CY 7790 Lecture 17: Differentially private SGD and auditing DP-SGD

Apra Gupta and Lisa Oakley

November 18, 2021

Differential Privacy Background Differential privacy is a topic that sets out to provide formal specifications and provable guarantees on privacy of a trained machine learning model. This field of study seeks to mitigate attacks which show that models trained on "anonymized" data still leak identifiable information about members of the training data set and result in breaches of privacy [MSS16, GMG+13]. There are two main formal defintions for differential privacy.

Definition 1 (ε-Differential Privacy)**.** A randomized algorithm A is ε-differentially private if, for every pair of neighboring data sets $D,\ D'$,

$$Pr(A(D) \in E) \leq e^{\varepsilon} \cdot Pr(A(D') \in E) \tag{1}$$

where neighboring data sets are defined by two data sets which differ (by addition, removal, or modification) in only one record. In other words, if two data sets differ in only one point, the distribution of the model on the two data sets should vary only a small amount. The lower ε, the less the model's distribution can vary over the adjacent sets resulting in more privacy.

This first definition is restrictive in practice, therefore a more relaxed definition introduces an additional parameter δ. A model which is (ε, δ)-differentially privacy is ε-differentially private with probability $1 - \delta$. Formally,

Definition 2 ((ε, δ)-Differential Privacy)**.** A randomized algorithm A is (ε, δ)-differentially private if, for every pair of neighboring data sets $D,\ D'$,

$$Pr(A(D) \in E) \leq e^{\varepsilon} \cdot Pr(A(D') \in E) + \delta \tag{2}$$

Papers Overview The papers presented in today's class focus on developing methods to find upper and lower bounds on privacy loss of a model. These bounds can be described by parameters $\varepsilon,\ \delta$.

1. The first paper (Abadi et al. *Deep Learning with Differential Privacy*) develops *upper bounds* on $\varepsilon,\ \delta$ by strategically adding noise to the gradient updates at every step of stochastic gradient descent. The use of the "moments accountant" allows for a tighter upper bound on privacy loss than in previous work.

2. The second paper (Jagielski et al. *Auditing Differentially Private Machine Learning: How Private is Private SGD?*) develops *lower bounds* on ε using a novel framework that looks at the distinguish-ability of poisoned points in a backdoored model using a novel clipping-aware backdoor attack.

1 Abadi et al. Deep Learning with Differential Privacy

Presented by John Abascal, Scribed by Lisa Oakley

Problem Statement. In this paper, the authors develop a new technique for training a deep neural network with differential privacy guarantees. When training a differentially private algorithm, it is typically the case that the target privacy parameters ε, δ are fixed ahead of time. The algorithm designer must achieve their best accuracy while maintaining an appropriate upper bound on privacy loss with respect to the fixed ε, δ. This upper bound can be computed during training by aggregating worst-case privacy loss at every step of the algorithm. However, if this computation and aggregation of loss upper bound is too loose, i.e. much more than the actual worst-case loss, the algorithm will add excess noise to the training procedure to compensate. This can result in decreased accuracy. The problem statement for this paper can be distilled to: **given a target bound (parameterized by ε, δ) for which (ε, δ)-differential privacy holds, the authors seek to reduce the amount of noise added during training, improving the accuracy of the model while maintaining privacy loss guarantees**.

Methodology. The solution proposed in the paper has two main facets: training with targeted noise and computing the bound on privacy loss using the moments accountant. The authors claim that this solution uses less noise at each step and has a more accurate bound on privacy loss, resulting in an algorithm with improved accuracy on the original task while maintaining the same privacy guarantees, as compared to prior work.

Training with Targeted Noise. The first facet of the solution is developing the differentially private algorithm which adds noise proportional to the impact of the record on the gradient. In other words, noise is added in a targeted way to meet the privacy guarantees while limiting the use excess noise which may negatively impact accuracy. The algorithm is composed of four key steps:

1. Computing the gradient of a random sample of points from the data set.
2. Clipping the gradient
3. Adding noise to the clipped gradient, calibrated to the norm
4. Update theta using a gradient update with the clipped, noisy gradients

The main source of privacy comes from the third step of the algorithm, where noise is added to the gradient. Noise is added proportionally to the norm of the clipped gradient, allowing for less noise to be used overall. Clipping (step 2) is used to bound the norm and thereby limit how much noise is added at any one iteration.

