CY 7790

Special Topics in Security and Privacy: Machine Learning Security and Privacy
Fall 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science

November 18, 2021

Deep Learning with Differential Privacy

Martín Abadi, Andy Chu, Ian GoodFellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, Li Zhang

John Abascal

Motivation for Differential Privacy

- **Anonymization is not enough!**
    - Netflix Prize
        - Contestants were given a dataset of user ratings where identifying information was anonymized and competed to create a recommendation engine
        - Researchers were able to de-anonymize the dataset by linking user ratings to public IMDB ratings
- Ideally, we need a way to learn general trends of the dataset without revealing any individual's private information/contribution

Intuition Behind Differential Privacy

- Suppose we have an algorithm/model, $\mathscr{M}$, which outputs the probability of a student, x, having an F in CY7790 ($\mathscr{M}(x) \in [0,1]$)

  - $\mathscr{M}$ is trained on dataset D, and when we make the query $\mathscr{M}_D(\text{John})$, the output is 0.55

  - $\mathscr{M}$ is trained on dataset D + John, and when we make the query $\mathscr{M}_{D+John}(\text{John})$, the output is 0.57. So it is unclear whether John is failing CY7790

[Mukul Rathi]

Intuition Behind Differential Privacy

- What if $\mathscr{M}_{D+John}$(John) outputs 0.80?
  - Then we have high confidence that John is failing CY7790
  - We get higher accuracy on our test set at the cost of privacy:

Privacy loss

$$log(\frac{\mathbb{P}[\mathscr{M}_{D+John}(John) = 1]}{\mathbb{P}[\mathscr{M}_D(John) = 1]}) \quad \Rightarrow \quad log(\frac{0.57}{0.55}) \approx 0.036 \quad\quad log(\frac{0.80}{0.55}) \approx 0.375$$

[Mukul Rathi]

How do we bound this privacy loss?

Differential Privacy

- Let D, D' be neighboring datasets ($\| D - D' \|_1 \leq 1$), E be some potential output of $\mathscr{M}$

- Then, using the definition of privacy loss from before:

$$log(\frac{\mathbb{P}[\mathscr{M}(D) \in E]}{\mathbb{P}[\mathscr{M}(D') \in E]}) \leq \epsilon$$

Privacy budget

(i.e. the maximum possible privacy loss)

$$\Rightarrow \frac{\mathbb{P}[\mathscr{M}(D) \in E]}{\mathbb{P}[\mathscr{M}(D') \in E]} \leq e^{\epsilon}$$

Differential Privacy

- Let D, D' be neighboring datasets ($\| D - D' \|_1 \leq 1$), E be some potential output of $\mathscr{M}$

- Then, using the definition of privacy loss from before:

<u>ϵ-Differential Privacy</u>

$$\mathbb{P}[\mathscr{M}(D) \in E] \leq e^{\epsilon} \cdot \mathbb{P}[\mathscr{M}(D') \in E]$$

Differential Privacy

$(\epsilon - \delta)$-Differential Privacy

$$\mathbb{P}[\mathscr{M}(D) \in E] \leq e^{\epsilon} \cdot \mathbb{P}[\mathscr{M}(D') \in E] + \boxed{\delta}$$

Failure probability

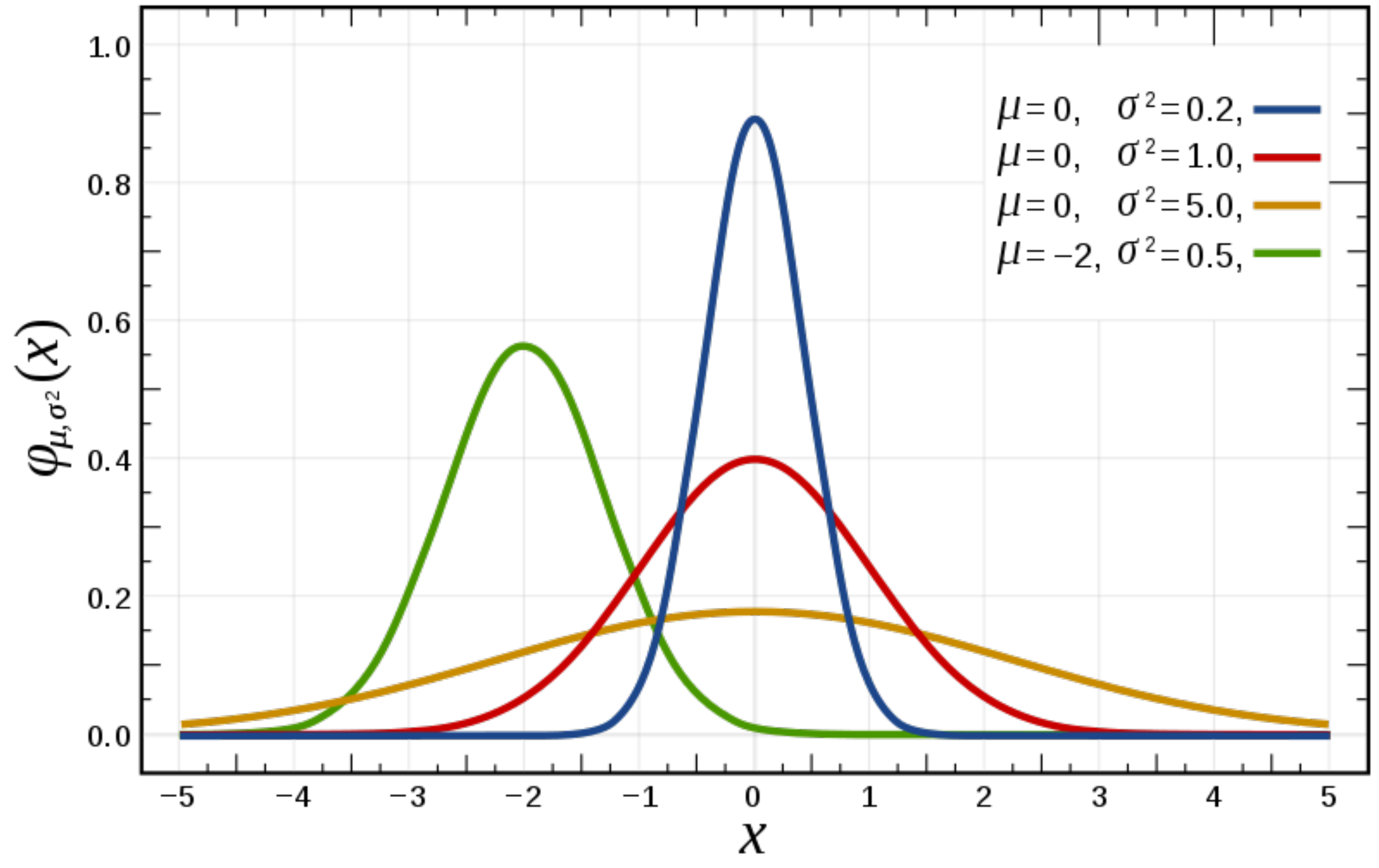
(i.e. our algorithm is ϵ-DP with probability $1 - \delta$)

