Convex Approximation of ReLU Networks for Hidden State Differential Privacy

To appear at NeurIPS 2025

Rob Romijnders, University of Amsterdam (r.romijnders@uva.nl)
Antti Koskela, Nokia Bell Labs (antti.h.koskela@nokia-bell-labs.com)

Code and slides: github.com/robromijnders/hiddenstate

Differential Privacy: What is the problem?

Users want to contribute data

Need a guarantee that their details are not leaked

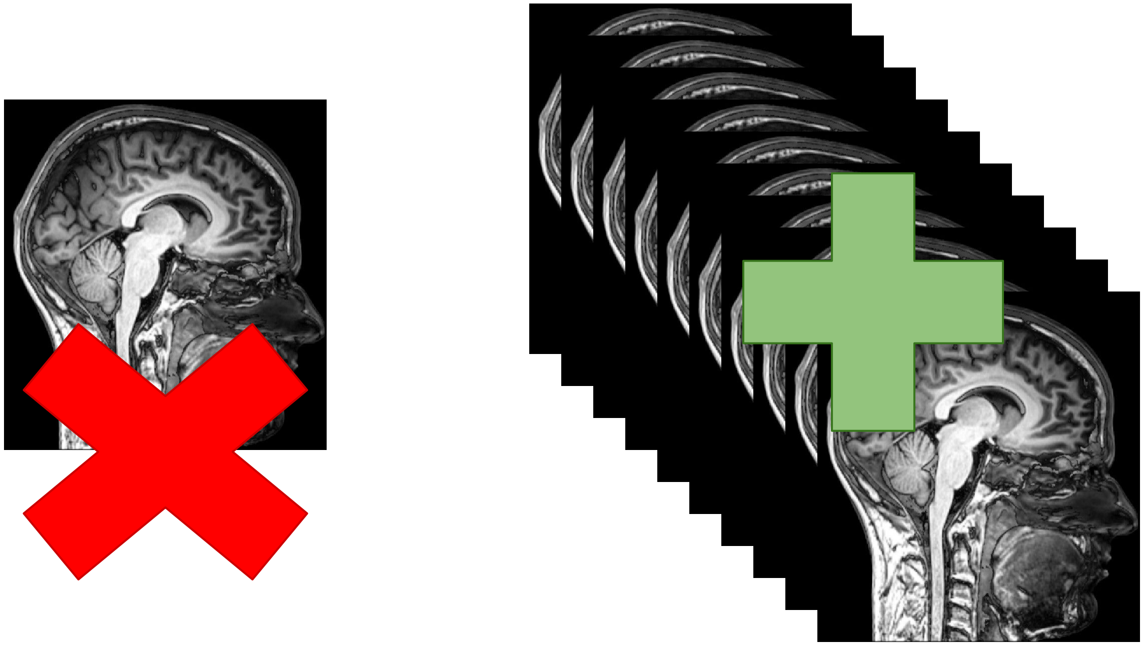


Image credits:
https://vectorportal.com/vector/hospital/34993
https://commons.wikimedia.org/wiki/File:MRI_brain_sagittal_section.jpg

Are you in the stack?

Example: https://huggingface.co/spaces/bigcode/in-the-stack

Instruction: Create a SQL query to get the list of employee names and ids with a monthly income greater than 4,000.
Input: n/a
Output:

```
SELECT id, name FROM Employees WHERE
monthly_income > 4000;
```

Instruction: Write a code to add two numbers without using the \"+\" operator.
Input:

```
num1 = 2
num2 = 7
```

Output:

```
num1 = 2
num2 = 7
# Without using \"+\" operator
sum = num1 - (-num2)
```

Instruction: Optimize the given Python program to improve the speed of execution.

Input:

```
def calc( num1, num2):
    result = 0
    for i in range(num1, num2):
       result += ((i+1) * (i+2))
    return result
```

Output:

```
def calc(num1, num2):
    result = (num1 + num2) * (num2
- num1 + 1) // 2
    return result
```

Example from https://arxiv.org/pdf/2305.07922

Basic pattern

Differential Privacy:

"Stochastic algorithm such that individual contribution is blurred, but collective patterns can be learned."

