sigma^2} \cdot (Tq^2 + T_f).\end{aligned}$$

By the tail bound [1], to guarantee Algorithm 2 to be (ϵ, δ)-differentially, we just need $\epsilon < c_1(q^2T + T_f)$ and $\sigma \geq c_2 \frac{\sqrt{(q^2T+T_f)\log(1/\delta)}}{\epsilon}$, by the similar argument of Theorem 1 in DPSGD [1]. □

5.4 Hyperparameter Tuning

Hyperparameter tuning for differentially private machine learning tasks is a cumbersome task and the parameters for the non-private version of the task do not apply directly to the private version [58]. To begin with, deep learning tasks have multiple parameters to tune (For e.g., network size, learning rate, no. of layers, activation function and type of optimizer) and to make the task even harder, differential privacy brings with itself parameters such as clipping threshold C and noise multiplier σ. To find a good set of hyperparameters using grid-search is not only computationally expensive but also time-consuming. From an end-to-end differential privacy stand view, even running grid-search multiple times would add up to the privacy cost every time we run a different set of parameters. This overhead cost can be reduced if a public set of points is available and the tuning can be done on that data instead. However, to handle this tuning issue privately, previous work has suggested multiple tuning techniques [48, 26, 9]. Chaudhuri et al. in Algorithm 4 of [9] present a technique similar to k-fold validation where the dataset is divided into n parts and the tuning for different parameters is done on each part separately by dividing the cost. In [48, 26], the authors provide techniques from theory which can reduce the search space to get the best parameters as we are interested only in the parameters which will result in maximum accuracy.

In DIFFINDO, we add two more parameters to tune, clipping threshold C_2 for G (Gradients with respect to input) and the removal multiplier p. C_2 can be tuned the exact same way as C and is inherently different for each dataset. Parameter p can be harder to tune as it controls the total number of points discarded at each removal iteration. A low value can diminish the removal rate and a high value can, in turn, remove a lot of good points. If the number of outlier points is known beforehand, then we observe that the best removal strategy is to remove 2.5 - 3x the total outliers. Ideally, the value of p should be set so that it removes at each iteration, 2.5x outliers over the total number of removal iterations. For our experiments, we assume that the model curator knows the total outlier points from prior and tune p by repeating our training procedure so that it removes approximately 2.5x the total outliers. Another good strategy to tune hyperparameters is to start by relaxing all privacy conditions (i.e; make $\sigma = 0$, $C = \infty$, $C_2 = \infty$) and slowly start tuning one parameter at a time. First, tune C for DPSGD so that none of the higher gradient norms is discarded due to clipping. Second, start tuning σ to get reasonable privacy guarantees and then finally, tune C_2 and removal multiplier for DIFFINDO to remove the desired number of points. We admit that this tuning method is not private and in Chapter 7, we discuss for future work how we plan wish to make the entire hyperparameter tuning procedure automatic by using a clean validation set.

Chapter 6

Evaluation

In this chapter, we present extensive experiments on how our algorithm DIFFINDO performs on real-life datasets with label flipped in the training set. We start by showing the experimental setup and then list down all the experiments that we perform and then show the hyperparameters that we use for each experiment.

6.1 Experimental Setup

Datasets

Our experiments are done on three real-world datasets from different domains of machine learning research and computer-aided medical diagnosis.

MNIST The MNIST handwritten numbers dataset [41], which consists of 60000 training and 10000 testing examples. All images are 768 pixels (28x28) labelled from 0 to 9, representing the number in its handwritten form. The training set has equal support for all classes. We create noisy label flipped versions of this dataset by targeted strong label flip and composite flips and train using two neural network models, MLP and CNN. The strongest label flips are generated by changing the labels of 1's to 7's in the training set. We choose 1 and 7 as they look very similar, and this kind of noise can affect the testing accuracy of the model. This choice has also been previously considered by prior work on data poisoning attacks [73, 49]. We vary the value of this element from 0 to 50% in our experiments with an increment of 5%. We also try a different form of label flip where multiple source classes are flipped, also called *composite label flips*. These types of flips are common in day to day practice. To show composite label flip attacks, we first show a weak composite flip attack where $10 - 20\%$ of multiple source classes are changed to different target classes (one-to-one mapping) and second, we show a strong composite flip attack $25-30\%$ where some of the label flips have been changed from different classes to the same output class (many-to-one mapping for, e.g. classes 0,4 and 9 have been flipped to class 1).

1. **Weak composite flips:** The weak composite attack has multiple $10 - 20\%$ label flips from a *single* source class to a *single* target class. The transition matrix for the attack looks as follows:

$$T_{weak} = \left[\begin{array}{ccccccccccc}
\text{Source} \downarrow & & & & & & & & & & \\
\text{Target} \rightarrow & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
0 & 0 & 0 & 0 & 0 & 0.15 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.15 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 \\
5 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.15 & 0 \\
7 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 \\
9 & 0 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]$$

2. **Strong composite flips:** The strong composite attack has multiple $25 - 30\%$ label flips from a *multiple* source classes to a *multiple* target classes. The transition matrix for the attack looks as follows:

$$T_{strong} = \left[\begin{array}{ccccccccccc}
\text{Source} \downarrow & & & & & & & & & & \\
\text{Target} \rightarrow & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
0 & 0 & 0.1 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 \\
4 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0 & 0 & 0 \\
9 & 0 & 0.05 & 0 & 0 & 0.15 & 0 & 0 & 0 & 0 & 0
\end{array}\right]$$

ENRON The ENRON email dataset [51] is a spam email dataset with 4137 training points and 1035 testing points. All data points have 5116 dimensions and are labelled as 1 if it's spam and 0 otherwise. Furthermore, this dataset has class imbalance with 70% ham emails and 30% spam emails. We simulate label flip attacks as well as adversarially additive attacks for this dataset. For our experiment, we flip labels from Class 0 to Class 1 by varying Δ from $0-30\%$ with an increment of 5%. We train on this dataset using a logistic regression model. As an extension of our study, we study some adversarial attacks on this dataset. We move the experiment to Appendix A as adversarial attacks are slightly out of context for this thesis.

APTOS The APTOS blindness dataset is a large set of 13000 retina images taken using fundus photography under a variety of imaging conditions [32]. These retina images are used to detect diabetic symptoms in a patient using diabetic retinopathy (DR). Each retina image is of different dimensions and belongs to one of 5 classes - 0 - No DR, 1 - Mild, 2 - Moderate. 3 - Severe, 4 - Proliferative DR. For our experiment purpose, we only take a curated subset of this dataset with 3600 points of 224x224 dimensions [33]. We also re-scale each image down to 50% of its original size to 112x112 for ease of computation. Due to high class imbalance in the dataset, we further combine classes 1, 2, 3 and 4 into one class, making it a binary classification task. Similar to ENRON, we create a noisy version of this dataset by flipping class 0 to class 1 by varying Δ from $0-30\%$ with an increment of 5% and train using a logistic regression model.

In Table 6.1, we provide a detailed description of all the different input datasets.

Metrics

Two metrics are used to measure the defence against label flips. First, we report the precision of the target class or how well the model is able to learn the target class. The second metric measures the accuracy of the model on the full testing set — the fraction of the testing examples that are reported correctly. We run each algorithm three times and report the average precision score or testing accuracy. Mathematically, testing accuracy (TA) and Precision (P_a) for class a can be written as follows:

$$TA = \frac{CorrectlyPredicted}{TotalPoints} \quad P_a = \frac{TruePositives_a}{TruePositives_a + FalsePositives_a}$$

In the above equation, $TruePositives_a$ and $FalsePositives_a$ is defined as the outcomes where the model correctly predicts class a and model incorrectly predicts some other class instead of a respectively.

Implementation

All experiments are done on a computer with specifications of 8GB Geforce GTX 1080 GPU, core i7 3.4 GHz processor and 62GB RAM. The source code for our experiments is written in Python using PyTorch, and implementation of DPSGD is used from the Pyvacy [81] library. We carry out experiments using 3 different models. The experiments on APTOS and ENRON are done on a simple logistic regression model. MNIST experiments are done using a 1 hidden layer fully connected Multilayer Perceptron (MLP) with Rectified Linear Unit (ReLU) activation and a CNN model with 2 convolutional layers and a fully connected layer at the end using cross entropy loss. We repeat all the experiments for 3 iterations and report the average.

Dataset	Noise	Dataset Size	Classes	Model	Support in Training Set
MNIST	i) Targeted Flip from 1 to 7; ii) Weak composite T_{weak}; iii) Strong composite T_{strong}	60000, 10000	10	i) MLP ii) CNN	Class 0 - 5923 Class 1 - 6742 Class 2 - 5958 Class 3 - 6131 Class 4 - 5842 Class 5 - 5421 Class 6 - 5918 Class 7 - 6265 Class 8 - 5851 Class 9 - 5949
ENRON	Targeted Flip from 0 to 1	4137, 1035	2	Logistic Regression	Class 0 - 2928 Class 1 - 1209
APTOS	Targeted Flip from 0 to 1	1857,1805	5	Logistic Regression	Class 0 - 1372 Class 1 - 1374

Table 6.1: List of datasets

Baseline

For our experiments, we compare DIFFINDO against three baselines. Our first baseline is DPSGD [72, 5, 1]. We run DPSGD on the training data with noisy labels for the same privacy budget. If DIFFINDO does not plan to remove any points (Lines 1-10 of Algorithm 2 or when $T_f = 0$), then it is equivalent to DPSGD. The privacy cost for removal in DIFFINDO is compensated by running DPSGD for more number of epochs. One may choose to use the extra privacy budget to decrease the value of σ, but that might not be sufficient enough for significant improvement compared with extra epochs of training or running DIFFINDO. Our second baseline is a private unsupervised clustering algorithm, IMSAT [30]. According to a recent survey [52, 3], IMSAT is the state of the art unsupervised deep clustering technique. It uses clustering entropy loss and self augmentation to achieve an accuracy of 98.4% on MNIST. We choose this as our DP baseline because one may wish to ignore the labels and learn the dataset using an unsupervised clustering task. We experiment on IMSAT by applying a differentially private counterpart of the optimizer used in IMSAT called Adam [36]. And our final baseline is the non-private vanilla SGD. We run a non-private version of DIFFINDO ($\epsilon = \infty$) and compare it against SGD when trained on the same data with noisy labels. For the non-private scenario for DIFFINDO, instead of using a filter condition and threshold for removal, we remove a pre-computed top percentile of points. The percentile is calculated approximately by dividing the 2 times the total number of outliers over total removal iterations (T_f). We do this because our experiments for filter condition show that DIFFINDO performs best when we remove points in every epoch. This is a smart choice for the non-private scenario now that we can use the FILTER function for free.