Tighter Bounds on Privacy Loss. The second facet of the solution is to improve the aggregation of the upper bound on worst-case privacy loss. Tightening the upper bound to be closer to the actual worst-case loss is beneficial as the algorithm is able to train for longer before exceeding the target bounds on privacy loss. There are three main properties of (ε, δ)-differential privacy when considering accumulation of privacy loss: composition, advanced composition, and amplification. These methods can be used to compute a cumulative upper bound on privacy loss, however the computed bound may still be much larger than the true worst-case privacy loss.

In this paper, the authors introduce the "moments accountant", a method of computing the aggregated upper bound on privacy loss. Here, the authors treat privacy loss as a random variable, and use higher order concentration inequalities (extensions of the Markov and Chebyshev inequalities) to bound privacy loss.

Evaluation. In Figure 1 (Figure 2 in the original paper), Abadi et al. demonstrate that, for the same algorithm, the moments accountant is able to compute a significantly tighter upper bound, ε, on privacy loss as compared to simply using the strong composition property of (ε, δ)-differential privacy. In Figure 2 (Figure 3 in the original paper), Abadi et al. demonstrate that for a feed forward neural network trained for the simple MNIST classification task, the accuracy remains high for relatively small ε, δ values. However, in Figure3 (Figure 6 in the original paper), there is a significant decrease in accuracy for the slightly more complex convolutional neural network trained for the CIFAR-10 image classification task when training privately. The authors leave solving this issue to future work, though this may well be an inherent issue with differential privacy.

Class Discussion.

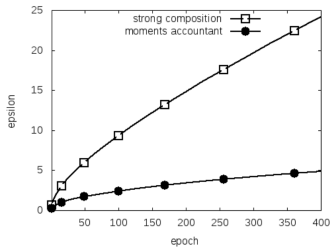
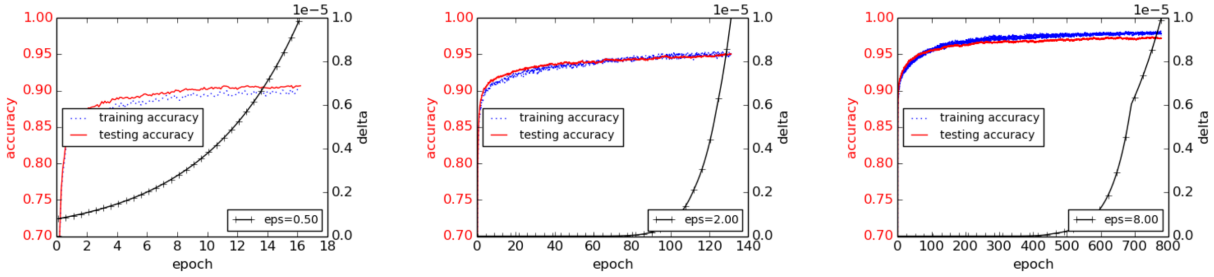


Figure 2: The ε value as a function of epoch E for $q = 0.01$, $\sigma = 4$, $\delta = 10^{-5}$, using the strong composition theorem and the moments accountant respectively.

Figure 1: Figure 2 from Abadi et al.



(1) Large noise (2) Medium noise (3) Small noise

Figure 3: Results on the accuracy for different noise levels on the MNIST dataset. In all the experiments, the network uses 60 dimension PCA projection, 1,000 hidden units, and is trained using lot size 600 and clipping threshold 4. The noise levels (σ, σ_p) for training the neural network and for PCA projection are set at (8, 16), (4, 7), and (2, 4), respectively, for the three experiments.

Figure 2: Figure 3 from Abadi et al.