Practical use of Differential Privacy

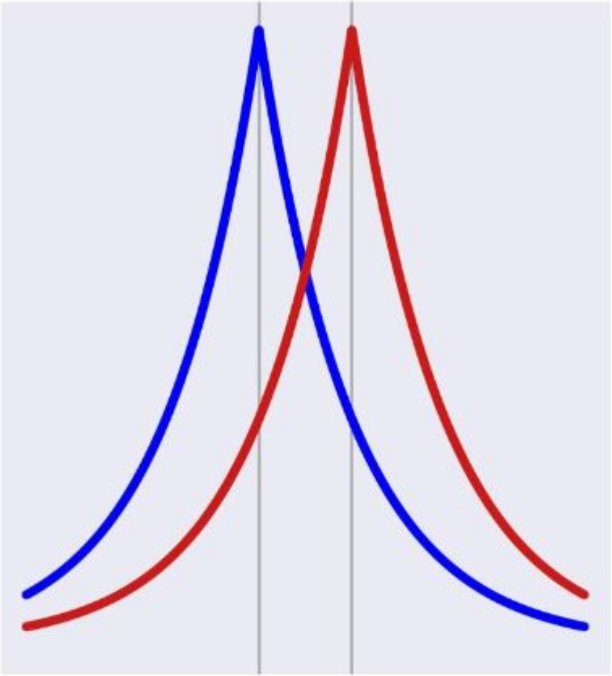
- Emoji suggestions at **Apple**
- QuickType suggestions at **Apple**
- **US Census** releases data under DP
- Executive order US gov. mentions Differential Privacy multiple times
- **Governments** releasing birth rate data
- **Facebook** releases mobility data of users during covid pandemic
- **Google GBoard** language next word prediction
- **LinkedIn** user analytics
- Telemetry on **Windows**

Ponomareva et al. JAIR 2023
Apple blog on DP
Executive order US white house
Facebook Mobility, 2020
LinkedIn (slides, post)
Windows telemetry (NeurIPS paper)
Israeli Birthrate release
Other great overview

Differential Privacy

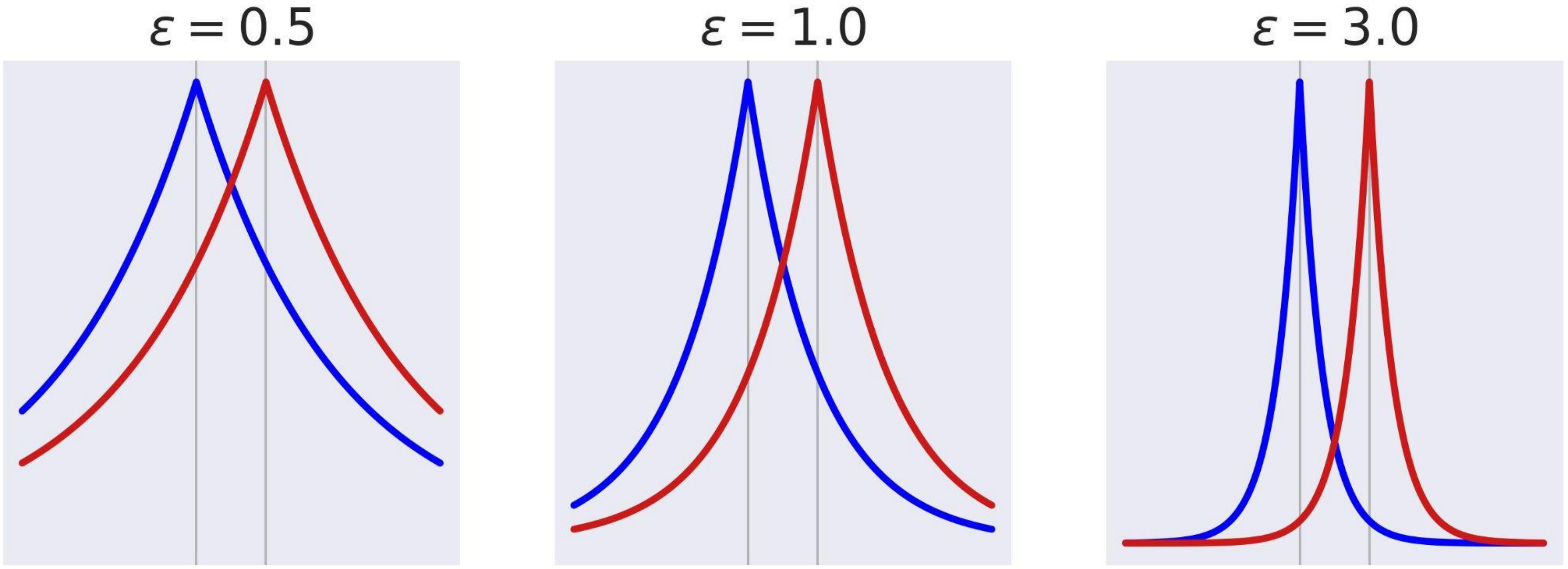
An Algorithm $A(\cdot)$ is (ε, δ)-DP when for any two adjacent data sets (D, D') that differ in at most one element, and for any subset of outcomes W, the following inequality holds:

$$\Pr[A(D) \in W] \leq e^{\varepsilon} \Pr[A(D') \in W] + \delta$$



Dwork & Roth, 2014, "Algorithmic Foundations of Differential Privacy"

Noise proportional to inverse epsilon



Problem setting

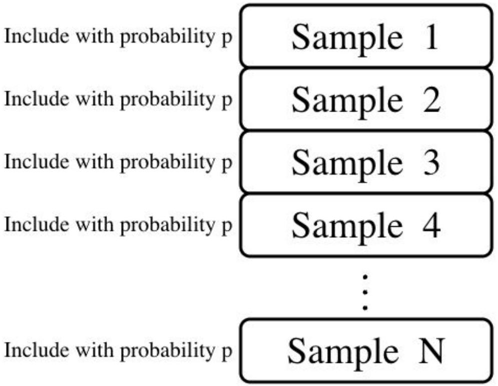
- In *common DP-SGD*, all intermediate models are released.
  In the **hidden state threat model**, release only the final model.

- Two problems with *common DP-SGD*
  - Samples are ignored (statistical inefficiency)
  - Compute is wasted (computational inefficiency)

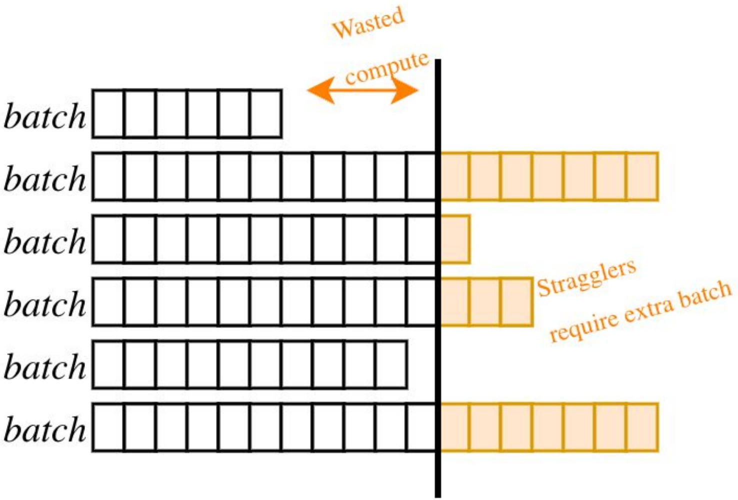
Computational problems with DP-SGD

Problem 1:
For 'poisson sampling,' some samples are repeated. Some are ignored.

In an epoch, more than 30% of data remains unused



Problem 2:
Compute is wasted with unevenly-sized mini-batches



NeurIPS 2025, arxiv.org/abs/2407.04884

State of literature

- Feldman et al. (2018) introduce Privacy Amplification by Iteration
- Bok et al. (2024) generalize this to f-DP characterization
- Ye and Shokri (2022) analyse shuffled mini-batch DP-SGD.

- This work: first non-linear function learning with hidden-state DP

Why is data ignored?