Differential Privacy in Practice

- How do we achieve differential privacy?
  - Just add noise… intelligently!
  - **Sensitivity:** Given a function $f : D \to \mathbb{R}$, we can define the S_f as $|f(x) - f(x')|$ where x, x' are neighboring points
- **The Gaussian Mechanism:**

$$\mathscr{M}(x) \triangleq f(x) + \mathscr{N}(0,\ S_f^2 \cdot \sigma)$$

  - $(\epsilon - \delta)$-DP for $\delta \geq \frac{4}{5}\exp(-(\sigma\epsilon)^2/2), \quad \epsilon < 1$



Properties

- **Composition:** If we run k, $(\epsilon - \delta)$-DP algorithms:
  - The resulting algorithm will be $(k\epsilon - k\delta)$-DP
- **Advanced composition:** If $k < 1/\epsilon^2$ The resulting algorithm is $(O(\sqrt{klog(1/\delta')}) \cdot \epsilon - k\delta + \delta')$-DP for all $\delta' > 0$
- **Amplification:** If we sample a fraction, q, of the data, our algorithm becomes $(q\epsilon - q\delta)$-DP

DP-SGD Algorithm

Goal: Minimize privacy loss at each iteration.
(Minimizes composed privacy loss)

1. Compute gradient of random sample

2. Clip gradient (i.e. force max ℓ_2 norm of gradient to be C in order to "bound sensitivity of the gradient")

3. Add noise calibrated to the clipping norm, C, times the noise scale, σ

4. Update θ

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialize θ_0 randomly

for $t \in [T]$ **do**

Take a random sample L_t with sampling probability L/N

Compute gradient

For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i)/\max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Analysis

- **Naive Approach**
  - We can use basic composition to arrive at an upper bound for our privacy budget
    - $(T\epsilon - T\delta)$-DP
  - Additionally, we can apply amplification to get a tighter bound on this budget
    - $(Tq\epsilon - Tq\delta)$-DP

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialize θ_0 randomly

for $t \in [T]$ **do**

Take a random sample L_t with sampling probability L/N

Compute gradient

For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Analysis

- **Using Advanced Composition**
  - $(O(q\epsilon \cdot \sqrt{Tlog(1/\delta)}) - Tq\delta)$-DP
  - So, our privacy budget is a function of $\sqrt{Tlog(1/\delta)}$ instead of T