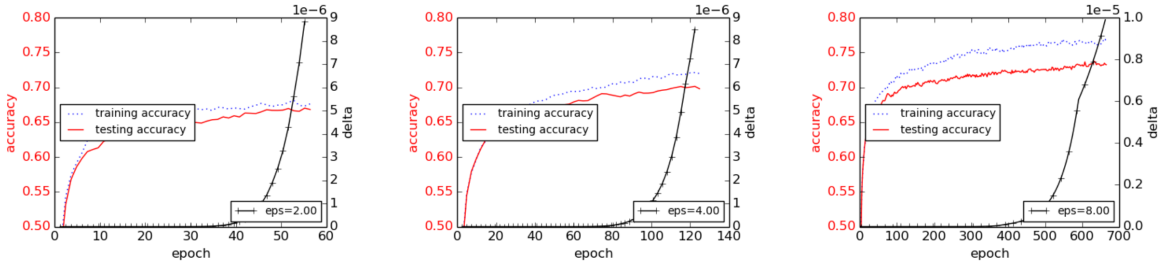
Choosing privacy parameters (ε, δ). Privacy parameter ε is predetermined (perhaps by a company, law, etc.). Then, the other hyperparameters are tuned to get the desired ε (at the sacrifice of accuracy). Usually fix δ to a small value. Can't say minimize ε because you can't get to zero.

Benefit of moments accountant. Moments accountant is keeping track/accumulating ε. A similar accountant can be implemented with any accumulation (e.g. standard and strong composition). Moments accountant is not necessarily the tightest bound — in fact recent work shows better bounds.

Privacy/accuracy tradeoff. Smaller ε means smaller allowed privacy loss, meaning you have to add more noise. More noise means at the gradient descent might not go in the correct direction. The more noise, the more likely you are to go in the wrong direction which leads to lower accuracy. 3 approaches to making differentially private algorithms:

1. take input data, add noise (maybe to objective), standard training
2. do standard training, add noise at end (like randomized smoothing)
3. look at internals of training and decide where and how to add noise to get better bounds

1,2 have been used for simpler things like logistic regression, however it is harder for deep neural networks as these result in a very loose bound. There is a whole line of research just on making ε smaller and smaller.



(1) $\varepsilon = 2$ (2) $\varepsilon = 4$ (3) $\varepsilon = 8$

Figure 6: Results on accuracy for different noise levels on CIFAR-10. With δ set to 10^{-5}, we achieve accuracy 67%, 70%, and 73%, with ε being 2, 4, and 8, respectively. The first graph uses a lot size of 2,000, (2) and (3) use a lot size of 4,000. In all cases, σ is set to 6, and clipping is set to 3.

Figure 3: Figure 6 from Abadi et al.

Role of sensitivity. We want to be able to compose mechanisms in a way that we know the maximum perturbation that we are adding. Don't want to add when we don't have to. Consider f which releases mean of dataset. We don't want to see a lot of difference in two means if record x is in dataset or not. Add noise proportional to the maximum difference in the means in order to compensate for this. Sensitivity is a function of how smooth the function is.

Gradient clipping. Clipping of the gradients is does not provide privacy, but it allows us to bound individual contribution of each gradient to the total. This is about the composition (accountant) rather than the privacy.

Clipping parameter (C). Authors choose median of gradients for C. If it is too large or too small, it affects the accuracy/privacy tradeoff. The parameter C doesn't show up explicitly in the asymptotics. C and σ provide the sensitivity which dictates the ε per round so it may be encoded in ε and δ. However, it should explicitly be present and future work does consider this (like DP auditing). This analysis does not seem to use it directly, the accuracy and training time definitely are affected by the clipping parameter C.

Computational overhead of adding noise and moments accountant. Must do numerical integration every time you want to bound using the nth moment. More practical to only look at a subset of the moments. At the time Tensorflow was not able to compute gradients in the convolutional layers (this has probably since been solved) — this might be why the authors use pre-trained CNNs.

Number of training epochs. You can train for longer, use mini-batching, but must have to choose other parameters (like q) to make up for it.

Gaussian vs. Laplace mechanism. The paper uses Gaussian mechanism, builds on prior work that also uses the Gaussian mechanism. This paper finds better bounds, improves on analysis from prior work. Work in differential privacy has shifted from Laplace to Gaussian mechanism for the sake of analysis. Gaussian mechanism satisfies only (ε, δ)-differential privacy, not ε-differential privacy. Laplace mechanism satisfies both.

2 Jagielski et al. Auditing Differentially Private Machine Learning: How Private is Private SGD?

Presented by Harsh Chaudhari, Scribed by Apra Gupta

Problem Statement. In this paper, authors investigate the extent to which Deferentially Private Stochastic Gradient Descent (shown in figure 4) empirically provides better privacy in practice than what its theoretical analysis (in prior work) suggests. They do so using a novel approach based on Backdoor Data Poisoning Attacks.