Include a sample with probability $p \in (0,1)$. For example, $p = \frac{64}{100{,}000} = 0.00064$
In one draw, the probability that a specific element is NOT selected is:

$$P(\text{not selected in one draw}) = 1 - p$$

Assumption: consider one epocht that is $\frac{1}{p} = \frac{100{,}000}{64}$ steps:

$$P(\text{never selected}) = (1-p)^N$$

There is a famous limit:

$$\lim_{n\to\infty}\left(1 - \frac{1}{n}\right)^n = \frac{1}{e}$$

Therefore, for small p:

$$\lim_{p^{-1}\to\infty}\left(1 - \frac{1}{p^{-1}}\right)^{p^{-1}} = \frac{1}{e}$$

$$P(\text{never selected}) \approx e^{-1} = \frac{1}{e}$$

Expected number of unique elements:

$$\begin{aligned} E[\text{unique elements}] &= 100{,}000 \times P(\text{selected at least once}) \\ &= 100{,}000 \times \left(1 - \frac{1}{e}\right) \approx \mathbf{63{,}200} \text{ unique elements} \end{aligned}$$

Illustrate DP-SGD stragglers

```
from scipy import stats
print(1 - stats.poisson.cdf(256, mu=230))
>> 0.04


# Inefficient: 230 instead of 256
# Stragglers: 4%
```

Truncate stragglers?

Scalable DP-SGD: Shuffling vs. Poisson Subsampling

Lynn Chua
Google Research
chualynn@google.com

Badih Ghazi
Google Research
badihghazi@gmail.com

Pritish Kamath
Google Research
pritishk@google.com

Ravi Kumar
Google Research
ravi.k53@gmail.com

Pasin Manurangsi
Google Research
pasin@google.com

Amer Sinha
Google Research
amersinha@google.com

Chiyuan Zhang
Google Research
chiyuan@google.com

“Scalable DP-SGD: Shuffling vs. poisson subsampling”
Chua et al. NeurIPS 2024

Proposition 3.2. *For distributions P, P', Q, Q' such that $d_{\mathrm{TV}}(P, P'), d_{\mathrm{TV}}(Q, Q') \le \eta$, and $D_{e^\varepsilon}(P' \| Q') \le \delta$, then $D_{e^\varepsilon}(P \| Q) \le \delta + \eta(1 + e^\varepsilon)$.*

Proof. For any event Γ we have that

$$P(\Gamma) \overset{\text{(i)}}{\le} P'(\Gamma) + \eta \overset{\text{(ii)}}{\le} e^\varepsilon Q'(\Gamma) + \delta + \eta \overset{\text{(iii)}}{\le} e^\varepsilon (Q(\Gamma) + \eta) + \delta + \eta \;=\; e^\varepsilon Q(\Gamma) + \delta + \eta(1 + e^\varepsilon),$$

where (i) follows from $d_{\mathrm{TV}}(P, P') \le \eta$, (ii) follows from $D_{e^\varepsilon}(P' \| Q') \le \delta$ and (iii) follows from $d_{\mathrm{TV}}(Q, Q') \le \eta$. Thus, we get that $D_{e^\varepsilon}(P \| Q) \le \delta + \eta(1 + e^\varepsilon)$. □

The batch size $|S_t|$ before truncation in $\mathcal{P}_{b,B,T}$ is distributed as the binomial distribution $\mathrm{Bin}(n, b/n)$, and thus, by a union bound over the events that the sampled batch size $|S_t| > B$ at any step, it follows that for any input dataset $\boldsymbol{x}$,

$$d_{\mathrm{TV}}(\mathsf{ABLQ}_{\mathcal{P}_{b,B,T}}(\boldsymbol{x}), \mathsf{ABLQ}_{\mathcal{P}_{b,\infty,T}}(\boldsymbol{x})) \le T \cdot \Psi(n, b, B),$$

where $\Psi(n, b, B) := \Pr_{r \sim \mathrm{Bin}(n, b/n)}[r > B]$. Applying Proposition 3.2 we get

Theorem 3.3. *For all $\sigma > 0$, $\varepsilon \ge 0$, and integers b, $n \ge b$, $B \ge b$, T, it holds that*

$$\delta_{\mathcal{P}}(\varepsilon) \le \max\{D_{e^\varepsilon}(P_{\mathcal{P}} \| Q_{\mathcal{P}}), D_{e^\varepsilon}(Q_{\mathcal{P}} \| P_{\mathcal{P}})\} + T \cdot (1 + e^\varepsilon) \cdot \Psi(n, b, B).$$

```
n = 1_000_000; b = 166; B = 256
T = 10; eps = 8.0

print(T*(1+np.exp(eps)) *
(1-stats.binom.cdf(k=B, n=n, p=b/n)))
>> 1e-6
```