6.2 End-to-End Evaluation

In the end-to-end evaluation, we compare DIFFINDO with the aforementioned baselines across multiple inputs.

6.2.1 Targeted Strong Label Flips

We show the performance of DIFFINDO on targeted label flip attacks for three different datasets – MNIST, ENRON and APTOS. These datasets are inherently unique. MNIST is a balanced image dataset whereas ENRON is a numerical data and suffers from class imbalance. APTOS is a medical image dataset and is also affected by class imbalanced but has been synthetically balanced by combining multiple output classes. Our experiments show that DIFFINDO can improve the testing accuracy and the precision of the targeted label when compared to baseline DPSGD for the same privacy budget for each of the datasets.

MNIST

In our experiments for MNIST, we run DIFFINDO with the following default setting. We set lot size (L) as 250 and run 50 epochs, where each epoch consists of n/L lots. The default values for σ, C and C_2 are 1.1, 1 and 0.002 respectively. The removal condition in Algorithm 2 is set such that the FILTER function (Algorithm 3) is called after the 10^{th} epoch in an interval of 3 epochs each. As the removal is done only after 12 epochs of training, we pay for the equivalent cost of an extra 24 epochs in the moments accountant analysis. The privacy cost of DIFFINDO with this configuration gives ϵ as 3.6 with δ as 10^{-5}. The filter multiplier p is set adaptively in every epoch from 1.6 to 2.2 times the score of the noisy average gradient by DIFFINDO.

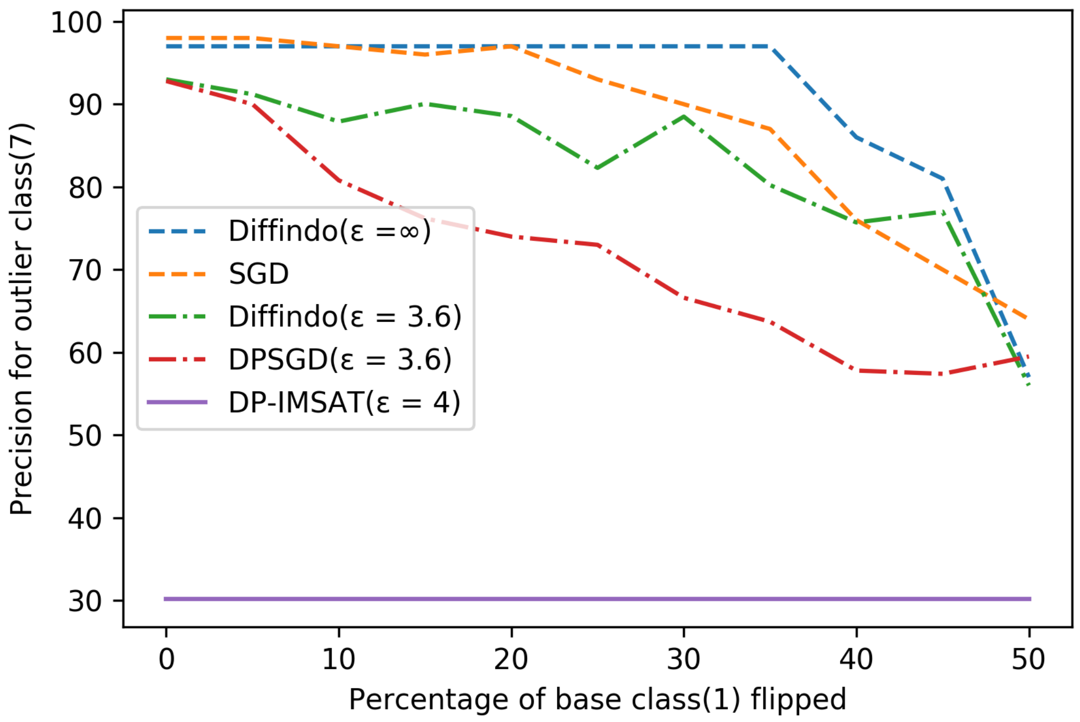

Figure 6.1: DIFFINDO has a higher precision than the baseline algorithms (DP-IMSAT and DPSGD).

We run DPSGD on the training data with noisy labels. Given the same privacy budget and σ, DPSGD can run for a total of 74 epochs, as no privacy budget is required for removing points.

Poison % (Δ)	**Precision for target label**	**Testing accuracy**	**Outliers removed**	**Removal %**
30	$90 \rightarrow 97$	$97 \rightarrow 97.5$	2022	100 %
35	$87 \rightarrow 97$	$96.8 \rightarrow 97.6$	2353	99.74 %
40	$76 \rightarrow 86$	$95.6 \rightarrow 97.6$	2507	92.98%
45	$70 \rightarrow 86$	$94.8 \rightarrow 97.7$	1822	60%

Table 6.2: Results on improvement of accuracy for DIFFINDO ($\epsilon = \infty$) compare to SGD on MNIST

Poison % (Δ)	**Precision for target label**	**Testing accuracy**	**Outliers removed**	**Removal %**
30	$66.6 \rightarrow 88.55$	$87.7 \rightarrow 92$	4220	86.59 %
35	$63.7 \rightarrow 80.2$	$86.5 \rightarrow 90.4$	4484	86 %
40	$57.8 \rightarrow 75.7$	$89 \rightarrow 89.74$	4647	71.58%
45	$57.4 \rightarrow 77$	$86.5 \rightarrow 89.7$	4805	59%

Table 6.3: Results on improvement of accuracy for DIFFINDO ($\epsilon = 3.6$) compared to DPSGD ($\epsilon = 3.6$) on MNIST

First, we show that our proposed algorithm DIFFINDO improves the precision of the target class with respect to the baseline algorithms, DPSGD and DP-IMSAT. As shown in Figure 6.1, given the same privacy budget $\epsilon = 3.6, \delta = 10^{-5}$, DIFFINDO (the green dot-dashed line) has a 5% to 20% higher precision than DPSGD algorithm (the red dot-dashed line), when the fraction of 1s flipped to 7s, Δ (T_{17} element of the noise transition matrix), is between 10% and 40%. When $\Delta = 0$, there are no noisy labels; both algorithms can achieve similar accuracy given a sufficient amount of iterations. However, when $\Delta = 50\%$, both algorithms have relatively poor precision for class 7, as the images for class 1 are no longer outliers to class 7 in the training data. However, DIFFINDO and DPSGD have much better precision compared to DP-IMSAT, a private unsupervised learning algorithm. This baseline is delicate to noise and has poor accuracy at any reasonable value of ϵ (e.g. $\epsilon = 4.0$ in Figure 6.1). The plot for DP-IMSAT is horizontal as it is independent of the labels.

We detail the comparison results between non-private DIFFINDO and SGD in Table 6.2 for the range of poison fraction values $\{30, 35, 40, 45\}\%$. We see that both the precision for target class and the test accuracy for the full testing data are improved by DIFFINDO with respect to SGD. The total number and percentage of correct outliers are also reported in the last two columns. The precision for the targeted class increases by 7-16%, and the test accuracy is restored to 97-98% by removing points.

Similarly, we show the accuracy improvement of DIFFINDO with respect to DPSGD at $\epsilon = 3.6$ and $\delta = 10^{-5}$ in Table 6.3. We show that DIFFINDO improves the precision of the target class of DPSGD by 10-16% and restores the test accuracy to 89-92%.

ENRON

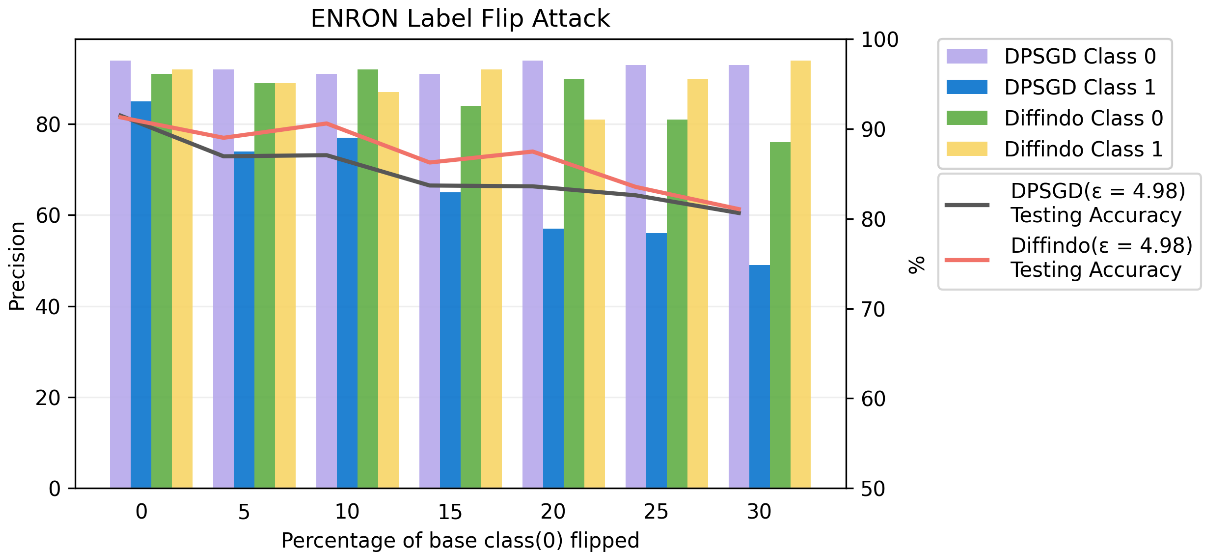

Figure 6.2: Precision increase for target label (1) for DIFFINDO on ENRON

In our experiments for ENRON, we run DIFFINDO with the following default setting. We set lot size (L) as 250 and run 50 epochs, where each epoch consists of n/L lots. The default values for σ, C and C_2 are 2, 1 and 0.002 respectively. The removal condition in Algorithm 2 is set such that the FILTER function (Algorithm 3) is called after the 15^{th} epoch in an interval of 5 epochs each. As the removal is done only after 7 epochs of training, we pay for the equivalent cost of an extra 14 epochs in the moments accountant analysis. The privacy cost of DIFFINDO with this configuration gives ϵ as 4.98 with δ as $2 * 10^{-4}$. The filter multiplier p is set adaptively in every epoch from 1.05 to 1.14 times the score of the noisy average gradient by DIFFINDO.