Algorithm 1: DP-SGD

Data: Input: Clipping norm C, noise magnitude σ, iteration count T, batch size b, dataset D, initial model parameters θ_0, learning rate η

For $i \in [T]$
- $G = 0$
- **For** $(x, y) \in$ *batch of* b *random elements of* D
  - $g = \nabla_\theta \ell(\theta_i; (x, y))$
  - $G = G + b^{-1} g \cdot \min(1, C||g||_2^{-1})$
- $\theta_i = \theta_{i-1} - \eta(G + \mathcal{N}(0, (C\sigma)^2 \mathbb{I}))$

Return θ_T

Figure 4: Differentially Private Stochastic Gradient Descent

Threat Model: The Differential Privacy framework aims to limit the amount of information an adversary can extract about the training data of an algorithm. Figure 5 shows the definition of differential privacy and subsequent Lemma for group privacy guarantees considered by this paper. In backdoor data poisoning attacks, which is what the author's use to empirically investigate differential privacy guarantees, the **Adversarial Goal** is to perturb parts of the training data so as to maintain model performance on natural data while changing the predicted class of perturbed data. To test out worst case privacy guarantees, when administering the backdoor attack, authors consider **Adversarial Knowledge** to be white box. I.e, adversaries have full knowledge of the model, it's architecture, parameters, loss function, training algorithm etc- as well as the training data. **Adversarial Capabilities** for the backdoor attacks are that adversaries can add small perturbations to the training data and then conduct model inference attacks on the trained model.

Definition 2.1 ([DMNS06])**.** An algorithm $\mathcal{A} : \mathcal{D} \mapsto \mathcal{R}$ is (ε, δ)-*differentially private* if for any two datasets D_0, D_1 which differ on at most one row, and every set of outputs $\mathcal{O} \subseteq \mathcal{R}$:

$$\Pr[\mathcal{A}(D_0) \in \mathcal{O}] \leq e^{\varepsilon} \Pr[\mathcal{A}(D_1) \in \mathcal{O}] + \delta, \tag{1}$$

where the probabilities are taken only over the randomness of $\mathcal{A}$.

Lemma 1 (Group Privacy)**.** Let D_0, D_1 be two datasets differing on at most k rows, $\mathcal{A}$ is an (ε, δ)-differentially private algorithm, and $\mathcal{O}$ an arbitrary output set. Then

$$\Pr[\mathcal{A}(D_0) \in \mathcal{O}] \leq e^{k\varepsilon} \Pr[\mathcal{A}(D_1) \in \mathcal{O}] + \tfrac{e^{k\varepsilon}-1}{e^{\varepsilon}-1} \cdot \delta. \tag{2}$$

Group privacy will give guarantees for poisoning attacks that introduce multiple points.

Figure 5: Differential Privacy Definition and Group Privacy Lemma

Methodology Algorithms with certain Differential Privacy guarantees must guarantee a certain upper bound (ϵ) on the difference between model outputs on any two datasets that differ on at most k rows, as seen in figure 5. If we can create two datasets differing in at most k rows where the difference in the outputs is high, that empirically gives us a lower bound on ϵ. I.e, the model's differential privacy guarantee cannot be stronger that ϵ. Through this intuition, the authors propose using backdoor attacks paired with a test to classify points as poisoned or not using model loss on poisoned label to statistically lower bound ϵ. Their proposed framework for empirical analysis of practical Differential Privacy Guarantees is as follows:

Algorithm 2: Empirically Measuring ε

Data: Algorithm $\mathcal{A}$, datasets D_0, D_1 at distance k, output set $\mathcal{O}$, trial count T, confidence level α

