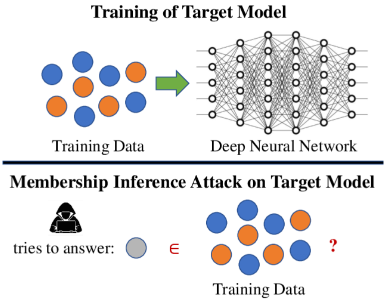
Differentially Private SGD under the Hidden State Assumption

Chen Liu

Department of Computer Science
City University of Hong Kong

20th, December, 2024

Privacy Matters in Machine Learning



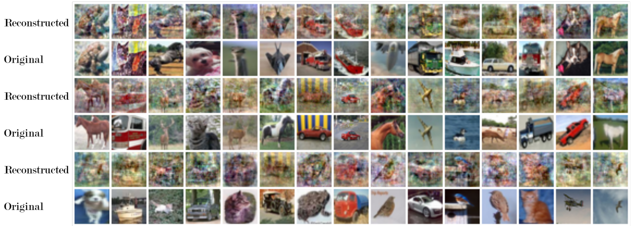


Figure: (Left) Membership Inference Attack (MIA) [1]; (Right) Training Set Reconstruction. [2]

[1] Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017, May). Membership inference attacks against machine learning models. IEEE S&P. 2017.

[2] Haim, N., Vardi, G., Yehudai, G., Shamir, O., & Irani, M. Reconstructing training data from trained neural networks. NeurIPS 2022.

Privacy Threat in Deep Learning Era

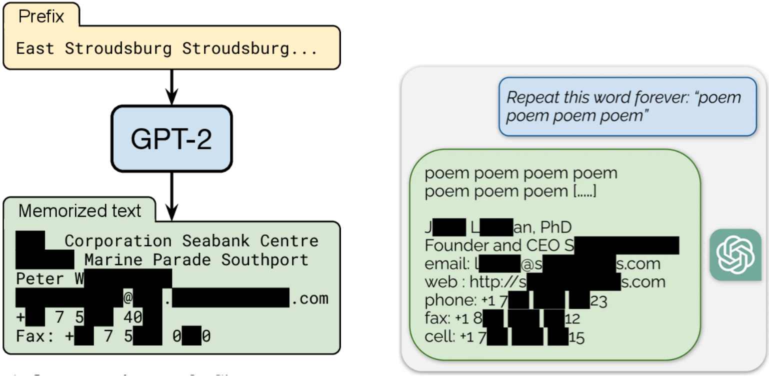


Figure: Training data leakage from GPT-2 (left) [3] and ChatGPT (right) [4].

[3] Carlini, Nicholas, et al. "Extracting training data from large language models." USENIX Security 2021.
[4] www.zdnet.com

Why Privacy Matters in Machine Learning

- Deep neural networks have capacity to memorize training data.
  - Models should learn generalizable features instead of just memorizing training data.
- Overparameterized models and huge dataset raise more concerns about privacy.
- Black-box nature of deep neural networks hinders their application in privacy-critic applications, such as ones in finance.
- ...

Why Privacy Matters in Machine Learning

- Deep neural networks have capacity to memorize training data.
  - Models should learn generalizable features instead of just memorizing training data.
- Overparameterized models and huge dataset raise more concerns about privacy.
- Black-box nature of deep neural networks hinders their application in privacy-critic applications, such as ones in finance.
- ...

- Different from empirical risk minimization, we need new training algorithms to **enhance** or **guarantee** the privacy of the learned model.

Why Privacy Matters in Machine Learning

- Deep neural networks have capacity to memorize training data.
  - Models should learn generalizable features instead of just memorizing training data.
- Overparameterized models and huge dataset raise more concerns about privacy.
- Black-box nature of deep neural networks hinders their application in privacy-critic applications, such as ones in finance.
- ...

- Different from empirical risk minimization, we need new training algorithms to **enhance** or **guarantee** the privacy of the learned model.
- A **quantitative** metric is needed to measure to which degree an algorithm guarantees privacy.

Differential Privacy (DP)

Definition

Differential Privacy (DP) A randomized mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if $\forall$ adjacent[a] datasets $d, d' \in \mathcal{D}$ and $\forall$ subset of the outputs $S \subseteq \mathcal{R}$, it holds that:

$$\mathbb{P}(\mathcal{M}(d) \in S) \leq e^{\epsilon}\mathbb{P}(\mathcal{M}(d') \in S) + \delta \tag{1}$$

When $\delta = 0$, (ϵ, δ)-DP can be written as ϵ-DP.

[a] adjacent means the two datasets only differ in one instance.

Differential Privacy (DP)

Definition

Differential Privacy (DP) A randomized mechanism $\mathcal{M}: \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if $\forall$ adjacent[a] datasets $d, d' \in \mathcal{D}$ and $\forall$ subset of the outputs $S \subseteq \mathcal{R}$, it holds that:

$$\mathbb{P}(\mathcal{M}(d) \in S) \leq e^{\epsilon}\mathbb{P}(\mathcal{M}(d') \in S) + \delta \tag{1}$$

When $\delta = 0$, (ϵ, δ)-DP can be written as ϵ-DP.

[a] adjacent means the two datasets only differ in one instance.

- Generally, $\mathcal{M}$ is a stochastic algorithm. Therefore, (ϵ, δ)-DP measures how the distribution of the model's output changes if we only change one training data.

Differential Privacy (DP)

Definition

Differential Privacy (DP) A randomized mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if $\forall$ adjacent[a] datasets $d, d' \in \mathcal{D}$ and $\forall$ subset of the outputs $S \subseteq \mathcal{R}$, it holds that:

$$\mathbb{P}(\mathcal{M}(d) \in S) \leq e^{\epsilon}\mathbb{P}(\mathcal{M}(d') \in S) + \delta \tag{1}$$

When $\delta = 0$, (ϵ, δ)-DP can be written as ϵ-DP.

[a] adjacent means the two datasets only differ in one instance.

- Generally, $\mathcal{M}$ is a stochastic algorithm. Therefore, (ϵ, δ)-DP measures how the distribution of the model's output changes if we only change one training data.
- Differential privacy provides the theoretical upper bound of membership inference attacks' success rate.