We run DPSGD on the noisy labelled dataset for 64 epochs as a baseline for this experiment.

Poison % (Δ)	**Precision for target label**	**Testing Accuracy**	**Outliers removed**	**Removal %**	**Wrong points removed**
5	74 $\rightarrow$ 89	86.93 $\rightarrow$ 89	266	91.15 %	132
10	77 $\rightarrow$ 87	87.06 $\rightarrow$ 90.59	414	92.12 %	145
15	65 $\rightarrow$ 92	83.68 $\rightarrow$ 86.24	1089	99.08 %	654
20	57 $\rightarrow$ 81	83.59 $\rightarrow$ 87.46	650	85.29 %	151
25	56 $\rightarrow$ 90	82.59 $\rightarrow$ 83.53	1575	99.72 %	845
30	49 $\rightarrow$ 94	80.62 $\rightarrow$ 81.06	1915	99.54 %	1041

Table 6.4: Results on improvement of accuracy for DIFFINDO ($\epsilon = 4.98$) compared to DPSGD ($\epsilon = 4.98$) on ENRON

In Figure 6.2, we show that given the same privacy budget $\epsilon = 4.98, \delta = 2 * 10^{-4}$, the precisions of classes 0 and 1 as well as the testing accuracies for both DPSGD and DIFFINDO. Note that, the second y-axis for accuracy on the right has limits from 50 to 100 for better scaling. First, we show that there is a considerable increase in the precision of the target label (Blue bar vs Yellow bar) and testing accuracy (Black line vs Orange line). However, this increase comes at the cost of the precision for the base class 0 (Lilac bar vs Green bar). This decrease is seen due to the class imbalance in the training set and wrong removal of representative points.

We detail the results of the experiment in Table 6.4. DIFFINDO has target label precision increase of $10 - 45\%$ when $10 - 30\%$ of 0s have been flipped to 1s and successfully removes 85-90% of the outlier points. The removal of outlier points not only increases the precision but also results in up to 3% increase in the testing accuracy. However, we notice that out of all the points removed, DIFFINDO removes approx. 50% wrong points which result in the decrease of the base class precision.

APTOS

We also run DIFFINDO on a diabetic retinopathy classification task on the APTOS dataset with similar settings as that for ENRON. The experiment uses the same default values for σ, C and C_2 as 2, 1 and 0.002 respectively. The removal condition in Algorithm 2 is set

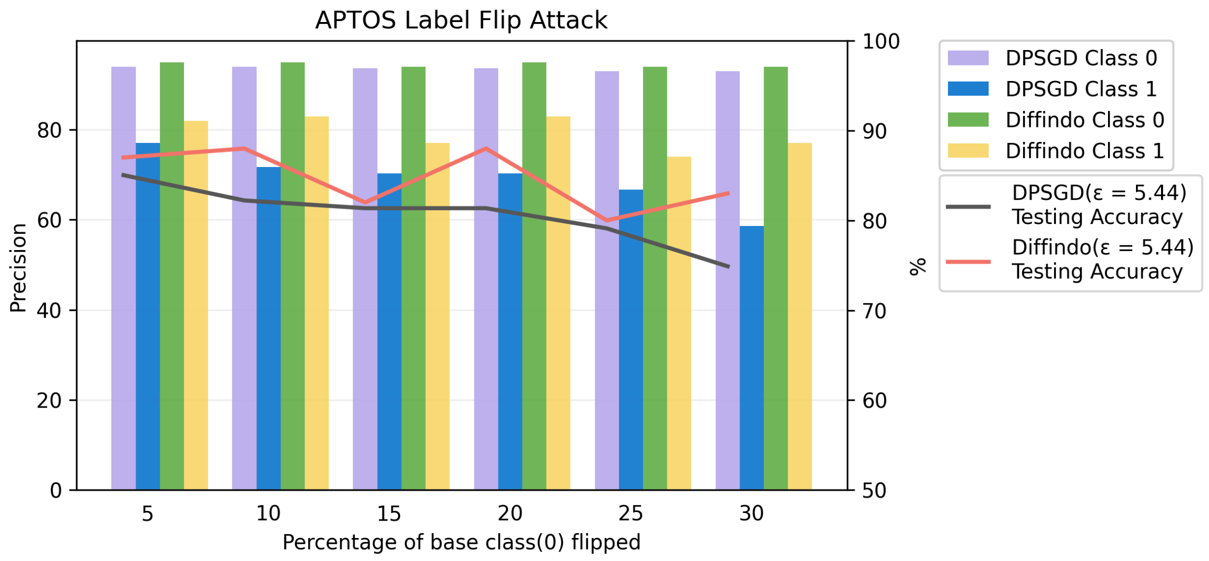

Figure 6.3: Performance of DIFFINDO on medical dataset APTOS

such that the FILTER function (Algorithm 3) is called after the 15th epoch in an interval of 5 epochs each. As the removal is done only after 7 epochs of training, we pay for the equivalent cost of an extra 14 epochs in the moments accountant analysis. The privacy cost of DIFFINDO with this configuration gives ϵ as 5.44 with δ as $3 * 10^{-4}$. As multiple classes have been combined to make the positive class for this dataset, many points lie very close to each other. This affects the removal fraction p to be highly sensitive and even 0.1 change in the value results in heavy removal of points thus, p is set adaptively in every epoch from 1 to 1.015 times the score of the noisy average gradient by DIFFINDO to remove a reasonable number of points.

The logistic regression model used for this experiment can get to about 92% accuracy in the non-private scenario however the private version gets about 88% accuracy. As a private baseline for this experiment, we run DPSGD on the training data with noisy labels for 64 epochs. In Figure 6.3, we show the precisions of classes 0 and 1 as well as the testing accuracies for both DPSGD and DIFFINDO given the same privacy budget $\epsilon = 5.44, \delta = 2 * 10^{-4}$. The right y-axis denotes the accuracy % and is limited from 50-100 for better visibility. First, we show that label flipped noise drastically affects this dataset as the testing accuracy is lowered to 75% when Δ is 30% and the precision of target label is decreased to 59% (almost equal to random guessing). Secondly, we show that DIFFINDO can increase the precision of the target label (Blue bar vs yellow bar) by an average of

10% while also increasing the testing accuracy (Black line vs Orange line) considerably. Despite the sensitive nature of the APTOS dataset, we notice that each point in the 0 class is highly representative and shows more stable behaviour than the ENRON dataset. This is proven by the fact that removing points does not affect the base class precision (Lilac bar vs Green bar).

Poison % (Δ)	**Precision for target label**	**Testing Accuracy**	**Correct/ Outliers removed**	**Removal %**
5	$77 \rightarrow 82$	$85.03 \rightarrow 87$	53/288	77.94 %
10	$71.66 \rightarrow 83$	$82.22 \rightarrow 88$	74/122	54.01 %
15	$70.33 \rightarrow 77$	$81.35 \rightarrow 82$	140/487	68.29 %
20	$70.33 \rightarrow 83$	$81.35 \rightarrow 88$	200/328	72.99 %
25	$66.66 \rightarrow 74$	$79.11 \rightarrow 80$	165/521	48.10 %
30	$58.66 \rightarrow 77$	$74.89 \rightarrow 83$	160/405	38.92 %

Table 6.5: Results on improvement of accuracy for DIFFINDO ($\epsilon = 5.44$) compared to DPSGD ($\epsilon = 5.44$) on APTOS

We detail the results of the experiment in Table 6.4. DIFFINDO shows an increase of 5 - 19% precision of the target label when Δ is changed from 5-30%. Despite the high removal of wrong points (for e.g., 365 wrong vs 165 correct when Δ = 25%), DIFFINDO shows an increase of 2-9% in testing accuracy. This behaviour shows the stable nature of the APTOS dataset and in such cases, DIFFINDO can be allowed some leverage to remove more points.

In this section, we showed how DIFFINDO increases the metrics– testing accuracy and precision of the target label for models which train on targeted label flipped datasets. Furthermore, we show that imbalanced datasets show a decrease in base class precision after using DIFFINDO however balanced and synthetically balanced datasets do not show this behaviour.

6.2.2 Composite Label Flips

Composite label flips are attacks where multiple output classes are affected. In real life datasets, it is more common to see composite label flips than targeted flips as this closely resembles minor human labelling errors. In this experiment, we run DIFFINDO on two curated composite flip attacks – weak composite and strong composite attack. The weak composite has label flips ranging from $\Delta = 10-20\%$ with a one-to-one mapping of source to target and the strong composite attack ranges from $\Delta = 15-30\%$ with multiple instances of many-to-one mapping of source to target. The transition matrices for both these attacks are described in Section 6.1. The values for the transition matrices are chosen at random to demonstrate various label flip instances. We run DIFFINDO on the same default setting as targeted strong label flips on MNIST for both types of composite attacks. The results are compared to a baseline of DPSGD on the same input.