$\text{ct}_0 = 0, \text{ct}_1 = 0$

For $i \in [T]$

  If $\mathcal{A}(D_0) \in \mathcal{O}$ $\text{ct}_0 = \text{ct}_0 + 1$

  If $\mathcal{A}(D_1) \in \mathcal{O}$ $\text{ct}_1 = \text{ct}_1 + 1$

$\hat{p}_0 = \textsc{ClopperPearsonLower}(\text{ct}_0, T, \alpha/2)$

$\hat{p}_1 = \textsc{ClopperPearsonUpper}(\text{ct}_1, T, \alpha/2)$

Return $\varepsilon_{LB} = \ln(\hat{p}_0/\hat{p}_1)/k$

Figure 6: Algorithm to estimate p_0 and p_1

Theorem 2. When provided with black box access to an algorithm $\mathcal{A}$, two datasets D_0 and D_1 differing on at most k rows, an output set $\mathcal{O}$, a trial number T and statistical confidence α, if Algorithm 2 returns ε_{LB}, then, with probability $1 - \alpha$, $\mathcal{A}$ does not satisfy ε'-DP for any $\varepsilon' < \varepsilon_{LB}$.

Proof of Theorem 2. First, the guarantee of the Clopper-Pearson confidence intervals is that, with probability at least $1 - \alpha$, $\hat{p}_0 \leq p_0$ and $\hat{p}_1 \geq p_1$, which implies $p_0/p_1 \geq \hat{p}_0/\hat{p}_1$. Second, if $\mathcal{A}$ is ε-DP, then by group privacy we would have $p_0/p_1 \leq \exp(k\varepsilon)$, meaning $\mathcal{A}$ is *not* ε'-DP for any $\varepsilon' < \frac{1}{k}\ln(p_0/p_1)$. Combining the two statements, $\mathcal{A}$ is *not* ε' for any $\varepsilon' < \frac{1}{k}\ln(\hat{p}_0/\hat{p}_1) = \varepsilon_{LB}$. □

Figure 7: Proof of confidence interval on ϵ_{lb} using 6

1. First, construct two datasets D_0 and D_1 as follows: D_1 contains k rows of perturbed data whose labels are changed to a target label (we will later consider the exact poisoning methodology)

2. We then train the model on on the poisoned dataset D_1.

3. Construct the output set O as the impact of poisoning. It represents whether the backdoored points are distinguishable by a threshold on their loss, the exact method of used to conduct the backdoor test will be detailed later.

4. We then wish to estimate the probabilities $p_o = Pr[A(D_0) \in O]$ and $p_1 = Pr[A(D_1) \in O]$ (where O is the output set of the algorithm), so we can construct a lower bound ϵ_{lb} on ϵ such that Lemma 2 from figure 5 holds.With $\delta = 0$ this simplifies to $\epsilon_{lb} = ln(p_0/p_1)$. Author's rely on Monte Carlo estimation and Clopper Pearson confidence intervals to upper bound p_0 and lower bound p_1 with confidence $1 - \alpha/2$ each so that the estimate of ϵ_{lb} holds with confidence $> 1 - \alpha$. This algorithm and the subsequent proof of the confidence on ϵ_{lb} is detailed in 6 and 7 respectively. T trials of this algorithm are run on each attack to estimate p_0 and p_1 using the the respective backdoor tests (BackdoorTest and ClipBKDTest for each attack (detailed below)).

5. Now all that remains is to detail the poisoning attacks as well as the backdoor test used to construct the output set O. Authors experiment with two backdoor poisoning attacks: a basic baseline attack and a clipping aware attack. These two attacks and their corresponding tests along with the shortcomings of the basic backdoor attack are detailed below:

Algorithm 3: Baseline Backdoor Poisoning Attack and Test Statistic (Section 3.1)

Data: Dataset X, Y, poison size k, perturbation function $Pert$, target class y_p

Function BACKDOOR$(X, Y, k, Pert, y_p)$**:**

$X_p = \text{GETRANDOMROWS}(X, k)$

$X'_p = Pert(X_p)$

$X^p_{tr} = \text{REPLACERANDOMROWS}(X, X'_p)$

$Y^p_{tr} = \text{REPLACERANDOMROWS}(Y, y_p)$

return $D_0 = (X, Y), D_1 = (X^p_{tr}, Y^p_{tr})$

Data: Model f, dataset (X, Y), pert. function $Pert$, target class y_p, loss function ℓ, threshold Z

Function BACKDOORTEST$(f, X, Y, Pert, y_p, \ell, Z)$**:**

$X_p = Pert(X)$

If $\sum_{x_p \in X_p} \ell(f; (x_p, y_p)) > Z$ **Return** Backdoored

Return Not Backdoored

Figure 8: Basic backdoor poisoning attack and test

1. *Basic Backdoor Poisoning Attack(Backdoor):* In this simple attack described in figure 8, a small perturbation is added to the k poisoned data-points and the adversary changes the predicted class of the poisoned data before feeding it into the model. The perturbation function adds a pattern in the corner of an image. After the model is trained, the backdoor test considered is simply a threshold on the total loss incurred by the dataset, i.e, if the sum of losses across the poisoned dataset is above a certain threshold Z, BackdoorTest returns True. (see figure 8) for details.