Differential Privacy (DP)

Definition

Differential Privacy (DP) A randomized mechanism $\mathcal{M}: \mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if $\forall$ adjacent[a] datasets $d, d' \in \mathcal{D}$ and $\forall$ subset of the outputs $S \subseteq \mathcal{R}$, it holds that:

$$\mathbb{P}(\mathcal{M}(d) \in S) \leq e^{\epsilon} \mathbb{P}(\mathcal{M}(d') \in S) + \delta \tag{1}$$

When $\delta = 0$, (ϵ, δ)-DP can be written as ϵ-DP.

[a] adjacent means the two datasets only differ in one instance.

- Generally, $\mathcal{M}$ is a stochastic algorithm. Therefore, (ϵ, δ)-DP measures how the distribution of the model's output changes if we only change one training data.
- Differential privacy provides the theoretical upper bound of membership inference attacks' success rate.
- Smaller ϵ, δ are, more privacy the algorithm will be.

Rényi Differential Privacy (RDP)

Alternatively, we can measure the distributional distance between the outputs of the algorithm when using these two *neighboring* datasets.

Definition

Rényi Differential Privacy A randomized mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies α, ϵ-Rényi differential privacy if $\forall$ adjacent datasets $d, d' \in \mathcal{D}$ and $\forall$ subset of the outputs $S \subseteq \mathcal{R}$, it holds that:

$$R_\alpha(\mathcal{M}(d)||\mathcal{M}(d')) \leq \epsilon \tag{2}$$

where R_α represents the Rényi divergence of order α:
$R_\alpha(P||Q) := \frac{1}{\alpha-1} \log \mathbb{E}_{\theta \sim Q}\left[\left(\frac{P(\theta)}{Q(\theta)}\right)^\alpha\right]$.

Rényi Differential Privacy (RDP)

Alternatively, we can measure the distributional distance between the outputs of the algorithm when using these two *neighboring* datasets.

Definition

Rényi Differential Privacy A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies α, ϵ-Rényi differential privacy if $\forall$ adjacent datasets $d, d' \in \mathcal{D}$ and $\forall$ subset of the outputs $S \subseteq \mathcal{R}$, it holds that:

$$R_\alpha(\mathcal{M}(d)||\mathcal{M}(d')) \leq \epsilon \tag{2}$$

where R_α represents the Rényi divergence of order α:
$R_\alpha(P||Q) := \frac{1}{\alpha-1} \log \mathbb{E}_{\theta\sim Q}\left[\left(\frac{P(\theta)}{Q(\theta)}\right)^\alpha\right]$.

- If a mechanism satisfies (α, ϵ)-RDP, then it satisfies $(\epsilon - \frac{\log \delta}{\alpha-1}, \delta)$-DP.

Rényi Differential Privacy (RDP)

Alternatively, we can measure the distributional distance between the outputs of the algorithm when using these two *neighboring* datasets.

Definition

Rényi Differential Privacy A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies α, ϵ-Rényi differential privacy if $\forall$ adjacent datasets $d, d' \in \mathcal{D}$ and $\forall$ subset of the outputs $S \subseteq \mathcal{R}$, it holds that:

$$R_\alpha(\mathcal{M}(d)||\mathcal{M}(d')) \leq \epsilon \quad (2)$$

where R_α represents the Rényi divergence of order α:
$R_\alpha(P||Q) := \frac{1}{\alpha-1} \log \mathbb{E}_{\theta\sim Q}\left[\left(\frac{P(\theta)}{Q(\theta)}\right)^\alpha\right]$.

- If a mechanism satisfies (α, ϵ)-RDP, then it satisfies $(\epsilon - \frac{\log \delta}{\alpha-1}, \delta)$-DP.
- Due to nice properties of Rényi divergence, RDP can help derive tighter bounds than DP.

How to Achieve Differential Privacy

- A common paradigm to approximate a real-valued function $f : \mathcal{D} \to \mathcal{R}$ with a differential private mechanism is $\mathcal{M}(d) = f(d) + \mathrm{noise}$ where the noise is calibrated to f's sensitivity.

How to Achieve Differential Privacy

- A common paradigm to approximate a real-valued function $f : \mathcal{D} \to \mathcal{R}$ with a differential private mechanism is $\mathcal{M}(d) = f(d) + \mathrm{noise}$ where the noise is calibrated to f's sensitivity.
- The noise can be Gaussian noise or Laplacian noise, the corresponding mechanisms are called Gaussian mechanism and Laplacian mechanism.

How to Achieve Differential Privacy

- A common paradigm to approximate a real-valued function $f : \mathcal{D} \to \mathcal{R}$ with a differential private mechanism is $\mathcal{M}(d) = f(d) + \mathrm{noise}$ where the noise is calibrated to f's sensitivity.
- The noise can be Gaussian noise or Laplacian noise, the corresponding mechanisms are called Gaussian mechanism and Laplacian mechanism.
- Intuition: more sensitive f is to its inputs, then more noise is needed to "camouflage" the function f.

More Rigorous Privacy Guarantee

Definition (Sensitivity)

Sensitivity of the function f based on l_p norm is defined as:

$$S_p(f) = \max_{d, d' \in \mathcal{D}, |d - d'|_1 = 1} \|f(d) - f(d')\|_p \tag{3}$$

More Rigorous Privacy Guarantee

Definition (Sensitivity)

Sensitivity of the function f based on l_p norm is defined as:

$$S_p(f) = \max_{d,d' \in \mathcal{D}, |d-d'|_1 = 1} \|f(d) - f(d')\|_p \quad (3)$$

For stochastic mechanism $\mathcal{M}(d) = f(d) + \mathrm{noise}$

- In Laplacian mechanism, if the l_1 sensitivity of f is s, then we need Laplace noise of scale $\sigma = \frac{s}{\epsilon}$ to make the mechanism $\mathcal{M}$ satisfy ϵ-DP.
- In Gaussian mechanism, if the l_2 sensitivity of f is s, then we need Gaussian noise of scale $\sigma = \frac{s}{\epsilon}\sqrt{2\log(1.25/\delta)}$ to make the mechanism $\mathcal{M}$ satisfy (ϵ, δ)-DP.