Some work for DP-SGD

- A recent line of work considers DP guarantees for DP-SGD with disjoint batches:

$$\theta_{j+1} = \theta_j - \eta \left(\frac{1}{b} \sum_{x \in B_j} \nabla_\theta f(\theta_j, x) + Z_j \right)$$

where $Z_j \sim \mathcal{N}(0, \sigma^2 I_d)$ and the data is divided into disjoint batches B_j.

Chua et al. (2025), Choquette-Choo et al. (2025), Feldman and Shenfeld (2025).
however, do not accommodate fixed-size batches and, in particular, unshuffled data.

Convex 2-layer neural net

Consider training a ReLU network (with hidden-width m) $f : \mathbb{R}^d \to \mathbb{R}$ (Pilanci and Ergen, 2020),

$$f(x) = \sum_{j=1}^{m} \phi(u_j^T x)\alpha_j. \tag{3.1}$$

The weights are $u_i \in \mathbb{R}^d$, $i \in [m]$ and $\alpha \in \mathbb{R}^m$. The ReLU activation function is $\phi(t) = \max\{0, t\}$. For a vector x, ϕ is applied element-wise, i.e. $\phi(x)_i = \phi(x_i)$.

Suppose the dataset D consists of n tuples of the form $z_i = (x_i, y_i)$, $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, for $i \in [n]$. Using the squared loss and L_2-regularization with a regularization constant $\lambda > 0$, the 2-layer ReLU minimization problem can be written as

$$\min_{\{u_i,\alpha_i\}_{i=1}^m} \frac{1}{2} \left\| \sum_{i=1}^{m} \phi(Xu_i)\alpha_i - y \right\|_2^2 + \frac{\lambda}{2} \sum_{i=1}^{m} (\|u_i\|_2^2 + \alpha_i^2), \tag{3.2}$$

where $X \in \mathbb{R}^{n \times d}$ denotes the matrix of the feature vectors, i.e., $X^T = [x_1 \quad \ldots \quad x_n]$ and $y \in \mathbb{R}^n$ denotes the vector of labels.

The convex reformulation is based on enumerating all the possible activation patterns of $\phi(Xu)$, $u \in \mathbb{R}^d$. The set of activation patterns that a ReLU output $\phi(Xu)$ can take for a data feature matrix $X \in \mathbb{R}^{n \times d}$ is described by the set of diagonal boolean matrices

$$\mathcal{D}_X = \{\Lambda = \operatorname{diag}(\mathbb{1}(Xu \geq 0)) \,:\, u \in \mathbb{R}^d\}, \tag{3.3}$$