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialize θ_0 randomly

for $t \in [T]$ **do**

Take a random sample L_t with sampling probability L/N

Compute gradient

For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

Descent

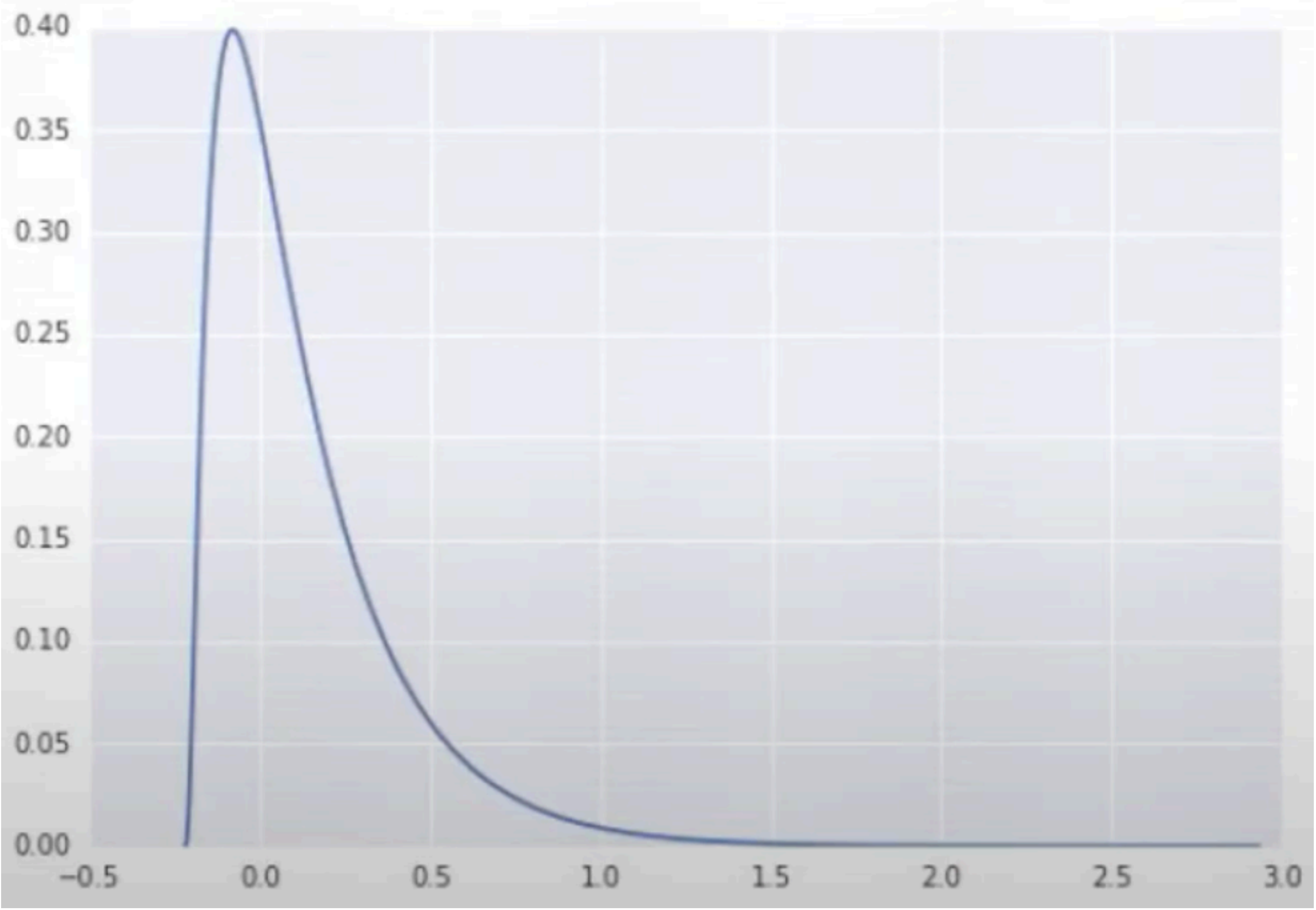
$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Does advanced composition give us the best bound?

Analysis

- **The Moments Accountant**
  - The authors decide to treat privacy loss as its own random variable
  - The privacy loss has a quick drop-off but has a very long tail
    - So, we can use **concentration inequalities** to bound privacy loss



log(privacy loss)