   The DP-SGD training algorithm introduces the clipping of gradients into the learning process of a model.Many poisoning attacks perform significantly worse in the presence of clipping. As we will see in the evaluation section, the basic backdoor attack performs very poorly with a weak lower bound. Standard poisoning attacks work by increasing the model loss on poisoned points and labels thereby increasing the size of the gradient update in the direction of the poisoned data. However, in the presence of clipping, this relationship is broken because the size of the gradient update is always restricted (clipped). The clipping aware attack detailed below attempts to solve this problem.

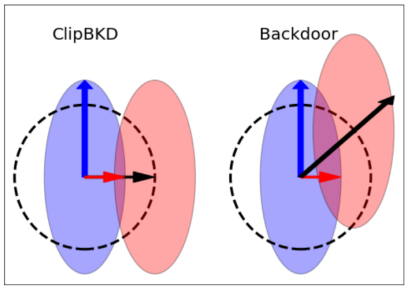


Figure 1: The distribution of gradients from an iteration of DP-SGD under a clean dataset (blue ellipse) and a poisoned dataset (red ellipse). The right pair depicts traditional backdoors while the left pair depicts our backdoors. Our attack pushes in the direction of least variance, so is impacted less by gradient clipping, which is indicated by the two distributions overlapping less.

Figure 9: Demonstration of clipping-aware attack's effects on the gradients

2. *Clipping-Aware Backdoor Poisoning Attack (ClipBKD):*To be more effective in the presence of clipping, the attack must produce not only large gradients, but distinguishable gradients. That is, the distribution of gradients arising from poisoned and cleaned data must be significantly different. To do this, the clipping aware attack attempts to choose poisoned points that minimize the variance between the benign and poisoned points, so the attack pushes in the direction of least variance, so is impacted less by gradient clipping as demonstrated in figure 9. This variance is minimized by using singular value decomposition: the poisoned point x_p is the singular vector corresponding to the smallest singular value (i.e. the direction of least variance), scaled to a similar norm as the rest of the dataset. The poisoned label y_p is the smallest probability class on x_p The exact algorithm is shown in figure 10. Then, k copies of the poisoned point and label are inserted into the training dataset. After the model is trained, the backdoor test considered (ClipBKDTest) is slightly different from the one considered for the simple backdoor attack, it looks at the difference between the model's output on poisoned points $f(x_p)$ and 0^d multiplied by the target class y_p (see figure 10 for the exact test)

Algorithm 4: Clipping-Aware Backdoor Poisoning Attack and Test Statistic (Section 3.2)

Data: Dataset X, Y, pretrained model f, poison size k, dataset dimension d, norm m
Function ClipBkd(X, Y, k, f, m)**:**
  $U, D, V = SVD(X)$ ▷ Singular value decomposition
  $x_p = mV_d$ ▷ V_d is the singular vector for smallest singular value
  $y_p = \arg\min_i f(x_p)$ ▷ Pick class maximizing gradient norm
  $X_{tr}^p = \text{ReplaceRandomRows}(X, [x_p] * k)$ ▷ Add poisoning point k times
  $Y_{tr}^p = \text{ReplaceRandomRows}(Y, [y_p] * k)$ ▷ Add targeted class k times
  return $D_0 = (X, Y), D_1 = (X_{tr}^p, Y_{tr}^p)$
Data: Model f, Poison Data x_p, y_p, Threshold Z
Function ClipBkdTest(f, x_p, y_p, Z)**:**
  If $(f(x_p) - f(0^d)) \cdot y_p > Z$ **Return** Backdoored
  Return Not Backdoored