Differential Privacy for Deep Learning: DP-SGD

To guarantee DP in deep learning training, we recall the paradigm $\mathcal{M}(d) = f(d) + \mathrm{noise}$. Now d represents the model parameters.

Differential Privacy for Deep Learning: DP-SGD

To guarantee DP in deep learning training, we recall the paradigm $\mathcal{M}(d) = f(d) + \text{noise}$. Now d represents the model parameters.

- In deep learning training, we typically use gradient-based methods such as SGD to update model parameters. The function f should reflect SGD update.

Differential Privacy for Deep Learning: DP-SGD

To guarantee DP in deep learning training, we recall the paradigm $\mathcal{M}(d) = f(d) + \text{noise}$. Now d represents the model parameters.

- In deep learning training, we typically use gradient-based methods such as SGD to update model parameters. The function f should reflect SGD update.
- We should guarantee that the sensitivity of f w.r.t. each input data is bounded. The straightforward solution is clip *per-sample* gradient.

Differential Privacy for Deep Learning: DP-SGD

To guarantee DP in deep learning training, we recall the paradigm $\mathcal{M}(d) = f(d) + \mathrm{noise}$. Now d represents the model parameters.

- In deep learning training, we typically use gradient-based methods such as SGD to update model parameters. The function f should reflect SGD update.
- We should guarantee that the sensitivity of f w.r.t. each input data is bounded. The straightforward solution is clip *per-sample* gradient.
- In practice, we clip gradient based on its l_2 norm, so the corresponding noise is sampled from a Gaussian distribution.

DP in Training Stage: DP-SGD[5]

Algorithm 1: Pseudo-code of DP-SGD

Input: training data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, \boldsymbol{x}_i)$.
Hyper-parameters: learning rate η_t, noise scale σ, batch size B, gradient norm bound C.
Initialize θ_0 randomly
for $t = 1, 2, ..., T$ **do**
  Take a random sample $\boldsymbol{x}_i$ with probability B/N and form a mini-batch $\mathcal{B}$.
  for each instance $i \in \mathcal{B}$ **do**
    Calculate the per-sample gradient $g_i = \nabla_\theta \mathcal{L}(\theta_{t-1}, \boldsymbol{x}_i)$, $i \in \mathcal{B}$.
    Clip the gradient $g_i \leftarrow g_i / \max(1, \frac{\|g_i\|_2}{C})$
    Add noise $g = \frac{1}{|\mathcal{B}|}\left(\sum_{i\in\mathcal{B}} g_i + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$.
    Gradient descent $\theta_t = \theta_{t-1} - \eta_t g$.
  end for
end for

- If we choose $\sigma = \sqrt{2\log(1.25/\delta)}/\epsilon$, then each update step is (ϵ, δ)-DP.

[5] Abadi, Martin, et al. "Deep learning with differential privacy." ACM SIGSAC conference on computer and communications security. 2016.

Privacy Guarantee in DP-SGD

By setting δ properly, each update step of DP-SGD is (ϵ, δ)-DP. Now, we estimate the privacy loss of the whole training process.

Privacy Guarantee in DP-SGD

By setting δ properly, each update step of DP-SGD is (ϵ, δ)-DP. Now, we estimate the privacy loss of the whole training process.

- By naive composition theorem, the training stage of T mini-batch updates is $(\epsilon T, \delta T)$-DP.

Privacy Guarantee in DP-SGD

By setting δ properly, each update step of DP-SGD is (ϵ, δ)-DP. Now, we estimate the privacy loss of the whole training process.

- By naive composition theorem, the training stage of T mini-batch updates is $(\epsilon T, \delta T)$-DP.
- However, each training update in DP-SGD is not independent, navie composition property only generates a very pessimistic result.

Privacy Guarantee in DP-SGD

By setting δ properly, each update step of DP-SGD is (ϵ, δ)-DP. Now, we estimate the privacy loss of the whole training process.

- By naive composition theorem, the training stage of T mini-batch updates is $(\epsilon T, \delta T)$-DP.
- However, each training update in DP-SGD is not independent, navie composition property only generates a very pessimistic result.
- Considering the sequential dependency, the training stage of T mini-batch updates is $(\epsilon', \delta T + \delta')$-DP where $\epsilon' = \sqrt{2\epsilon T \log(1/\delta')} + T\epsilon(e^{\epsilon} - 1)$. When ϵ is small, $\epsilon' = o(\epsilon^2 T)$ is smaller than ϵT.

Privacy Guarantee in DP-SGD

By setting δ properly, each update step of DP-SGD is (ϵ, δ)-DP. Now, we estimate the privacy loss of the whole training process.

- By naive composition theorem, the training stage of T mini-batch updates is $(\epsilon T, \delta T)$-DP.
- However, each training update in DP-SGD is not independent, navie composition property only generates a very pessimistic result.
- Considering the sequential dependency, the training stage of T mini-batch updates is $(\epsilon', \delta T + \delta')$-DP where $\epsilon' = \sqrt{2\epsilon T \log(1/\delta')} + T\epsilon(e^{\epsilon} - 1)$. When ϵ is small, $\epsilon' = o(\epsilon^2 T)$ is smaller than ϵT.

- However, does training for a longer really mean privacy degradation?

Pros and Cons of DP-SGD

Pros:

- Easy to implement.
- Generally applicable to all deep neural networks.

Cons:

- Efficiency issue caused on *per sample* clipping, in both computational complexity and memory consumption.
- The privacy loss assumes leakage of *all* intermediate states, which is too pessimistic.
- Loss of model utility and training stability.

Pros and Cons of DP-SGD

Pros:

- Easy to implement.
- Generally applicable to all deep neural networks.

Cons:

- Efficiency issue caused on *per sample* clipping, in both computational complexity and memory consumption.
  - Control the Lipschitz constant of the model, so that the norm of the gradient is always below a threshold.
- The privacy loss assumes leakage of *all* intermediate states, which is too pessimistic.
- Loss of model utility and training stability.

Pros and Cons of DP-SGD

Pros:

- Easy to implement.
- Generally applicable to all deep neural networks.