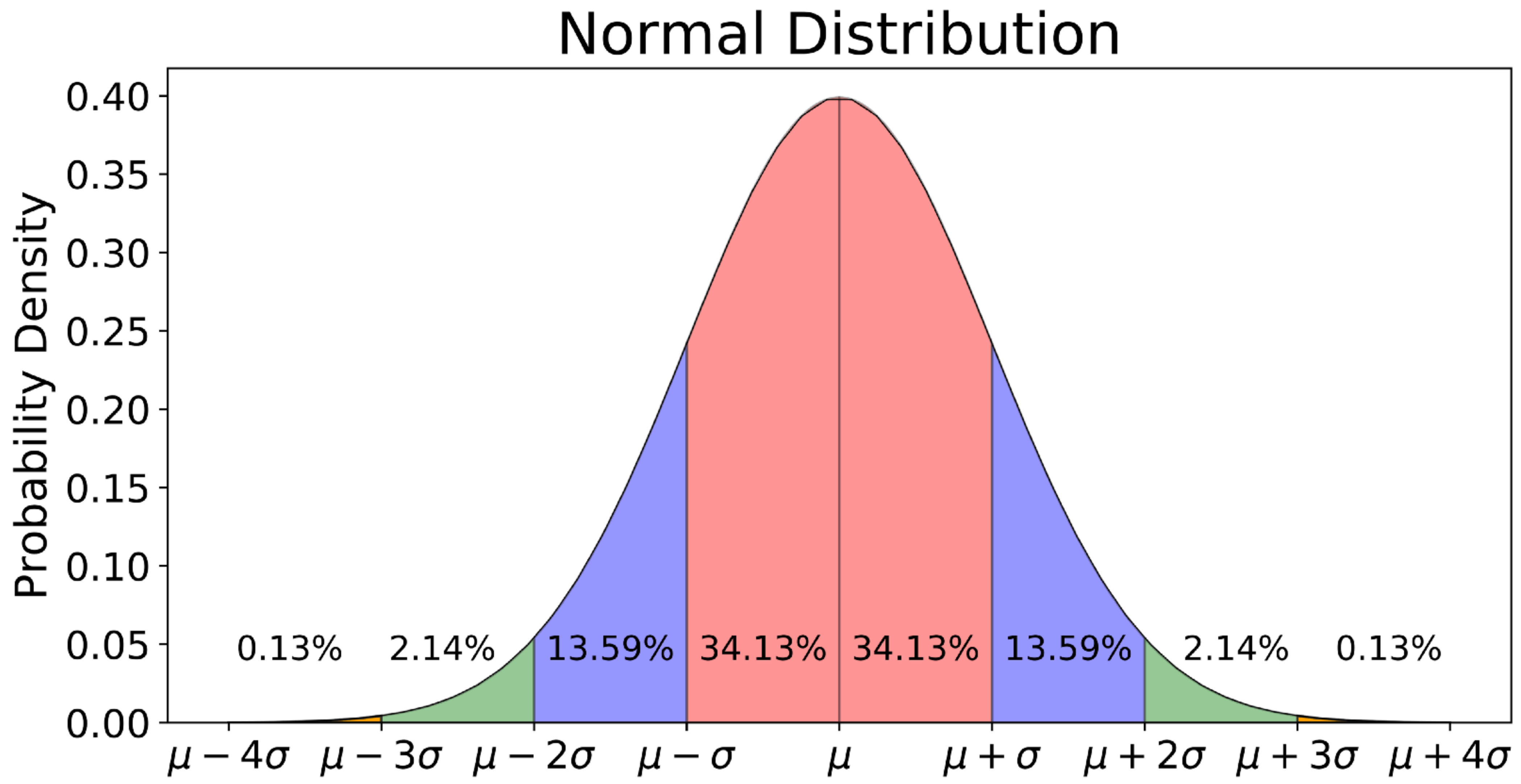
Analysis

- **Concentration Inequalities**
  - Markov's Inequality:

$$\mathbb{P}[X \geq a] \leq \frac{\mathbb{E}[X]}{a}$$

  - Chebyshev's Inequality:

$$\mathbb{P}[\,|X-\mu| \geq a] \leq \frac{\mathbb{E}[\,|X-\mu|^2]}{a^2} = \frac{var(X)}{a^2}$$



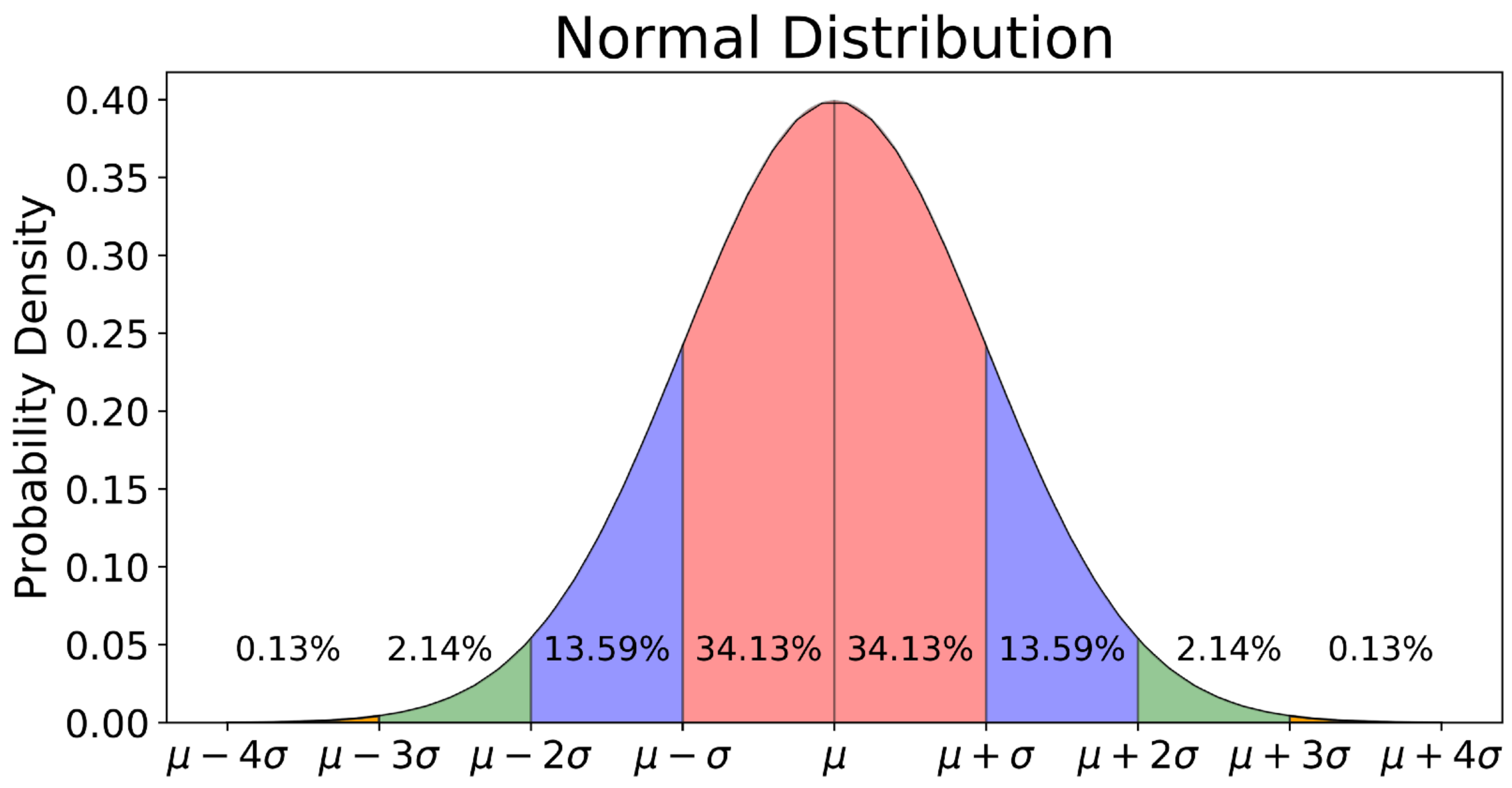
Analysis

- **Concentration Inequalities**
  - We can extend these inequalities to higher order moments using the moment generating function:

$$\mathbb{P}[\,|X-\mu|\geq a] \leq \frac{\mathbb{E}[\,|X-\mu|^k]}{a^k}$$

$$\downarrow$$

$$\mathbb{P}[(X-\mu)\geq a] = \mathbb{P}[e^{\lambda(X-\mu)} \geq a] \leq \frac{\mathbb{E}[e^{\lambda(X-\mu)}]}{e^{\lambda a}}$$



Analysis

- **Concentration Inequalities**
  - Then, we can optimize our choice of λ to obtain the tightest result
  - (For practical reasons, the authors only keep track of the first 32 moments)
- Using this approach, the authors find that the algorithm is $(q\epsilon\sqrt{T} - \delta)$-DP

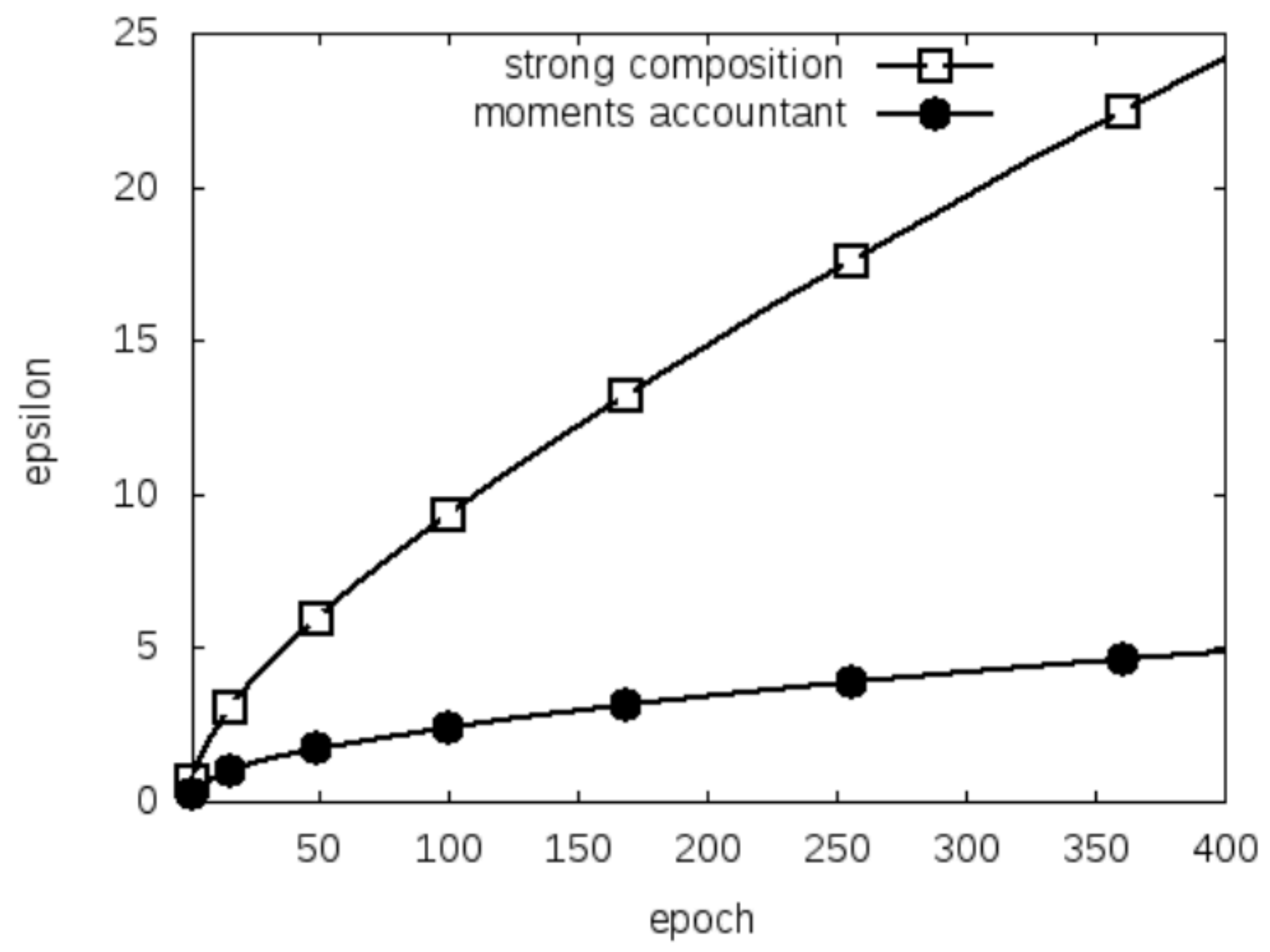


Figure 2: The ε value as a function of epoch E for $q = 0.01$, $\sigma = 4$, $\delta = 10^{-5}$, using the strong composition theorem and the moments accountant respectively.