Figure 10: Clipping aware backdoor poisoning attack and test

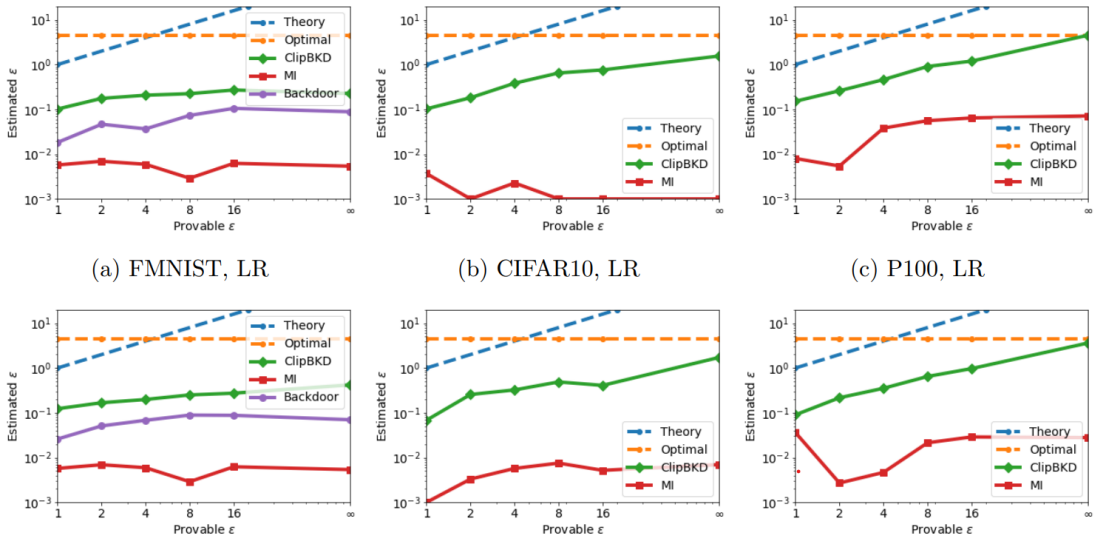
Dataset	Epochs	Learning Rate	Batch Size	ℓ_2 Regularization
FMNIST	24	0.15	250	0
CIFAR10	20	0.8	500	0
P100	100	2	250	10^{-4} / 10^{-5}

Table 1: Training details for experiments in Section 4. P100 regularization is 10^{-5} for logistic regression and 10^{-4} for neural networks, following [JE19].

Figure 11: Data and Hyperparameter setup in experiments

The proposed framework is evaluated using 3 different datasets: FMNIST, CIFAR10 and P100. For each dataset, both a logistic regression (LR) model and a two-layer feedforward neural network (FNN), trained with DP-SGD using various hyperparameters are considered. Figure 11 details the hyperparameters considered for each dataset.12 presents a direct comparison of the privacy bounds produced by the novel clipping-aware attack ($T = 500$ and $\alpha = 0.01$ using the the best result from $k = 1, 2, 4, 8$ poisoning points),the basic backdoor attack, and Membership Inference (using 1000 samples, and average over 10 trained models). The best theoretical upper bound on ϵ_{th} is shown on the x-axis ϵ_{OPT} of ϵ_{LB} produced using trials and confidence level the y-axis.For every dataset and model, they find that ClipBKD significantly outperforms MI, by a factor of 2.5x–1500x. They also find ClipBKD always improves over

basic backdoors: on FMNIST by an average factor of 3.84x, and standard backdoors never reach positive ϵ_{LB} on CIFAR, due to the large number of points required to poison the pretrained model. They also find that ClipBKD returns ϵ_{LB} that are close to ϵ_{th} ; for finite ϵ_{th} , the majority of gaps are a factor of ¡ 12.3x, reaching as low as 6.6x.



(a) FMNIST, LR (b) CIFAR10, LR (c) P100, LR

(d) FMNIST, FNN (e) CIFAR10, FNN (f) P100, FNN

Figure 2: Performance of privacy attacks MI, Backdoor, and ClipBKD on our datasets. LR = logistic regression, FNN = two-layer neural network. Backdoor attacks have not been developed for Purchase-100, so only MI and Clip-BKD were run. Backdoors do not provide positive ε_{LB} on CIFAR10 due to difficulty with the pretrained model.