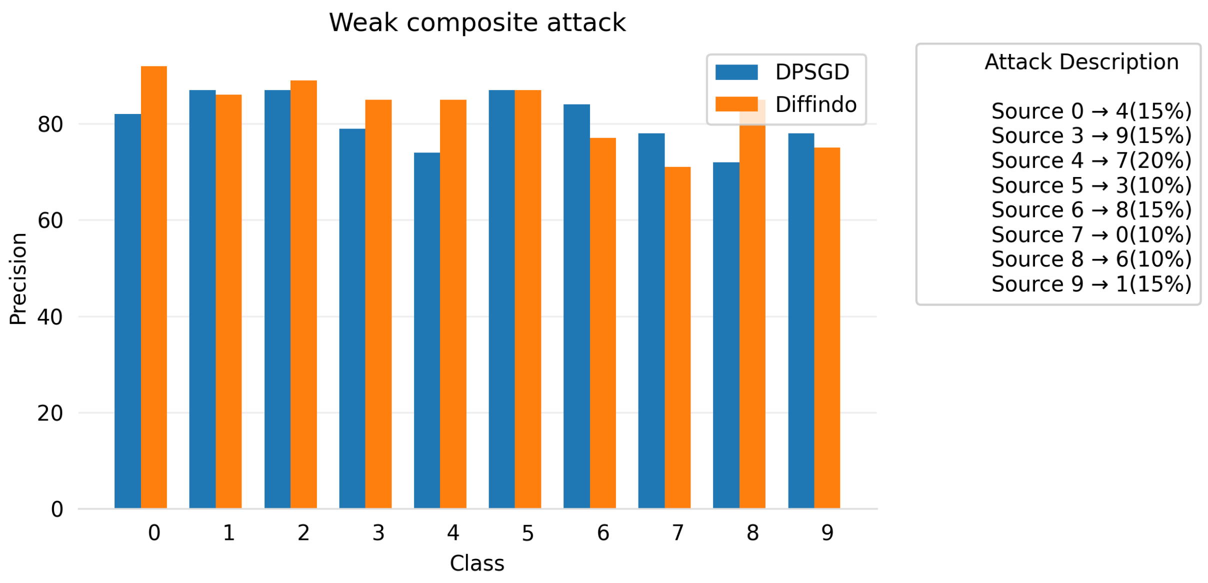

Figure 6.4: Precision comparison for all output classes on MNIST T_{weak}

We plot the change in precision for all the class in Figure 6.4 and Figure 6.5. The plots show the precision for each label in the dataset for DIFFINDO and baseline DPSGD along with a brief description of the attack in the legend to its right. The weak attack has a total of 7400 label flipped points and DIFFINDO is successful in removing 60% of the points by removing a total of 10540 points. There is an overall increase in testing accuracy from 80.6% to 84.3% but as can be seen from the Figure 6.4. The strong attack has a total of 9161 label flipped points and DIFFINDO is successful in removing 44% of the points by removing a total of 7213 points and an increase of 4% in testing accuracy from 76% to 80%.

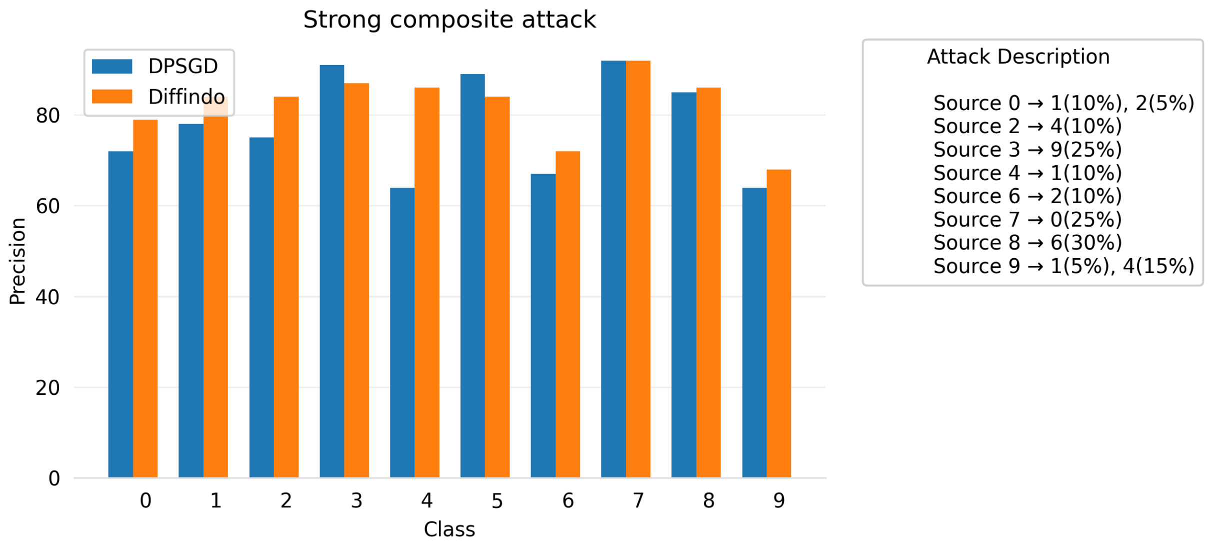

Figure 6.5: Precision comparison for all output classes on MNIST T_{strong}

	Weak Attack Precisions (%)		Strong Attack Precisions (%)	
Class	**DPSGD**	**Diffindo**	**DPSGD**	**Diffindo**
0	82	92	72	79
1	87	86	78	84
2	87	89	75	84
3	79	85	91	87
4	74	85	64	86
5	87	87	89	84
6	84	77	67	72
7	78	71	92	92
8	72	85	85	86
9	78	75	64	68

Table 6.6: Composite attack results for DIFFINDO ($\epsilon = 3.6$) compared to DPSGD ($\epsilon = 3.6$)

The details of the experimental results can be found in Table 6.6. A decrease in precision is observed for the non-affected or weakly affected classes. Classes 6,7 and 9 in the weak composite attack and classes 5 and 3 in the strong composite show this dip in precision. This behaviour can be observed abundantly when a higher percentage of points have been flipped, and the algorithm tries to remove points from multiple classes to counter the attack.

In this section, we showed the performance of DIFFINDO on composite label flipped attacks. Our experiments on MNIST for a weak and strong composite attack showed that DIFFINDO increases the final testing accuracy. However, the precision for weakly affected classes decreases due to removal of superflous points.

6.2.3 Performance on Convolutional Neural Network

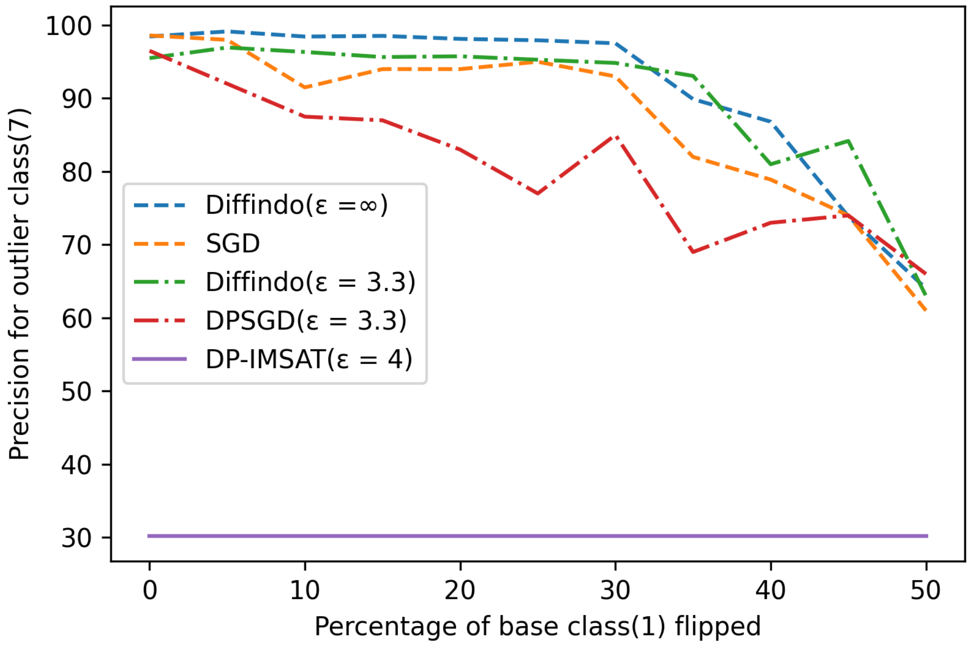

Figure 6.6: CNN Model for DIFFINDO vs baseline algorithms (DP-IMSAT and DPSGD).

We show another experiment using the same attack but on a convolutional neural network (CNN) to show that DIFFINDO is not specific to a Neural Network(NN) model but can also be used on more complex networks. The non-private accuracy on MNIST with SGD optimizer for this CNN model is about 99%, and CNNs are generally also more favourable for image datasets. For this experiment, we repeat the same settings and baselines of the targeted strong label flip experiment. We show that even for a CNN, Diffindo can perform better than the DPSGD baseline for the same privacy budget ($\epsilon = 3.3, \delta = 1e - 5$) and can also sometimes perform better than the non-private baseline.

We keep all similar settings but change the parameters $C = 0.5$ and $C_2 = 0.003$ to correctly clip gradients for this new network. The comparison of Diffindo with the baselines is given in Figure 6.6. The plot compares the precision of the targeted label for each algorithm. First, we show that DP-IMSAT is the weakest baseline and is the least favourite for learning with label flips. Second, we show that label flips strongly affect the precision of the target label (7) in both cases of the orange line of SGD and the red line DPSGD ($\epsilon = 3.3$). Thirdly, we show that non-private Diffindo (Blue) and Diffindo ($\epsilon = 3.3$) (Green) perform better than their respective counterparts and considerably increase the precision. Fourth and most interestingly, we show that private Diffindo ($\epsilon = 3.3$) shows higher precision compared to SGD when $\Delta \geq 10\%$.