Empirical Results

- Using a simple feed-forward neural network trained on MNIST the authors achieved the following results
  - No privacy $\rightarrow$ 98.3% accuracy
  - $\epsilon = 8,\ \delta = 10^{-5}$ $\rightarrow$ 97% accuracy (Compared to 80% in previous work)
  - $\epsilon = 2,\ \delta = 10^{-5}$ $\rightarrow$ 95% accuracy
  - $\epsilon = 0.5,\ \delta = 10^{-5}$ $\rightarrow$ 90% accuracy

Empirical Results

- Using a simple feed-forward neural network trained on CIFAR10 the authors achieved the following results
  - No privacy → 80% accuracy
  - $\epsilon = 8,\ \delta = 10^{-5}$ → 73% accuracy
  - $\epsilon = 2,\ \delta = 10^{-5}$ → 67% accuracy

Strengths

- The authors show that differentially private deep learning can achieve good accuracy with a reasonable amount of privacy leakage
- Moments accountant is a strong tool for computing privacy of iterative algorithms
  - It gives us more room for accuracy without the privacy loss found using advanced composition

Limitations

- A reasonable tradeoff between privacy and accuracy does not seem to hold for more complicated datasets, like CIFAR10
- The authors only consider a simple neural network in their experiments

Auditing Differentially Private ML: How Private is Private SGD

Matthew Jagielski, Jonathan Ullman, Alina Oprea

Harsh Chaudhari

Main Idea

- Investigate the extent to which DP-SGD does or doesn’t give better privacy in practice than what its theoretical counterparts suggest.

- How to Investigate? -> Use data poisoning attacks

Roadmap

- Background:
  - Differential Privacy
  - DP-SGD
  - Backdoor Attack
- Statistically Measuring Differential Privacy
- Poisoning Attacks
- Experiments

Differential Privacy

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.

DP-SGD

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

        $g = \nabla_\theta \ell(\theta_i; (x, y))$

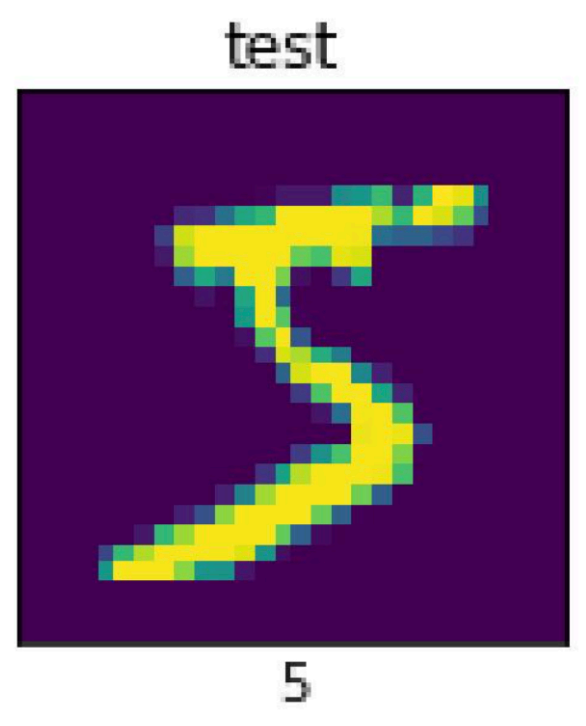
        $G = G + b^{-1} g \cdot \min(1, C||g||_2^{-1})$ ← Gradient Clipping

    $\theta_i = \theta_{i-1} - \eta(G + \mathcal{N}(0, (C\sigma)^2 \mathbb{I}))$ ← Adding Noise

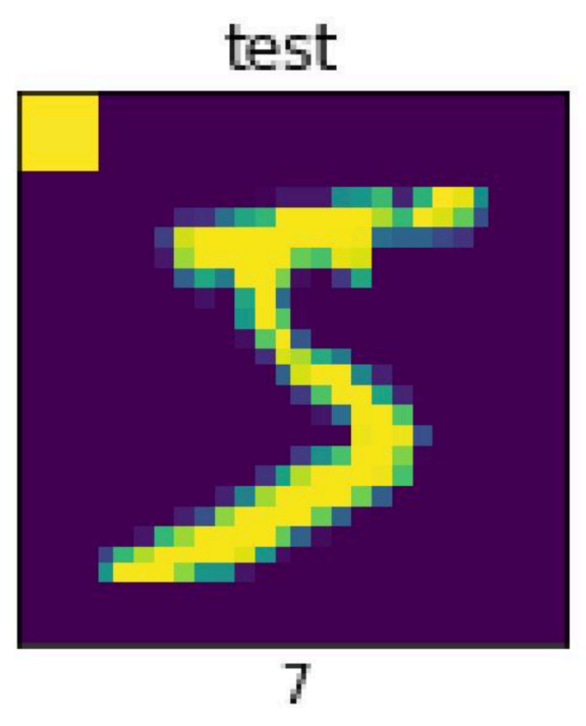
Return θ_T

Backdoor Attack

- Introduce a Backdoor pattern on images during training phase with the desired target label.



- Apply the backdoor pattern on test data and check if the test samples get misclassified to the target label.