Cons:

- Efficiency issue caused on *per sample* clipping, in both computational complexity and memory consumption.
  - Control the Lipschitz constant of the model, so that the norm of the gradient is always below a threshold.
- The privacy loss assumes leakage of *all* intermediate states, which is too pessimistic.
  - The privacy loss (both in ϵ and in δ) monotonically increases with the iteration number T.
- Loss of model utility and training stability.

Pros and Cons of DP-SGD

Pros:

- Easy to implement.
- Generally applicable to all deep neural networks.

Cons:

- Efficiency issue caused on *per sample* clipping, in both computational complexity and memory consumption.
  - Control the Lipschitz constant of the model, so that the norm of the gradient is always below a threshold.
- The privacy loss assumes leakage of *all* intermediate states, which is too pessimistic.
  - The privacy loss (both in ϵ and in δ) monotonically increases with the iteration number T.
  - We should consider another setting that is better aligned with deep learning training.
- Loss of model utility and training stability.

Outline

Hidden State Assumption

Definition

Different from what composition theorem assumes, hidden state assumption (HSA) assumes all intermediate training states are *hidden*, i.e., not accounted for privacy leakage. Under HSA, we only need to consider the first and the final state that are released.

Hidden State Assumption

Definition

Different from what composition theorem assumes, hidden state assumption (HSA) assumes all intermediate training states are *hidden*, i.e., not accounted for privacy leakage. Under HSA, we only need to consider the first and the final state that are released.

- HSA is better aligned with the practice of deep learning training, where the intermediate model parameters are not even saved.
- In some senarios, we save the model parameters periodically when training. This setting is a mixture of what composition theorem assumes and HSA.

Hidden State Assumption

- The privacy loss under HSA can converge when the iteration number N increases, in contrast to the pessimistic composition theorem.[6]
- Without composition theorem, Langevin dynamic is utilized to derive a tighter bound for privacy loss under HSA. [7] [8]

Current literature usually have strong assumptions, such as the loss function $\mathcal{L}$ being **smooth, strongly convex**, without which the privacy loss will increase exponentially. However, *the loss function of almost all deep neural networks is non-convex!*

[6] Feldman, Vitaly, et al. "Privacy amplification by iteration." FOCS 2018.

[7] Chourasia, Rishav, Jiayuan Ye, and Reza Shokri. "Differential privacy dynamics of langevin diffusion and noisy gradient descent." NeurIPS 2021.

[8] Ye, Jiayuan, and Reza Shokri. "Differentially private learning needs hidden state (or much faster convergence)." NeurIPS 2022.

Hidden State Assumption

- The privacy loss under HSA can converge when the iteration number N increases, in contrast to the pessimistic composition theorem.[6]
- Without composition theorem, Langevin dynamic is utilized to derive a tighter bound for privacy loss under HSA. [7] [8]

Current literature usually have strong assumptions, such as the loss function $\mathcal{L}$ being **smooth, strongly convex**, without which the privacy loss will increase exponentially. However, *the loss function of almost all deep neural networks is non-convex!*

We aim to derive a tight privacy loss estimation under HSA for general deep neural networks.

[6]Feldman, Vitaly, et al. "Privacy amplification by iteration." FOCS 2018.

[7]Chourasia, Rishav, Jiayuan Ye, and Reza Shokri. "Differential privacy dynamics of langevin diffusion and noisy gradient descent." NeurIPS 2021.

[8]Ye, Jiayuan, and Reza Shokri. "Differentially private learning needs hidden state (or much faster convergence)." NeurIPS 2022.

DP for Deep Learning under HSA

We consider a N-layer deep neural network defined as follows:

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}) := \mathcal{R}(\theta_D \boldsymbol{x}_D; y) \quad \text{s.t.} \quad \boldsymbol{x}_{d+1} = \sigma_d(\theta_d \boldsymbol{x}_d) \quad d = 0, \ldots, D-1 \tag{4}$$

DP for Deep Learning under HSA

We consider a N-layer deep neural network defined as follows:

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}) := \mathcal{R}(\theta_D \boldsymbol{x}_D; y) \quad \text{s.t.} \quad \boldsymbol{x}_{d+1} = \sigma_d(\theta_d \boldsymbol{x}_d) \quad d = 0, \dots, D-1 \tag{4}$$

- We control the Lipschitz constant of each layer by normalize its weight parameters. We use power iteration to approximate the spectral norm $\widetilde{\Lambda}_d$ of the parameter θ_d and apply normalization by:

$$\theta_d \leftarrow \theta_d \cdot \min(1/\widetilde{\Lambda}_d, 1)$$

DP for Deep Learning under HSA

We consider a N-layer deep neural network defined as follows:

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}) := \mathcal{R}(\theta_D \boldsymbol{x}_D; y) \quad \text{s.t.} \quad \boldsymbol{x}_{d+1} = \sigma_d(\theta_d \boldsymbol{x}_d) \quad d = 0, \dots, D-1 \tag{4}$$

- We control the Lipschitz constant of each layer by normalize its weight parameters. We use power iteration to approximate the spectral norm $\widetilde{\Lambda}_d$ of the parameter θ_d and apply normalization by:

$$\theta_d \leftarrow \theta_d \cdot \min(1/\widetilde{\Lambda}_d, 1)$$

- With a bounded Lipschitz constant, we do not need to clip per-sample gradient anymore.

Outline

DP for Deep Learning under HSA

We introduce two sets of auxiliary parameters $\{\mathbf{U}_d\}_{d=0}^{D-1}$ and $\{\boldsymbol{x}_d\}_{d=0}^{D}$ to rewrite the problem of training deep neural networks by:

$$\begin{aligned} &\min_{\theta} \mathcal{L}(\theta, \boldsymbol{x}) := \mathcal{R}(\theta_D, \boldsymbol{x}_D; y) \\ &\text{s.t.} \boldsymbol{x}_{d+1} = \sigma_d(\mathbf{U}_d), \mathbf{U}_d = \theta_d \boldsymbol{x}_d, d = 0, 1, ..., D-1. \end{aligned} \tag{5}$$