Poison % (Δ)	**Precision for target label**	**Testing accuracy**	**Outliers removed**	**Removal %**
5	$98 \rightarrow 99.15$	$98.67 \rightarrow 99.06$	714	97.03 %
10	$91.5 \rightarrow 98.45$	$97.9 \rightarrow 98.9$	1411	97.77 %
15	$94 \rightarrow 98.54$	$98 \rightarrow 98.99$	2098	99.80 %
20	$94 \rightarrow 98.13$	$98 \rightarrow 98.99$	2778	99.70 %
25	$95 \rightarrow 97.94$	$97 \rightarrow 98.94$	3452	99.28 %
30	$93 \rightarrow 97.53$	$98 \rightarrow 98.72$	4116	99.30 %
35	$82 \rightarrow 89.9$	$91.9 \rightarrow 97.97$	4773	96.48 %
40	$78.9 \rightarrow 86.81$	$95 \rightarrow 97.42$	5424	87.16 %

Table 6.7: Results on improvement of accuracy for Diffindo ($\epsilon = \infty$) compare to SGD

In Table 6.7, we detail the comparison between the SGD and Diffindo ($\epsilon = \infty$) for Δ from 5 - 40%. Diffindo is successful in removing approx. 99% of the outlier points and restore the precision of the target label and the testing accuracy perfectly to 98% and 99% respectively when $\Delta \leq 30\%$. For $\Delta > 30\%$, the removal is weaker, yet the precision is increased by up to 8%, and the testing accuracy is restored to approx. 98%.

Similarly, in Table 6.8, we describe the comparison results of DPSGD vs Diffindo for the same privacy budget. In this experiment setting, Diffindo requires a privacy cost of $\epsilon = 3.3$ and thus DPSGD is allowed to run for 24 extra epochs to compensate for the extra removal privacy cost ($3.3 - 2.6 = 0.7$) of Diffindo. In the $\Delta = 0$ setting i.e no label flip setting, DPSGD gets about 96% accuracy. We see in our experiments that for $\Delta \leq 30\%$,

Poison % (Δ)	**Precision for target label**	**Testing accuracy**	**Outliers removed**	**Removal %**
5	$92 \to 96.95$	$91 \to 96.24$	5007	99.10 %
10	$87.5 \to 96.34$	$90.3 \to 96.6$	5370	99.85 %
15	$87 \to 95.65$	$90.1 \to 96.4$	5652	99.60 %
20	$83 \to 95.75$	$89 \to 96.46$	5791	99.77 %
25	$77 \to 95.28$	$89.22 \to 96.32$	6399	99.22 %
30	$85 \to 94.84$	$89 \to 96.14$	6768	97.87 %
35	$69 \to 93.07$	$87.1 \to 95.9$	6919	95.71 %
40	$73 \to 81$	$88 \to 93.89$	6206	46.95 %

Table 6.8: Results on improvement of accuracy for DIFFINDO ($\epsilon = 3.3$) compared to DPSGD ($\epsilon = 3.6$)

DIFFINDO correctly removes 99% of the points restoring the precision of the target label and testing accuracy to 95% and 96% respectively. For $\Delta > 30\%$, DIFFINDO removed a lower number of points but increases the precision for up to 14% and accuracy up to 7%. Stronger label flips than 40% cause poor precision for both private and non-private optimizers as class 1 elements are no more outliers.

Our experiments in this section show the performance of DIFFINDO on targeted label flipped MNIST when trained on a more complex neural network, CNN. DIFFINDO removed almost 100% of the outlier points in both private and non-private settings when the $\Delta \leq 35\%$.

6.2.4 Performance on Varying Noise Levels

The noise multiplier σ affects the testing accuracy and the privacy cost of the models. For the previous experiments, we choose only one value of noise multiplier for a reasonable value of ϵ and vary the other parameters. However, for this experiment, we keep all values constant except σ to test the performance of DIFFINDO on varying noise. We experiment on the MNIST dataset with the MLP model and a targeted strong attack of $\Delta = 30\%$. We compare the precision of the targeted label with a baseline of DPSGD with the same privacy budget.

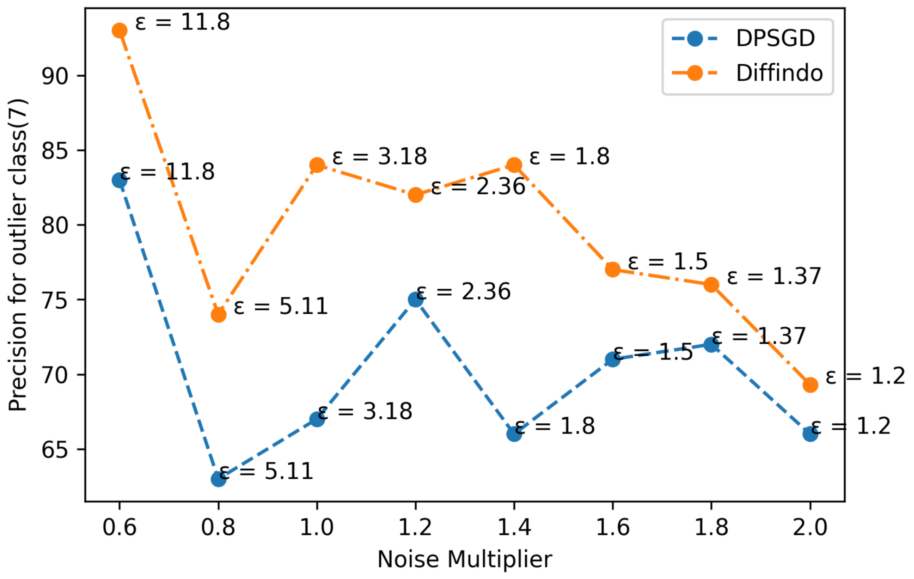

Figure 6.7: Comparison of Diffindo vs DPSGD on varying noise

Noise Mult % (σ)	**Privacy parameter ϵ**	**Precision for target label**	**Testing accuracy**	**Removal %**
0.6	11.8	$83 \to 93$	$90 \to 90.5$	80.76%
0.8	5.11	$63 \to 74$	$88 \to 90.1$	75%
1	3.18	$67 \to 84$	$88 \to 91.2$	87%
1.2	2.36	$75 \to 82$	$89 \to 90.5$	72%
1.4	1.8	$66 \to 84$	$87 \to 90.45$	89.16%
1.6	1.5	$71 \to 77$	$88 \to 89$	70.17%
1.8	1.37	$72 \to 76$	$87 \to 88.13$	70.32%
2	1.2	$66 \to 69.31$	$86 \to 87$	61%

Table 6.9: Performance of Diffindo on varying noise levels

We choose the same task of flipping points with labels from 1 to 7 and keep the Δ value as 30% by varying the value of σ from 0.6 to 2 in increments of 0.2. We also change the value of p from 0.8 when σ is 0.6 and 2.4 when σ is 2 to adapt to the noise added to the gradients. We compare the performance of DPSGD with Diffindo using different noise multipliers in Table 6.9 and plot in Figure 6.7, the precision of the targeted class on the y-axis and the corresponding noise multiplier on the x-axis. We notice a gradual decrease in the removal % and testing accuracy when the value σ is increased. This effect is expected because the noise added prevents the model from learning and impacts the noise to signal ratio.

In this section, we showed that DIFFINDO shows better precision of the targeted label when compared to DPSGD on varying noise levels and privacy parameter ϵ while also increasing the final testing accuracy.

6.3 Effect of Parameters

Now, we study the effect of each tunable parameter for DIFFINDO. The performance of DIFFINDO is determined by multiple parameters that must be carefully tuned for the optimal testing accuracy and removal of outliers. These parameters include the noise multiplier (σ), Clipping thresholds (C_1, C_2), removal multiplier p, filter condition (FC) and learning rate (η). Some of the parameters are specific to the learning of the model (learning rate), some are related privacy (σ, C_1) and the others influence the removal of outliers (C_2, p, FC). We carry out these experiments on the MNIST dataset trained using the MLP model, and the strong label flips from base class 1 to target class 7 with $\Delta = 30\%$ (2022 outliers).

Parameters	**Values**
Clipping Threshold 1(C_1)	$0.5, 1, 1.5, 2, 3, 4, 5$
Clipping Threshold 2(C_2)	$0.001, 0.002, 0.003, 0.004, 0.005$
Filter Interval	$1, 2, 3, 4, 5$
Learning Rate (η)	$0.0001, 0.001, 0.01, 0.1, 1$
Removal parameter (p)	$1, 1.2, 1.4, 1.6, 1.8, 2$

Table 6.10: Parameter study values

To demonstrate the effect of these parameters, we change these parameters by keeping the rest constant. The reference values we set for the model are as follows: lot size (L) as 250, gradient clipping threshold C_1 as 1, C_2 as 0.002, σ as 1, learning rate (η) as 0.1 and initializing the removal multiplier p as 1.6 and adaptively set it in every epoch to a final value of 2.2 times the score of the noisy average gradient. We run the algorithm 50 epochs and set the filter condition such that the FILTER function (Algorithm 3) is called after the 12th epoch and in an interval of 3 epochs each. The results of this study are shown in Figure 6.8. Each graph shows the two metrics – testing accuracy and precision of the target label as well as the total number of correct and wrong points removed for each variable parameter. The blue bar denotes the correct number of points removed, and orange denotes the wrong points removed. Also, note that the second y-axis has is limited from 50 - 100% for better interpretation.

6.3.1 Clipping Thresholds

Diffindo clips the gradient norm bounds at two places. The first gradient norm bound C_1 clips the gradients at each iteration, setting the sensitivity for the Gaussian noise (since we add noise based on σC) and removing the unbiasedness of the gradient estimate. It should also be noted that one might decide to have multiple bounds that can be used for different layers of the neural network. The second gradient norm similarly clips the gradients with respect to input (G). Setting this threshold is crucial as higher bounds might add more noise, and lower bounds may destroy the gradients with high variance. In practice, these thresholds differ across models and datasets and as suggested by [1], it is a good idea to set the clipping norm to the median of gradients.

Our experiments also show that lower values for C_1 are better as it adds lesser noise which results in higher testing accuracy and lower wrong points removal. However, conversely, lower values of C_2 remove a higher number of unnecessary points, which hampers the testing accuracy, and high values result in not being able to remove the outliers. Therefore, moderate values result in optimum performance.

6.3.2 Learning Rate

We experiment on values of learning rate (η) in [0.0001, 0.001, 0.01, 0.1, 1] and notice that if the learning rate is moderately high, the testing accuracy remains the same. Our experiments suggest that with lower values of η, the gradients become smaller and lower clipping threshold C_2 is required and with higher values of η, the clipping threshold C_2 becomes too small and more number of points are removed. Hence, these two values are correlated with each other, and the proper tuning of both should reach the same level of accuracy.