Statistically Measuring Differential Privacy

- Given algorithm A, adjacent datasets D_0 and D_1 and outputs O:

$$\Pr[\mathcal{A}(D_0) \in \mathcal{O}] \leq e^{\varepsilon} \Pr[\mathcal{A}(D_1) \in \mathcal{O}]$$

- Begin by computing probabilities:

$$p_0 = \Pr[\mathcal{A}(D_0) \in \mathcal{O}] \qquad p_1 = \Pr[\mathcal{A}(D_1) \in \mathcal{O}]$$

- Lower bound on Epsilon:

$$\varepsilon_{LB} = \ln(p_0/p_1)$$

Statistically Measuring Differential Privacy

- To approximate p_0 and p_1, use Monte Carlo Estimation
- Clopper Pearson confidence intervals: Produce epsilon lower bound with probability 99%

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

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

For $i \in [T]$

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

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

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

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

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

Statistically Measuring Differential Privacy

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 $\varepsilon' < \frac{1}{k}\ln(\hat{p}_0/\hat{p}_1) = \varepsilon_{LB}$. $\square$

Poisoning Attack: Backdoor

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 $\textsc{Backdoor}(X, Y, k, Pert, y_p)$**:**

$X_p = \textsc{GetRandomRows}(X, k)$

$X_p' = Pert(X_p)$

$X_{tr}^p = \textsc{ReplaceRandomRows}(X, X_p')$

$Y_{tr}^p = \textsc{ReplaceRandomRows}(Y, y_p)$

return $D_0 = (X, Y), D_1 = (X_{tr}^p, Y_{tr}^p)$

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

Function $\textsc{BackdoorTest}(f, X, Y, Pert, y_p, \ell, Z)$**:**

$X_p = Pert(X)$

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

Return Not Backdoored

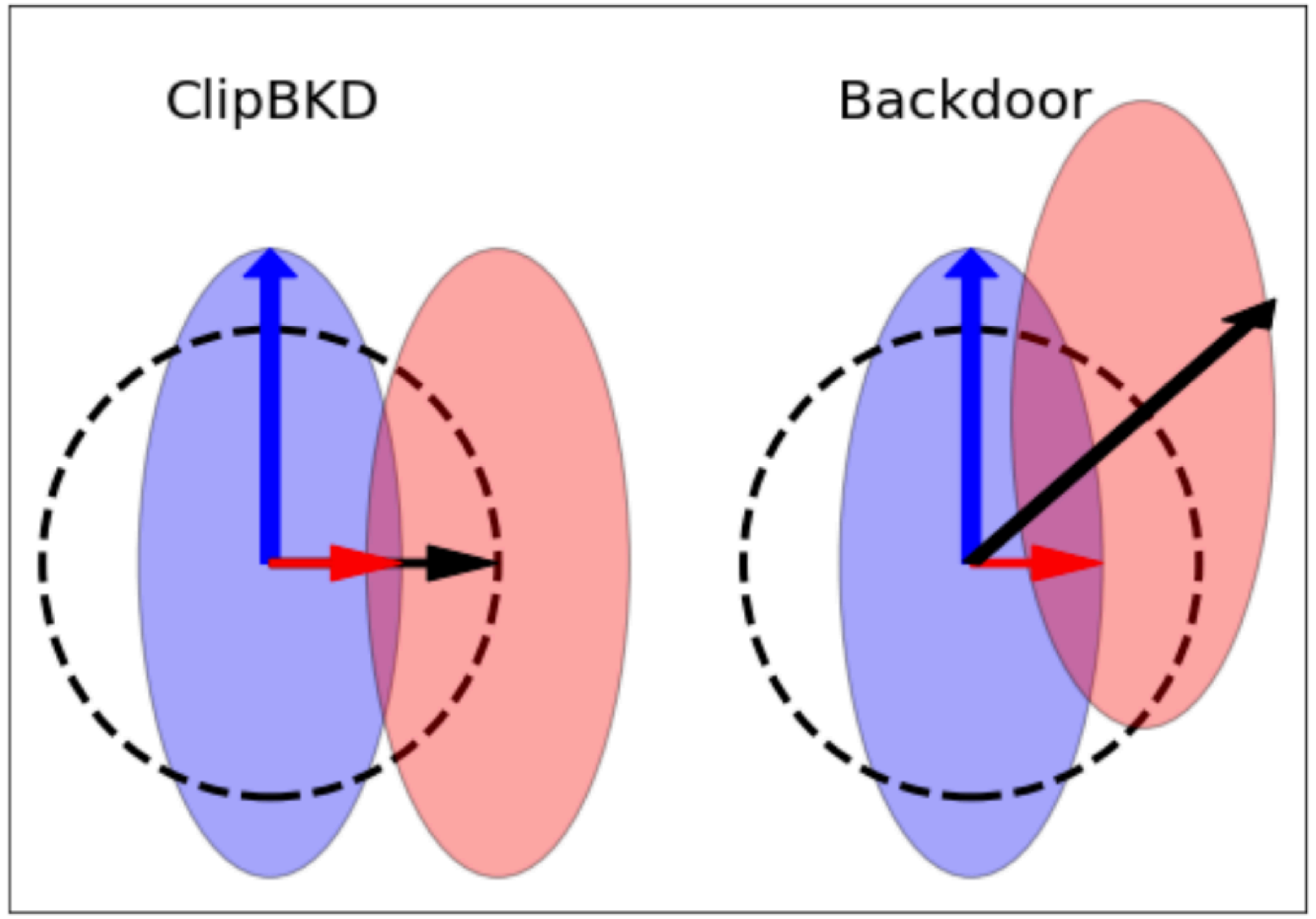
Clipping-Aware Backdoor

- Gradient of model parameters with respect to a poisoning point is

$$\nabla_w \ell(w, b; x_p, y_p) = \ell'(w \cdot x_p + b, y_p) x_p.$$

- For Backdoor attacks:
    - Large $|\ell'(w \cdot x_p + b, y_p)|$
    - Relationship gets broken after clipping
- Clipping-Aware Backdoor:
    - Choose x_p such that it minimizes