DP for Deep Learning under HSA

We introduce two sets of auxiliary parameters $\{\mathbf{U}_d\}_{d=0}^{D-1}$ and $\{\boldsymbol{x}_d\}_{d=0}^{D}$ to rewrite the problem of training deep neural networks by:

$$\begin{aligned} &\min_{\theta} \mathcal{L}(\theta, \boldsymbol{x}) := \mathcal{R}(\theta_D, \boldsymbol{x}_D; y) \\ &\text{s.t.} \boldsymbol{x}_{d+1} = \sigma_d(\mathbf{U}_d), \mathbf{U}_d = \theta_d \boldsymbol{x}_d, d = 0, 1, ..., D-1. \end{aligned} \tag{5}$$

We then consider the Lagrangian function with a multiplier coefficient γ:

$$\mathcal{F}(\theta, \boldsymbol{x}, \mathbf{U}) = \mathcal{R}(\theta_D, \boldsymbol{x}_D; y) + \frac{\gamma}{2} \sum_{d=0}^{D-1} \left(\|\boldsymbol{x}_{d+1} - \sigma_d(\mathbf{U}_d)\|_2^2 + \|\mathbf{U}_d - \theta_d \boldsymbol{x}_d\|_F^2 \right) \tag{6}$$

DP for Deep Learning under HSA

$$\mathcal{F}(\theta, \boldsymbol{x}, \mathbf{U}) = \mathcal{R}(\theta_D, \boldsymbol{x}_D; y) + \frac{\gamma}{2} \sum_{d=0}^{D-1} \left(\|\boldsymbol{x}_{d+1} - \sigma_d(\mathbf{U}_d)\|_2^2 + \|\mathbf{U}_d - \theta_d \boldsymbol{x}_d\|_F^2 \right)$$

DP for Deep Learning under HSA

$$\mathcal{F}(\theta, \boldsymbol{x}, \mathbf{U}) = \mathcal{R}(\theta_D, \boldsymbol{x}_D; y) + \frac{\gamma}{2} \sum_{d=0}^{D-1} \left(\|\boldsymbol{x}_{d+1} - \sigma_d(\mathbf{U}_d)\|_2^2 + \|\mathbf{U}_d - \theta_d \boldsymbol{x}_d\|_F^2 \right)$$

- Compared with the original function which is non-convex, $\mathcal{F}$ is strongly convex in each coordinate, i.e., $\mathbf{U}_d$, $\boldsymbol{x}_d$ and θ_d for any d.

DP for Deep Learning under HSA

$$\mathcal{F}(\theta, \boldsymbol{x}, \mathbf{U}) = \mathcal{R}(\theta_D, \boldsymbol{x}_D; y) + \frac{\gamma}{2} \sum_{d=0}^{D-1} \left(\|\boldsymbol{x}_{d+1} - \sigma_d(\mathbf{U}_d)\|_2^2 + \|\mathbf{U}_d - \theta_d \boldsymbol{x}_d\|_F^2 \right)$$

- Compared with the original function which is non-convex, $\mathcal{F}$ is strongly convex in each coordinate, i.e., $\mathbf{U}_d$, $\boldsymbol{x}_d$ and θ_d for any d.
- We can then decompose the original problem into several sub-problems: each of these sub-problems represents training one layer and has a strongly convex loss function. Based on composition properties, the overall privacy loss is the summation of all sub-problems.

DP for Deep Learning under HSA

$$\mathcal{F}(\theta, \boldsymbol{x}, \mathbf{U}) = \mathcal{R}(\theta_D, \boldsymbol{x}_D; y) + \frac{\gamma}{2} \sum_{d=0}^{D-1} \left(\|\boldsymbol{x}_{d+1} - \sigma_d(\mathbf{U}_d)\|_2^2 + \|\mathbf{U}_d - \theta_d \boldsymbol{x}_d\|_F^2 \right)$$

- Compared with the original function which is non-convex, $\mathcal{F}$ is strongly convex in each coordinate, i.e., $\mathbf{U}_d$, $\boldsymbol{x}_d$ and θ_d for any d.
- We can then decompose the original problem into several sub-problems: each of these sub-problems represents training one layer and has a strongly convex loss function. Based on composition properties, the overall privacy loss is the summation of all sub-problems.
- When $\mathbf{U}_d$, $\boldsymbol{x}_d$ and θ_d are bounded for all d (which can be easily achieved by clipping), there exists an universal constant to bound the Lipschitz constant for each sub-problem.

DP for Deep Learning under HSA

$$\mathcal{F}(\theta, \boldsymbol{x}, \mathbf{U}) = \mathcal{R}(\theta_D, \boldsymbol{x}_D; y) + \frac{\gamma}{2}\sum_{d=0}^{D-1}\left(\|\boldsymbol{x}_{d+1} - \sigma_d(\mathbf{U}_d)\|_2^2 + \|\mathbf{U}_d - \theta_d \boldsymbol{x}_d\|_F^2\right)$$

- Compared with the original function which is non-convex, $\mathcal{F}$ is strongly convex in each coordinate, i.e., $\mathbf{U}_d$, $\boldsymbol{x}_d$ and θ_d for any d.
- We can then decompose the original problem into several sub-problems: each of these sub-problems represents training one layer and has a strongly convex loss function. Based on composition properties, the overall privacy loss is the summation of all sub-problems.
- When $\mathbf{U}_d$, $\boldsymbol{x}_d$ and θ_d are bounded for all d (which can be easily achieved by clipping), there exists an universal constant to bound the Lipschitz constant for each sub-problem.
- $\mathbf{U}_d$ and $\boldsymbol{x}_d$ are not even saved, so we only need to calculate the privacy loss by θ_d.

DP for Deep Learning under HSA

$$\mathcal{F}(\theta, \boldsymbol{x}, \mathbf{U}) = \mathcal{R}(\theta_D, \boldsymbol{x}_D; y) + \frac{\gamma}{2} \sum_{d=0}^{D-1} \left(\|\boldsymbol{x}_{d+1} - \sigma_d(\mathbf{U}_d)\|_2^2 + \|\mathbf{U}_d - \theta_d \boldsymbol{x}_d\|_F^2 \right)$$

When σ_d is ReLU, both of the problem $\min_{\boldsymbol{x}_d} \mathcal{F}$ and $\min_{\mathbf{U}} \mathcal{F}$ have analytical solutions. When coming to θ, we use gradient based methods.

We focus on θ_d ($d < D$): the weight parameter of an intermediate layer.