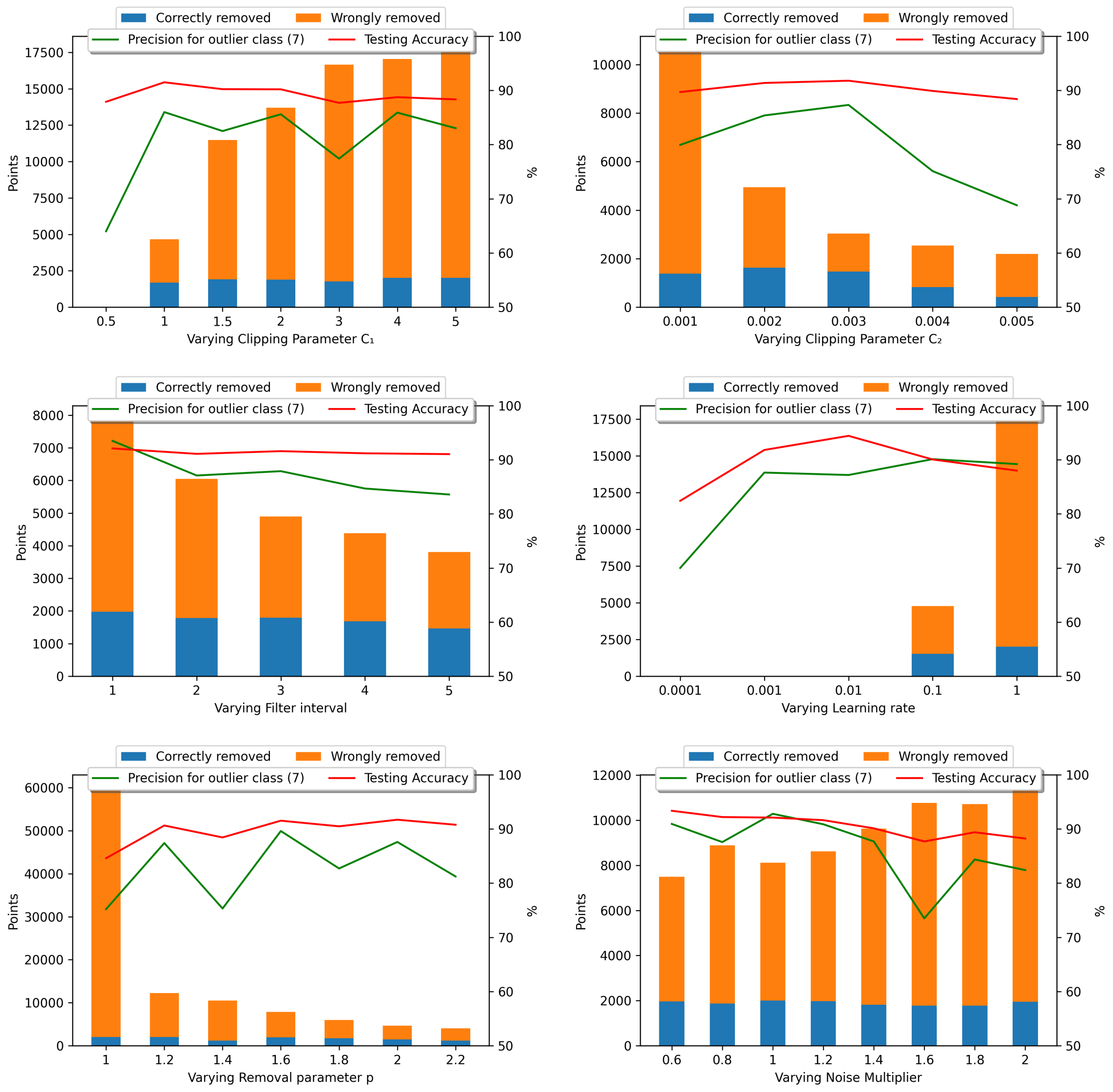

Figure 6.8: Performance of DIFFINDO for MNIST for varying one parameter, and others fixed at reference value

6.3.3 Filter Condition

The filter condition (FC) refers to line 11 in Algorithm 2 and is a crucial element to the algorithm. It decides the number of times that the FILTER function is called. It is important to set the filter condition appropriately as each execution of the FILTER function requires a privacy cost of 2 epochs worth and increases the computation time of the algorithm due to the top eigenvector calculation. We have noticed that the algorithm doesn't remove a noticeable number of outliers during the initial epochs of training. Additionally, the removal of wrong points is costly for the algorithm as the utility of the average gradient in line 16 is lost (due to the constant division by n).

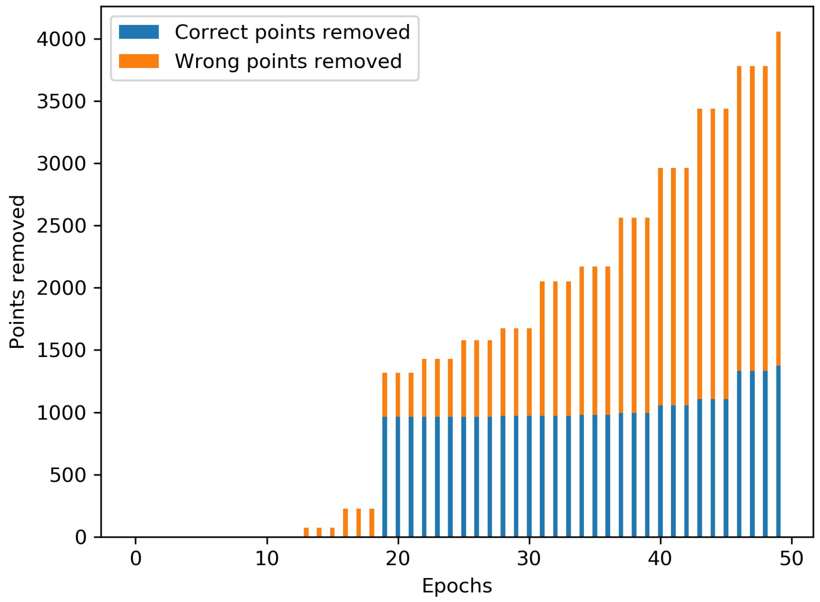

Figure 6.9: Outlier points removal in DIFFINDO at $\epsilon = 3.6$ and $\delta = 10^{-5}$ when Δ is 30%.

We show in Figure 6.9 the number of points removed by DIFFINDO in one run for the MNIST dataset at the default setting. The counts of real outliers are highlighted in blue, and the counts for points that are incorrectly removed are highlighted in orange. We can see in this figure that DIFFINDO started to remove a large number of outlier points from the 19^{th} epoch of training where it removes approximately 1000 points.

In Figure 6.8, we set the initial epochs of training as 12 and vary only the filter interval from 1 to 5. We notice that with interval 1, DIFFINDO removes 100% of the outlier points and achieves high precision value. However, with interval 1, the filter function is called at every epoch resulting in additional privacy cost of 76 epochs $((50-12)*2)$. Thus, privacy vs removal trade-off forces us to choose higher intervals, and an interval of 3 is the right balance between the utility and privacy.

6.3.4 Removal Multiplier

Removal multiplier p refers to the multiplier to the average gradient G in Algorithm 3, which sets the threshold for the filter function to remove outliers. We note that a smaller p removes many unnecessary points, and as we discussed in the previous paragraph, we observe that this removal can sometimes be a gradual process, and a reasonable estimate of the average gradient is required to prevent removal of wrong points. We can help the FILTER algorithm by slowly increasing the removal multiplier p. This is because as DIFFINDO signs of progress and points are removed; more gradients start becoming 0. A higher removal multiplier compensates for this decay and provides a higher threshold, which is required for influential removal.

6.3.5 Noise Multiplier

Noise Multiplier (σ) majorly controls the privacy of the algorithm. For studying the noise parameter, we notice that DIFFINDO removes all the points if the removal multiplier p is too low. For this reason, while changing σ, we also change p by increasing it to the same extent. Our experiments show that DIFFINDO removes approx. the same amount of outlier points in the presence of varying noise. However, with higher noise, the number of wrong points removed increases.

Chapter 7

Conclusion and Open Questions

In this thesis, we propose a novel method called Diffindo which can be used to remove points with noisy labels while preserving differential privacy. To the best of our knowledge, learning with noisy labels hasn't been studied in the private scenario before and Diffindo is the first algorithm to provide a significant increase in the testing accuracy and precision of the affected classes on labelled flipped data under strict privacy conditions. Diffindo is built on top of stochastic gradient descent which allows it to be used on most traditional machine learning and deep learning applications. We demonstrate that Diffindo can revive the testing accuracy of neural networks and logistic regression models trained on targeted and composite label flipped datasets from different domains when compared to DPSGD on the same privacy budget. Now, we would like to discuss some future work and questions that can be considered to make this algorithm stronger.

Tuning using validation set

As we have discussed in Chapter 5, the tuning of differentially private machine learning algorithms is a hard task. However, a public validation set can be used to not only tune the numerous hyperparameters for differential privacy but also can be used to automate Diffindo to find a good removal multiplier p. Although one might be tempted to argue that having a public validation set is a hard requirement for in a privacy setting, an *unlabelled* validation set like PATE [59] might be used or similar public dataset of the same domain may be used.

Federated Learning

Federated learning [38] is an area of research where the dataset is assumed to be divided and distributed into different shards scattered in different locations (might also be geographically distinct). The main idea behind federated learning is to train a machine learning model by the co-ordinating central server which can receive gradients from the different shards and average the gradients to take a step. The presence of distributed data is common in health research which also happens to have a lot of private sensitive data. Thus, algorithms like DIFFINDO will be highly useful in the federated learning scenario. A stronger notion of user differential privacy can be applied to DIFFINDO for learning in the multiple user federated domain.