$$Var_{(x,y)\in D}\left[\ell'(w \cdot x_p + b, y_p)x_p \cdot \ell'(w \cdot x + b, y)x\right] \leq Var_{(x,y)\in D}[x_p \cdot x].$$



Clipping-Aware Backdoor

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 $\textsc{ClipBkd}(X, Y, k, f, m)$:

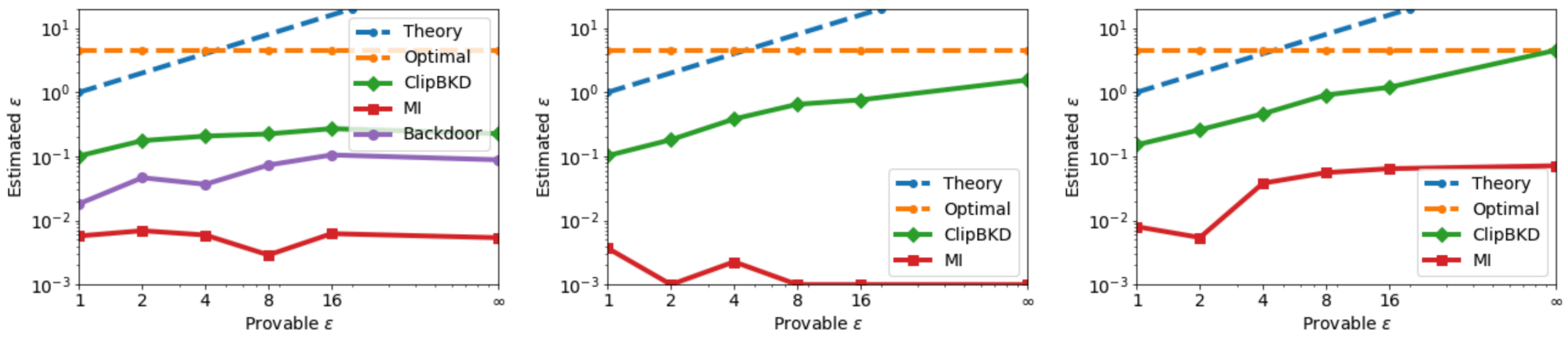
- $U, D, V = SVD(X)$ ▷ Singular value decomposition
- $x_p = mV_d$ ▷ V_d is the singular vector for smallest singular value
- $y_p = \arg\min_i f(x_p)$ ▷ Pick class maximizing gradient norm
- $X_{tr}^p = \textsc{ReplaceRandomRows}(X, [x_p] * k)$ ▷ Add poisoning point k times
- $Y_{tr}^p = \textsc{ReplaceRandomRows}(Y, [y_p] * k)$ ▷ 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

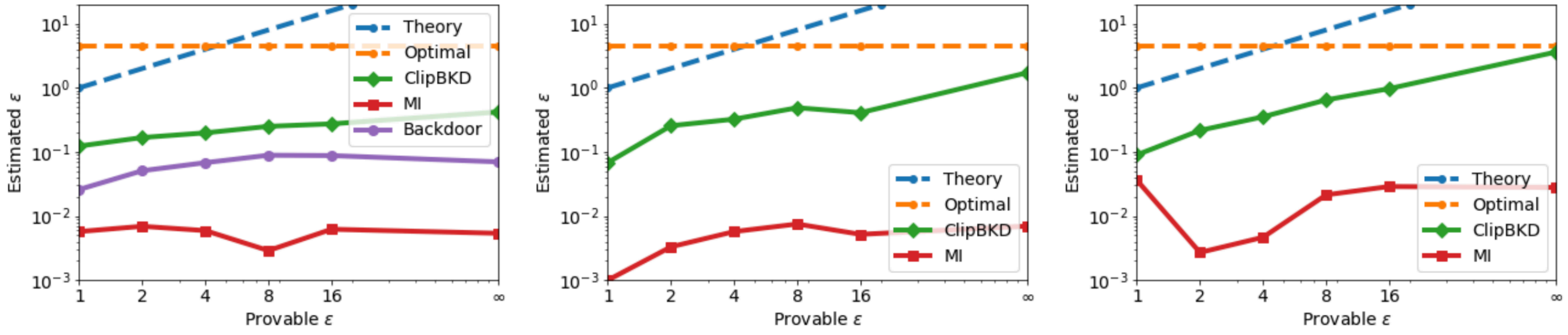
Function $\textsc{ClipBkdTest}(f, x_p, y_p, Z)$:

- **If** $(f(x_p) - f(0^d)) \cdot y_p > Z$ **Return** Backdoored
- **Return** Not Backdoored

Experiments



(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.

Experiments

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.

Thank You