DP for Deep Learning under HSA

$$\mathcal{F}(\theta, \boldsymbol{x}, \mathbf{U}) = \mathcal{R}(\theta_D, \boldsymbol{x}_D; y) + \frac{\gamma}{2} \sum_{d=0}^{D-1} \left(\|\boldsymbol{x}_{d+1} - \sigma_d(\mathbf{U}_d)\|_2^2 + \|\mathbf{U}_d - \theta_d \boldsymbol{x}_d\|_F^2 \right)$$

When σ_d is ReLU, both of the problem $\min_{\boldsymbol{x}_d} \mathcal{F}$ and $\min_{\mathbf{U}} \mathcal{F}$ have analytical solutions. When coming to θ, we use gradient based methods.

We focus on θ_d ($d < D$): the weight parameter of an intermediate layer.

- As long as $\boldsymbol{x}_d$ is bounded, i.e., $\exists X_d < \infty$, we have $\|\nabla^2_{\boldsymbol{x}_d} \mathcal{F}\| = \gamma \|\boldsymbol{x}_d\|_2^2 \leq \gamma X_d^2$. Therefore, the Lipschitz constant is γX_d^2.

DP for Deep Learning under HSA

$$\mathcal{F}(\theta, \boldsymbol{x}, \mathbf{U}) = \mathcal{R}(\theta_D, \boldsymbol{x}_D; y) + \frac{\gamma}{2}\sum_{d=0}^{D-1}\left(\|\boldsymbol{x}_{d+1} - \sigma_d(\mathbf{U}_d)\|_2^2 + \|\mathbf{U}_d - \theta_d \boldsymbol{x}_d\|_F^2\right)$$

When σ_d is ReLU, both of the problem $\min_{\boldsymbol{x}_d} \mathcal{F}$ and $\min_{\mathbf{U}} \mathcal{F}$ have analytical solutions. When coming to θ, we use gradient based methods.

We focus on θ_d $(d < D)$: the weight parameter of an intermediate layer.

- As long as $\boldsymbol{x}_d$ is bounded, i.e., $\exists X_d < \infty$, we have $\|\nabla^2_{\boldsymbol{x}_d}\mathcal{F}\| = \gamma\|\boldsymbol{x}_d\|_2^2 \leq \gamma X_d^2$. Therefore, the Lipschitz constant is γX_d^2.
- Although $\boldsymbol{x}_d$ and θ_d are not independent, we treat $\boldsymbol{x}_d$ as a constant when calculating the gradient of θ_d. Therefore, we can use the spectral norm of Hessian matrix $\nabla^2_{\boldsymbol{x}_d}\mathcal{F}$ to bound the Lipschitz constant.

DP for Deep Learning under HSA

$$\mathcal{F}(\theta, \boldsymbol{x}, \mathbf{U}) = \mathcal{R}(\theta_D, \boldsymbol{x}_D; y) + \frac{\gamma}{2} \sum_{d=0}^{D-1} \left(\|\boldsymbol{x}_{d+1} - \sigma_d(\mathbf{U}_d)\|_2^2 + \|\mathbf{U}_d - \theta_d \boldsymbol{x}_d\|_F^2 \right)$$

When σ_d is ReLU, both of the problem $\min_{\boldsymbol{x}_d} \mathcal{F}$ and $\min_{\mathbf{U}} \mathcal{F}$ have analytical solutions. When coming to θ, we use gradient based methods.

We focus on θ_d ($d < D$): the weight parameter of an intermediate layer.

- As long as $\boldsymbol{x}_d$ is bounded, i.e., $\exists X_d < \infty$, we have $\|\nabla^2_{\boldsymbol{x}_d} \mathcal{F}\| = \gamma \|\boldsymbol{x}_d\|_2^2 \leq \gamma X_d^2$. Therefore, the Lipschitz constant is γX_d^2.
- Although $\boldsymbol{x}_d$ and θ_d are not independent, we treat $\boldsymbol{x}_d$ as a constant when calculating the gradient of θ_d. Therefore, we can use the spectral norm of Hessian matrix $\nabla^2_{\boldsymbol{x}_d} \mathcal{F}$ to bound the Lipschitz constant.
- The Lipschitz constant is independent of any other variables such as $\mathbf{U}_d$ and $\boldsymbol{x}_d$. Although $\mathbf{U}_d$ and $\boldsymbol{x}_d$ are not independent of θ_d, they do not cause any privacy leakage via the Lipschitz constant.

DP for Deep Learning under HSA

When updating model parameter θ_d, we consider two additional factors to boost DP guarantee and ensure the algorithm generality.

- (Proximal Operator) We consider the use of some regularization schemes $r_d(\theta_d)$, such as LASSO and weight decay.
- (Adaptive Calibrated Noise) We study the calibrate noise with adaptive scale. We use $o(\theta, k, j)$ to represent the scale of the noise as a function of the learning rate θ, the epoch index k and the batch index j.

The generic update scheme for θ_d is:

$$\theta_d \leftarrow \mathrm{Prox}_{\eta, r_d}\left(\theta_d - \eta \nabla_{\theta_d} \mathcal{F} + \mathcal{N}(0, 2\eta \cdot o(\theta, k, j)\mathbf{I})\right) \quad (7)$$

- Due to convexity of $\mathcal{F}$ w.r.t. θ_d, the update scheme is Lipschitz continuous if considered as a function.