Adaptive Optimizers

DIFFINDO has only been implemented for SGD. However, deep learning applications especially in the natural language processing field use adaptive optimizers like Adam [36] or RMSProp. These optimizers have the innate ability to work with out-of-the-box hyper-parameters and often do not need further tuning making it a useful tool for differentially private learning algorithms like DIFFINDO.

Data Cleaning

Data cleaning is an important and crucial step in statistical analysis. Most machine learning practitioners in the industry, as well as academia, spent about 60 - 80% of their time on data collection and cleaning [64]. Data cleaning constitutes of many qualitative and quantitative errors and outlier noisy label error and detection happens to be only a fraction of the research. Previous work has proposed many machine learning-based methods for data cleaning which can help in statistical analysis but unfortunately, this happens to be a hard task even in the non-private scenario [11]. As we have already discussed in this thesis, privately cleaning a training dataset can be used in many different fields of research and there are many opportunities for differential privacy to make some of the state-of-the-art data cleaning tasks private.

Relabeling labels

Another extension of this work may include relabeling the datapoints outcasted by the filter algorithm instead of removing them completely from the training process. The relabeling process can be done by the semi-ready model currently in the training process or by a separate model which has been trained on a clean training set. The relabeling technique will not only increase the support of the affected class datapoints by reinstalling them back but also help in tackling the decrease in base class precision issue observed in imbalanced and non-representative datasets like ENRON and APTOS.

References

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

[2] Mohammad Adibuzzaman, Poching DeLaurentis, Jennifer Hill, and Brian D Benneyworth. Big data in healthcare–the promises, challenges and opportunities from a research perspective: a case study with a model database. In *AMIA Annual Symposium Proceedings*, volume 2017, page 384. American Medical Informatics Association, 2017.

[3] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, Maximilian Strobel, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*, 2018.

[4] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, pages 181–190, 2007.

[5] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proc. of the 2014 IEEE 55th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 464–473, 2014.

[6] Carla E Brodley and Mark A Friedl. Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167, 1999.

[7] Mark Bun, Jonathan Ullman, and Salil Vadhan. Fingerprinting codes and the price of approximate differential privacy. *SIAM Journal on Computing*, 47(5):1888–1938, 2018.

[8] Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *arXiv preprint arXiv:1802.08232*, 2018.

[9] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

[10] Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018.

[11] Xu Chu, Ihab F Ilyas, Sanjay Krishnan, and Jiannan Wang. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 International Conference on Management of Data*, pages 2201–2206, 2016.

[12] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high-dimensions without the computational intractability. *SIAM Journal on Computing*, 48(2):742–864, 2019.

[13] Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. *arXiv preprint arXiv:1803.02815*, 2018.

[14] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210, 2003.

[15] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

[16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, TCC '06, pages 265–284, Berlin, Heidelberg, 2006. Springer.

[17] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Machine Learning*, 9(3–4):211–407, 2014.

[18] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[19] Cynthia Dwork, Adam Smith, Thomas Steinke, Jonathan Ullman, and Salil Vadhan. Robust traceability from trace amounts. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 650–669. IEEE, 2015.

[20] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 11–20. ACM, 2014.

[21] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[22] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[23] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

[24] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[25] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, 316(22):2402–2410, 2016.

[26] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1106–1125. SIAM, 2010.

[27] Danna Gurari, Diane Theriault, Mehrnoosh Sameki, Brett Isenberg, Tuan A Pham, Alberto Purwada, Patricia Solski, Matthew Walker, Chentian Zhang, Joyce Y Wong, et al. How to collect segmentations for biomedical images? a benchmark evaluating the performance of experts, crowdsourced non-experts, and algorithms. In *2015 IEEE winter conference on applications of computer vision*, pages 1169–1176. IEEE, 2015.

[28] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[29] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS genetics*, 4(8), 2008.

[30] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1558–1567. JMLR. org, 2017.

[31] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. " i know what you did last summer" query logs and user privacy. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 909–914, 2007.

[32] kaggle. Aptos 2019 blindness detection. https://bit.ly/2RO3qb1.

[33] kaggle. Diabetic retinopathy 224x224 gaussian filtered. https://bit.ly/34LGmPc.

[34] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. *IEEE Transactions on Information Theory*, 63(6):4037–4049, 2017.

[35] Davood Karimi, Haoran Dou, Simon K Warfield, and Ali Gholipour. Deep learning with noisy labels: exploring techniques and remedies in medical image analysis. *arXiv preprint arXiv:1912.02911*, 2019.

[36] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[37] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2018.

[38] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[39] Jan Kremer, Fei Sha, and Christian Igel. Robust active label correction. In *International Conference on Artificial Intelligence and Statistics*, pages 308–316, 2018.

[40] Kevin A Lai, Anup B Rao, and Santosh Vempala. Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 665–674. IEEE, 2016.

[41] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[42] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[43] Choong Ho Lee and Hyung-Jin Yoon. Medical big data: promise and challenges. *Kidney research and clinical practice*, 36(1):3, 2017.

[44] Kimin Lee, Sukmin Yun, Kibok Lee, Honglak Lee, Bo Li, and Jinwoo Shin. Robust inference via generative classifiers for handling noisy labels. *arXiv preprint arXiv:1901.11300*, 2019.

[45] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5447–5456, 2018.

[46] Erich L Lehmann and George Casella. *Theory of point estimation.* Springer Science & Business Media, 2006.

[47] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Learning to learn from noisy labeled data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5051–5059, 2019.

[48] Jingcheng Liu and Kunal Talwar. Private selection from private candidates, 2018.

[49] Xuanqing Liu, Si Si, Xiaojin Zhu, Yang Li, and Cho-Jui Hsieh. A unified framework for data poisoning attack to graph-based semi-supervised learning. *arXiv preprint arXiv:1910.14147*, 2019.

[50] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In *2008 IEEE 24th international conference on data engineering*, pages 277–286. IEEE, 2008.

[51] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes-which naive bayes? In *CEAS*, volume 17, pages 28–69. Mountain View, CA, 2006.

[52] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[53] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[54] Shubhankar Mohapatra, Xi He, Gautam Kamath, and Om Thakkar. Diffindo! differentially private learning with noisy labels. *PPAI Workshop, AAAI*, 2019.

[55] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *CoRR*, abs/cs/0610105, 2006.

[56] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *arXiv preprint arXiv:1812.00910*, 2018.

[57] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[58] Nicolas Papernot, Steve Chien, Shuang Song, Abhradeep Thakurta, and Ulfar Erlingsson. Making the shoe fit: Architectures, initializations, and tuning for learning with privacy, 2020.

[59] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

[60] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1944–1952, 2017.

[61] Andrea Paudice, Luis Muñoz-González, and Emil C Lupu. Label sanitization against label flipping poisoning attacks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 5–15. Springer, 2018.

[62] Kees H Polderman, Lambert G Thijs, and Armand RJ Girbes. Interobserver variability in the use of apache ii scores. *The Lancet*, 353(9150):380, 1999.

[63] Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*, 2018.

[64] Gil Press. Cleaning big data: Most time-consuming, least enjoyable data science task, survey says. https://bit.ly/2V78OIi.

[65] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[66] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*, 2018.

[67] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[68] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

[69] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, 2004.

[70] Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. Genomic privacy and limits of individual detection in a pool. *Nature genetics*, 41(9):965, 2009.

[71] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[72] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[73] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, pages 3517–3529, 2017.

[74] Sainbayar Sukhbaatar, Joan Bruna Estrach, Manohar Paluri, Lubomir Bourdev, and Robert Fergus. Training convolutional networks with noisy labels. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[75] Latanya Sweeney. Only you, your doctor, and many others may know. *Technology Science*, 2015092903(9):29, 2015.

[76] Om Dipakbhai Thakkar. *Advances in Privacy-Preserving Machine Learning*. PhD thesis, School of Arts and Sciences Dissertation Advances in Privacy-Preserving, 2019.

[77] Kiran K Thekumparampil, Ashish Khetan, Zinan Lin, and Sewoong Oh. Robustness of conditional gans to noisy labels. In *Advances in neural information processing systems*, pages 10271–10282, 2018.

[78] John W Tukey. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians, Vancouver, 1975*, volume 2, pages 523–531, 1975.

[79] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning from noisy large-scale datasets with minimal supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 839–847, 2017.

[80] Sameer Wagh, Xi He, Ashwin Machanavajjhala, and Prateek Mittal. Dp - cryptography: Marrying differential privacy and cryptography in emerging applications, 2020.

[81] Christopher Waites. Pyvacy: Towards practical differential privacy for deep learning. *https://github.com/ChrisWaites/pyvacy*, 2019.

[82] Jinliang Wang and Anna W Santure. Parentage and sibship inference from multilocus genotype data under polygamy. *Genetics*, 181(4):1579–1594, 2009.

[83] Oliver Williams and Frank Mcsherry. Probabilistic inference and differential privacy. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2451–2459. Curran Associates, Inc., 2010.

[84] Ke Yan, Xiaosong Wang, Le Lu, and Ronald M Summers. Deeplesion: automated mining of large-scale lesion annotations and universal lesion detection with deep learning. *Journal of medical imaging*, 5(3):036501, 2018.

[85] Weihe Zhang, Yali Wang, and Yu Qiao. Metacleaner: Learning to hallucinate clean representations for noisy-labeled visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7373–7382, 2019.

[86] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review*, 22(3):177–210, 2004.

APPENDICES

Appendix A

Adversarial Additive Attacks on ENRON

In practice, especially for medical and security settings, the outliers in the training set might not be highly correlated and have more complex internal structures that might be difficult to model. This leads us to ask a conceptual question of whether DIFFINDO would be capable of removing outliers that have been specially curated by an adversary?

Inspired from [13], we model an attack on the ENRON dataset. Instead of flipping existing points in the dataset, we add some extra adversarially generated data points to the training set. These attacks are generated by influence functions and solving optimization problems to find appropriate points to fool the model [37]. The attacks are simulated by adding $0.5, 1, 1.5, 2, 3\%$ generated points. In the non-private scenario, these attacks have shown a test error increase from 3% to 24% by just adding 3% attack points. Here we show the experimental results of running adversarial additive attacks on the ENRON dataset privately. These attacks have shown a significant decrease in the testing accuracy by adding a modest 0.5% - 3% (Δ) adversarial points to the training set. At $\Delta = 0.5$, the worst performance of our method against all attacks was 8%, and at $\Delta = 3$, the error is relatively large at 17%. To investigate, we run 48 different files for varying Δ from 0.5 - 3 and run DIFFINDO to remove these points while measuring the testing accuracy for each run. The parameters used for this experiment are the same as the label flipping experiment for ENRON.