<< Draw data an Relu activations on the board >>

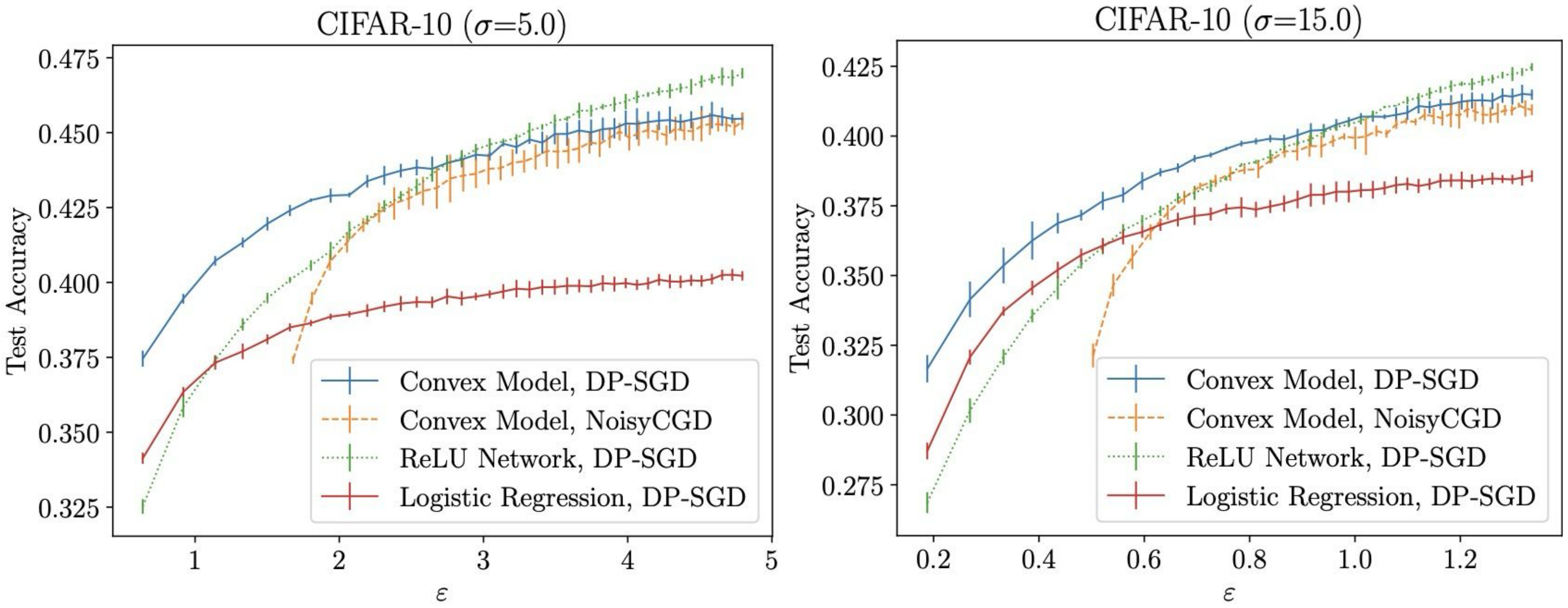


Figure 3: CIFAR10 Test accuracy versus the spent privacy budget ε, when each model is trained for 400 epochs. NoisyCGD and DP-SGD generally have comparable performance for the 2-layer ReLU network and much higher accuracy than logistic regression.

Comparable results between Noisy-CGD and DP-SGD

Table 1: A comparison of model accuracies vs. ε-values. The iterative methods generally score better than SSP and have comparable accuracy, which shows a high-utility result for hidden-state privacy analysis of a 2-layer neural network. The results are the mean accuracy among five random restarts.

	MNIST		CIFAR-10	
	$\varepsilon = 1.33$	$\varepsilon = 4.76$	$\varepsilon = 1.33$	$\varepsilon = 4.76$
Sufficient Statistics Perturbation (Convex Approx.)	$51.9_{\pm 1.1}$	$67.0_{\pm 0.2}$	$19.2_{\pm 1.1}$	$23.3_{\pm 0.9}$
Sufficient Statistics Perturbation (Random ReLU)	$52.5_{\pm 0.9}$	$65.6_{\pm 0.3}$	$19.3_{\pm 0.2}$	$25.9_{\pm 0.3}$
Sufficient Statistics Perturbation (RFF)	$64.1_{\pm 0.9}$	$77.4_{\pm 0.2}$	$21.3_{\pm 0.5}$	$28.5_{\pm 0.3}$
DP-SGD + Convex Approximation	$93.1_{\pm 0.1}$	$94.9_{\pm 0.1}$	$41.5_{\pm 0.2}$	$45.5_{\pm 0.2}$
DP-SGD + ReLU	$91.7_{\pm 0.1}$	$94.3_{\pm 0.1}$	$42.5_{\pm 0.1}$	$47.0_{\pm 0.2}$
NoisyCGD + Convex Approximation	$92.4_{\pm 0.2}$	$94.4_{\pm 0.1}$	$41.0_{\pm 0.2}$	$45.4_{\pm 0.3}$

NeurIPS 2025, arxiv.org/abs/2407.04884

Conclusion

DP-SGD has two problems and one quirk:

- Poisson sampling: some data are ignored, some oversampled
- Computational inefficiency: varying batch sizes
- Hidden-states accounted for but not released

Our work:

- First non-linear function learning with hidden-state DP
- However: one 2-layer net, and results are only on-par