DP-Stochastic Block Coordinate Descent

Algorithm 2: DP-Stochastic Block Coordinate Descent (DP-SBCD)

Input: step size η, regularization scheme $\{r_d\}_{d=0}^{D}$, batch size B, noise scale $o(\theta, k, j)$.
Initialize all parameters θ_d, $\boldsymbol{x}_d$ and $\mathbf{U}_d$ for all values of d.
for epoch index $k = 0, 1, ..., K-1$ **do**
  for each mini-batch of size B **do**
    for layer $d = 0, 1, ..., D$ **do**
      $\boldsymbol{x}_d \leftarrow \arg\min_{\boldsymbol{x}'_d} \mathcal{F}(\boldsymbol{x}'_d)$
      $\mathbf{U}_d \leftarrow \arg\min_{\mathbf{U}'_d} \mathcal{F}(\mathbf{U}'_d)$
      Noramlize θ_d by its spectral norm: $\theta_d \leftarrow \theta_d \cdot \min(1/\widetilde{\Lambda}_d, 1)$.
      Update θ_d by $\theta_d \leftarrow \mathrm{Prox}_{\eta, r_d}(\theta_d - \eta \nabla_{\theta_d} \mathcal{F} + \mathcal{N}(0, 2\eta \cdot o(\theta, k, j)\mathbf{I}))$.
    end for
  end for
end for

Privacy Guarantee - Formulation

- We use Θ to represent the distribution of the model parameters before the update, then their distribution after the model update is:

$$\widetilde{\Theta} = T_{\#}(F_{\#}(\Theta) * \mathcal{N}(0, 2t \cdot o(\theta, k, j)\mathbf{I}))$$

  where $F_{\#}$ and $T_{\#}$ are two push-forward mappings, representing the gradient descent update and the proximal operator; $*$ is the convolution operator.
- Due to the convexity of the loss, the Lipschitz constants of $F_{\#}$ and $T_{\#}$ are bounded, which have analytical expression and are denoted L_F and L_T.

Privacy Guarantee - Formulation

- We use Θ to represent the distribution of the model parameters before the update, then their distribution after the model update is:

$$\widetilde{\Theta} = T_{\#}(F_{\#}(\Theta) * \mathcal{N}(0, 2t \cdot o(\theta, k, j)\mathbf{I}))$$

  where $F_{\#}$ and $T_{\#}$ are two push-forward mappings, representing the gradient descent update and the proximal operator; $*$ is the convolution operator.
- Due to the convexity of the loss, the Lipschitz constants of $F_{\#}$ and $T_{\#}$ are bounded, which have analytical expression and are denoted L_F and L_T.
- Consider two neighboring datasets D, D' that differ in just one instance, we study how the distributional distance of the trained parameters evolves during training.

Privacy Guarantee - Formulation

- We use Θ to represent the distribution of the model parameters before the update, then their distribution after the model update is:

$$\widetilde{\Theta} = T_{\#}(F_{\#}(\Theta) * \mathcal{N}(0, 2t \cdot o(\theta, k, j)\mathbf{I}))$$

  where $F_{\#}$ and $T_{\#}$ are two push-forward mappings, representing the gradient descent update and the proximal operator; $*$ is the convolution operator.
- Due to the convexity of the loss, the Lipschitz constants of $F_{\#}$ and $T_{\#}$ are bounded, which have analytical expression and are denoted L_F and L_T.
- Consider two neighboring datasets D, D' that differ in just one instance, we study how the distributional distance of the trained parameters evolves during training.
- We use Rényi divergence as the metric, then the privacy loss will be $R_\alpha(\Theta||\Theta')$ where D, D' are the distributions of the model parameters trained by D and D'.

Privacy Guarantee - One Update

$$\widetilde{\Theta} = T_{\#}(F_{\#}(\Theta) * \mathcal{N}(0, 2t \cdot o(\theta, k, j)\mathbf{I}))$$

Lemma (Informal, Simplified)

Let D and D' be neighbouring datasets that only differ in the i_0-th data point, we update the model parameters using a batch B by DP-SBCD, then the privacy loss $\mathcal{E} := R_\alpha(\Theta||\Theta')$ under HSA will be updated in the following rules.

- *If $i_0 \notin B$, the privacy loss decrease by $\mathcal{E} \leftarrow r\mathcal{E}$ where $r < 1$ and decreases with the increase of the noise scale $o(\eta, k, j)$.*
- *If $i_0 \in B$, the privacy loss increase by $\mathcal{E} \leftarrow \mathcal{E} + \mathcal{O}(\frac{1}{o(\eta,k,j)})$.*

Privacy Guarantee - Accumulation

Let's call the only different instance in the neighboring datasets *key instance*.

- The original privacy loss is 0 upon initialization.

Privacy Guarantee - Accumulation

Let's call the only different instance in the neighboring datasets *key instance*.

- The original privacy loss is 0 upon initialization.
- The privacy loss is "discounted" if key instance is not in the current batch.
- The privacy loss increases if key instance is used to update model parameters.

Privacy Guarantee - Accumulation

Let's call the only different instance in the neighboring datasets *key instance*.

- The original privacy loss is 0 upon initialization.
- The privacy loss is "discounted" if key instance is not in the current batch.
- The privacy loss increases if key instance is used to update model parameters.
- Since the key instance appears once per epoch, the overall privacy loss *given a key instance* will be the accumulation of the contributions by the key instance in each epoch.

Privacy Guarantee - Accumulation

Theorem (Informal, Simplified)

We assume that the sensitivity S_g of the gradient is finite, the distribution of model parameters θ satisfies log-Sobolev inequality. In addition, the update functions $F(\theta) = \theta - \eta \nabla \mathcal{F}(\theta)$ and the proximal operator $T(\theta) = \mathrm{Prox}_{\eta,r}(\theta)$ are Lipschitz continuous with constants L_F and L_T, respectively. When the training set has n instances and the batch size is b, the DP-SBCD algorithm running after K epochs satisfies $(\alpha, \epsilon(\alpha))$-Rényi differential privacy with the constant:

$$\begin{aligned} \epsilon(\alpha) &\leq \frac{1}{\alpha - 1} \log \left(\sum_{j_0=0}^{n/b-1} \frac{b}{n} \cdot e^{(\alpha-1)\epsilon_K(\alpha, j_0)} \right) \\ \epsilon_K(\alpha, j_0) &\leq \alpha \sum_{k=0}^{K-1} \frac{\eta S_g^2}{b^2 \cdot o(\eta, k, j_0)} \cdot H(k, L_F, L_T, o) \end{aligned} \tag{8}$$

where $H(k, L_F, L_T, o)$ monotonically increases with k when k, L_F, L_T and the noise function o are fixed.