In Figure A.1 and Figure A.2, we show histograms for accuracy increase vs total number of attacks each value of Δ for DIFFINDO ($\epsilon = 4.7$, 50 epochs) vs DPSGD ($\epsilon = 4.7$, 64 epochs) and DIFFINDO ($\epsilon = \infty$, 50 epochs) vs SGD respectivelty. DIFFINDO is tuned to remove 2-2.x the outlier points in every run.

For each value of Δ in $[0.5, 1, 1.5, 2, 3]$, we calculate the change in testing accuracy for using DIFFINDO over the baselines. The x-axis shows the accuracy increase or decrease, and the y-axis shows the total no. of attacks in that range. We see that the PDF for each histogram is shifted to the positive side, which shows that majority of the attacks had accuracy gains when DIFFINDO was used over baselines.

	DPSGD vs DIFFINDO (Accuracy Increase %)			**SGD vs DIFFINDO ($\epsilon = \infty$) (Accuracy Increase %)**		
Δ (Outliers)	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**
0.5 (21)	8.37	1.89	-3.15	8	2.31	-1
1 (41)	9.93	1.56	-3.15	6	2.45	-1
1.5 (62)	7.62	0.96	-3.87	8	1.85	-1
2 (83)	7.62	0.40	-5.53	12	0.9	-5
3 (122)	9.21	0.28	-4.43	12	0.7	-7

Table A.1: Summary of DIFFINDO vs baselines on adversarial attacks

We detail the max, average and min accuracy increases over all 48 runs for each value of Δ for both the comparisons in Table A.1. The maximum accuracy is observed when DIFFINDO can remove most outlier points while the minimum accuracy is observed when none of the points was removed. However in the average case, DIFFINDO shows a positive trend for all values of Δ.

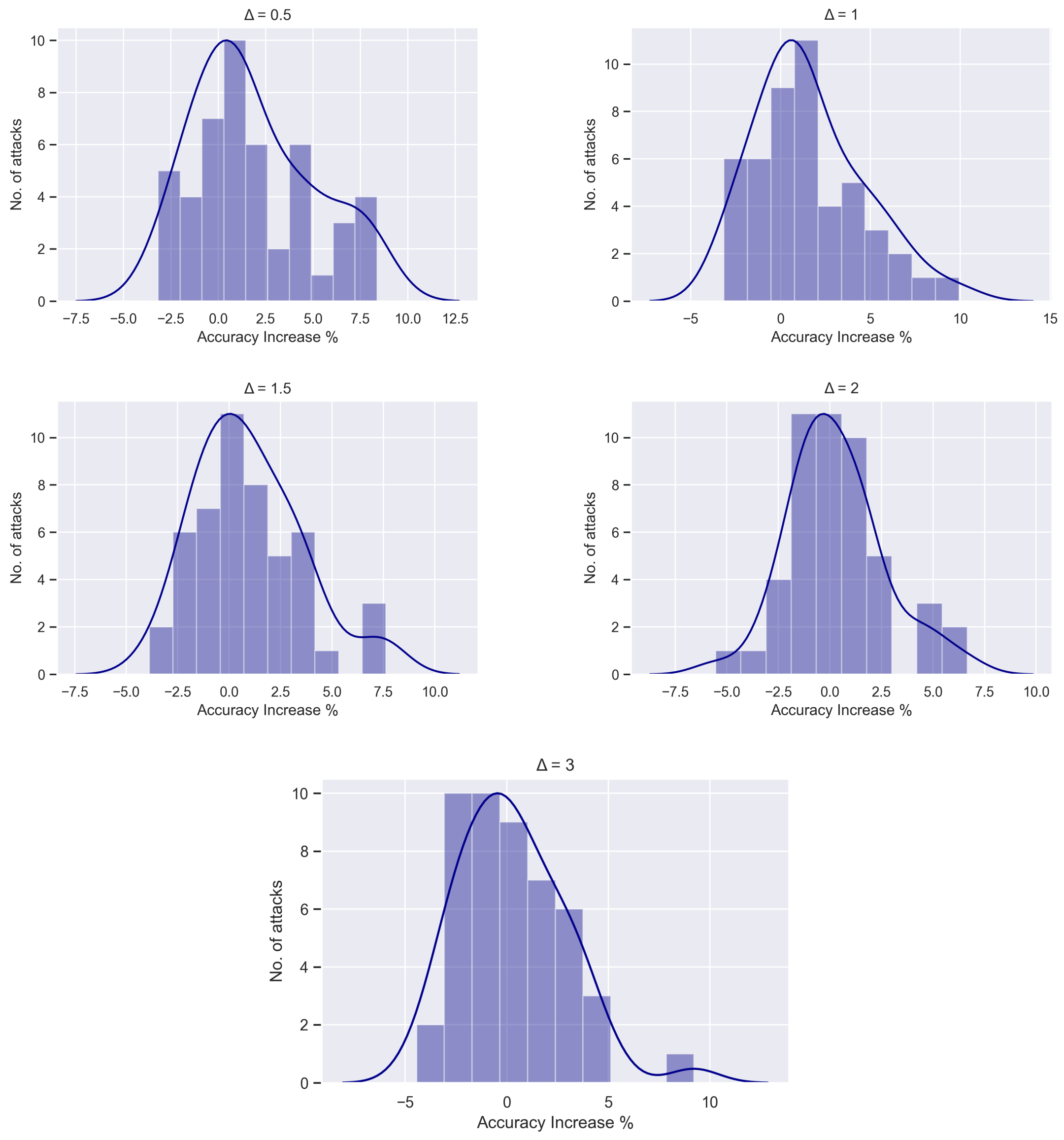

Figure A.1: DIFFINDO shows testing accuracy gains for the same privacy budget vs DPSGD over 48 attacks for each Δ

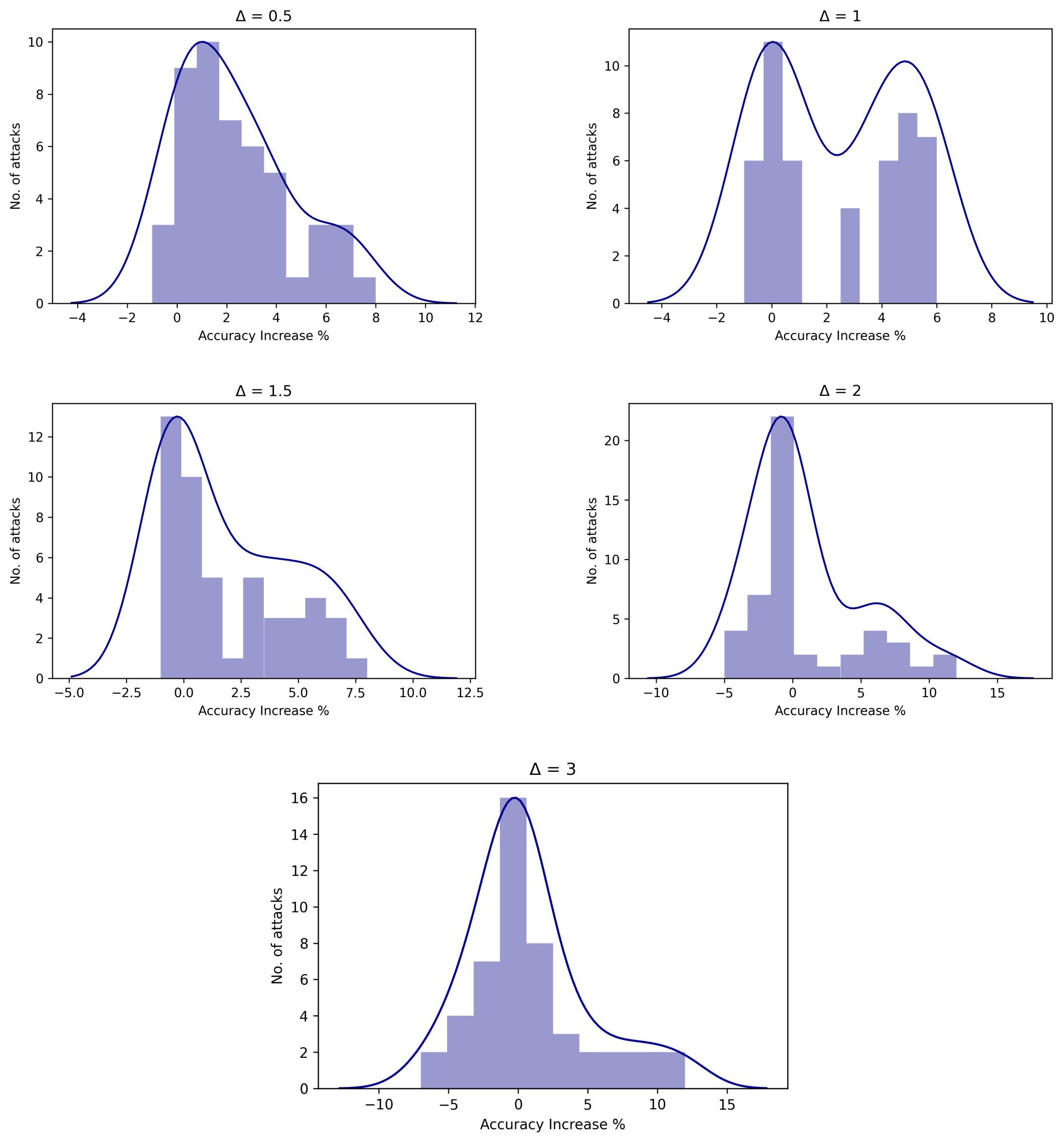

Figure A.2: DIFFINDO ($\epsilon = \infty$) shows testing accuracy gains vs SGD over 48 attacks for each Δ