Figure 12: Results of experimental evaluation

Authors also provide a more thorough evaluation of ClipBKD's performance as a function of DP-SGD's hyper-parameters.They vary clipping norm between 0.5, 1, and 2 and vary the noise to ensure ϵ_{th} between 1, 2, 4, 8, 16, and∞. They also vary the initialization randomness between random normal initializations with variance 0 (fixed initialization), $0.5\sigma, \sigma$, and 2σ, where σ is the variance of Glorot normal initialization. Figure 13 reports the best ϵ_{LB} produced by the attack over $k = 1, 2, 4, 8$. These experiments show that:

1. As ϵ_{th} (equivalently, the noise level decreases), ϵ_{LB} also increases.
2. As the initialization randomness decreases, ϵ_{LB} increases.
3. As clipping norm decreases, ϵ_{LB} decreases, except when the initialization is fixed.

Params	Fixed Init	Init Rand $= 0.5\sigma$	Init Rand $= \sigma$	Init Rand $= 2\sigma$
$\varepsilon_{th} = 1, \sigma_{GD} = 5.02$	0.13 / 0.15 / 0.13	0.13 / 0.17 / 0.13	0.06 / 0.12 / 0.09	0.01 / 0.06 / 0.08
$\varepsilon_{th} = 2, \sigma_{GD} = 2.68$	0.33 / 0.37 / 0.28	0.27 / 0.33 / 0.39	0.10 / 0.17 / 0.27	0.01 / 0.06 / 0.17
$\varepsilon_{th} = 4, \sigma_{GD} = 1.55$	0.89 / 0.75 / 0.71	0.28 / 0.52 / 0.78	0.08 / 0.20 / 0.54	0.02 / 0.10 / 0.18
$\varepsilon_{th} = 8, \sigma_{GD} = 1.01$	1.61 / 1.85 / 1.90	0.33 / 0.55 / 1.27	0.07 / 0.25 / 0.53	0.01 / 0.05 / 0.20
$\varepsilon_{th} = 16, \sigma_{GD} = 0.73$	2.15 / 2.16 / 2.43	0.36 / 0.80 / 1.39	0.13 / 0.27 / 0.72	0.02 / 0.08 / 0.16
$\varepsilon_{th} = \infty, \sigma_{GD} = 0$	4.54 / 4.54 / 4.54	0.29 / 0.95 / 2.36	0.10 / 0.42 / 0.79	0.03 / 0.09 / 0.27

Table 2: Lower bound ε_{LB} measured with ClipBKD for clipping norms of (0.5 / 1 / 2) for two-layer neural networks trained on FMNIST. Training accuracy for all models is 96%-98%. Results are the maximum over $k = 1, 2, 4, 8$. σ_{GD} refers to the DP-SGD noise multiplier, while σ is Glorot initialization randomness [GB10]. All reported values of ε_{LB} are valid with 99% confidence over the randomness of our experiments.

Figure 13: Results thorough evaluation of ClipBKD's performance as a function of DP-SGD's hyperparameters

Class Discussion.

Does the data have to be i.i.d? There is no notion of i.i.d considered/required here.

Why use Monte Carlo Sampling + Clopper Pearson Confidence Intervals? We need to use approximate methods such as Monte Carlo Sampling + Clopper Pearson Confidence Intervals because we do not know the true probability that a certain point belongs to either the poisoned or the clean data-set.

Why does the simple Badnets attack not work? In order for the simple badnets attack to work, the gradient of model parameters with respect to one poisoning point should be large, however this relationship is broken after clipping which is an important part of the DP-SGD algorithm.

Why do we test on loss rather than accuracy when testing whether a datapoint is poisoned? k (number of poisoning points) is low, because if we use too large of a k, there is not reasonable lower bound. Loss is more granular so you can get a threshold.

References

[GMG+13] Melissa Gymrek, Amy L McGuire, David Golan, Eran Halperin, and Yaniv Erlich. Identifying personal genomes by surname inference. *Science*, 339(6117):321–324, 2013.

[MSS16] Richard McPherson, Reza Shokri, and Vitaly Shmatikov. Defeating image obfuscation with deep learning. *arXiv preprint arXiv:1609.00408*, 2016.