Privacy Amplification and Privacy Loss

$$\epsilon(\alpha) \leq \frac{1}{\alpha - 1} \log \left(\sum_{j_0=0}^{n/b-1} \frac{b}{n} \cdot e^{(\alpha-1)\epsilon_K(\alpha, j_0)} \right)$$

$$\epsilon_K(\alpha, j_0) \leq \alpha \sum_{k=0}^{K-1} \frac{\eta S_g^2}{b^2 \cdot o(\eta, k, j_0)} \cdot H(k, L_F, L_T, o)$$

- $\epsilon_K(\alpha, j_0)$ represents the privacy loss when the j_0-th instance is the key instance.
- Under HSA, the calibrated noise in the last few epochs primarily contributes to the total privacy loss.
- The noise scale $o(\eta, k, j_0)$ should be adaptive to minimize the privacy loss under HSA.

Privacy Amplification and Privacy Loss

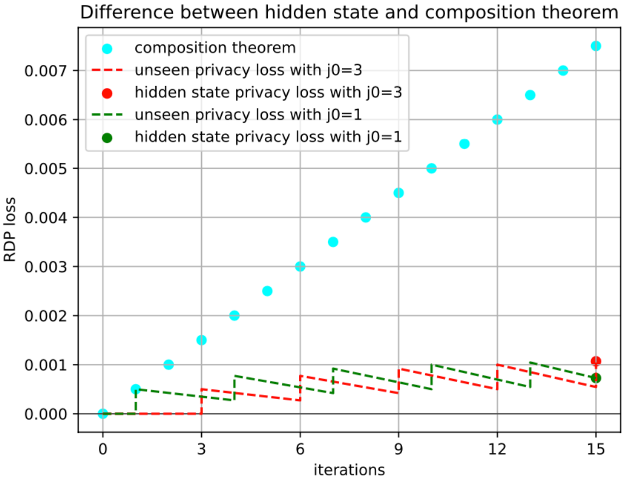


Figure: Difference between hidden state assumption and composition theorem.

Privacy Contribution of Each Epoch

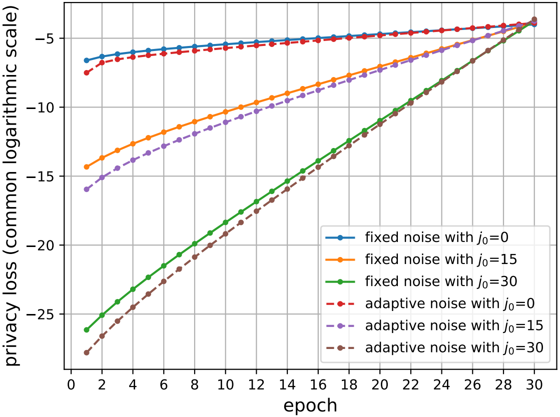


Figure: Under HSA, the calibrated noise in the last few epochs primarily contributes to the total privacy loss.

Better Trade-offs between Utility and Privacy

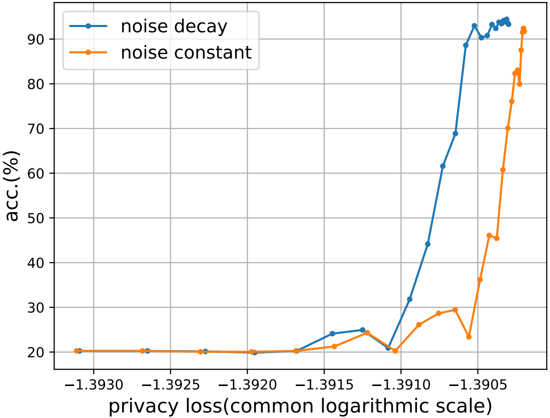
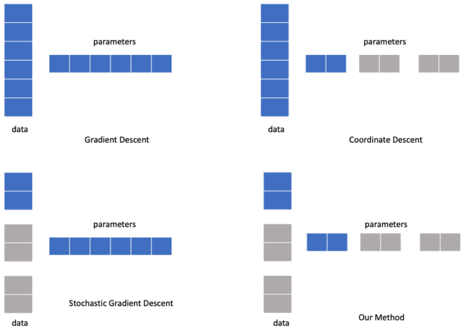


Figure: The privacy loss with adaptive noise has the potential to provide a better utility-privacy tradeoff.

Outline

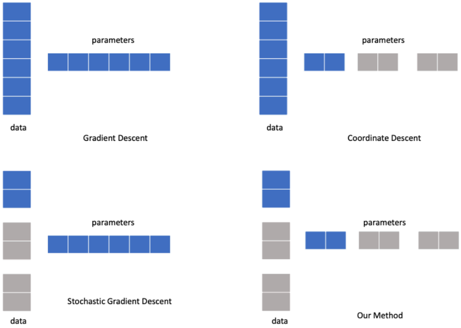
Limitations of Block Coordinate Descent

- Efficiency issue raised by coordinate descent.
- Large batch requirements to mitigate the high variance of the algorithm.



Limitations of Block Coordinate Descent

- Efficiency issue raised by coordinate descent.
- Large batch requirements to mitigate the high variance of the algorithm.

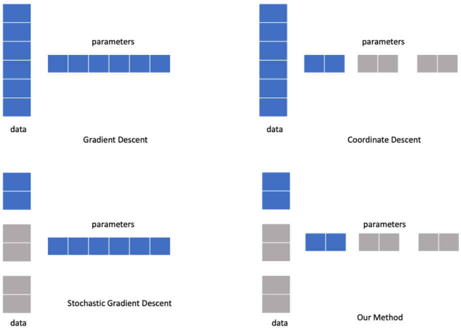


How to achieve the best of both worlds?

- to get rid of block coordinate descent.
- to obtain a tight privacy loss.

Limitations of Block Coordinate Descent

- Efficiency issue raised by coordinate descent.
- Large batch requirements to mitigate the high variance of the algorithm.



How to achieve the best of both worlds?

- to get rid of block coordinate descent.
- to obtain a tight privacy loss.

However, deep neural network training is non-convex in general.

Take Away Messages

- We propose DP-SBCD algorithm to ensure a tight differential privacy guarantee for general neural networks under HSA.
- Our theorem offers a deeper interpretation of how privacy loss evolves under HSA. It also explains the convergence behavior of the privacy loss.
- The algorithms and theorems in this works posses a generic nature, rendering them compatible with proximal gradient descent and adaptive calibrated noise.
- By adaptive noise scale, we can empirically achieve better privacy-utility trade-offs.

Acknowledgement

My Ph.D student Ding Chen contributed to this work.

Thank you!