Sharpness-Aware Private Aggregation of Teacher Ensembles (Sharp-PATE)

Momin Abbas

Department of Electrical, Computer, and Systems Engineering

Rensselaer Polytechnic Institute

email: abbasm2@rpi.edu

April 22, 2022

Motivation

- Some machine learning applications involve training data that is **sensitive**. Examples:
  - users' personal contacts, private photographs, medical records or genetic sequences
- **Learning algorithms should protect the privacy of users' training data**, e.g., by guaranteeing that the output model generalizes away from the specifics of any individual use
- However, recent attacks methods have demonstrated that **private training data can be recovered from models** (Fredrikson et al., 2015)

Background

- Attempts to provide privacy for machine learning models led to a series of efforts
- Privacy-preserving distributed SGD algorithm was introduced by (Shokri and Shmatikov, 2015)
  - Problem: Does not provide a meaningful privacy guarantee
- (Abadi et al., 2016) introduced a differentially private SGD (DP-SGD) algorithm with much stricter privacy bounds
  - Problem: accuracy as well as the privacy cost are still not too convincing

The Problem

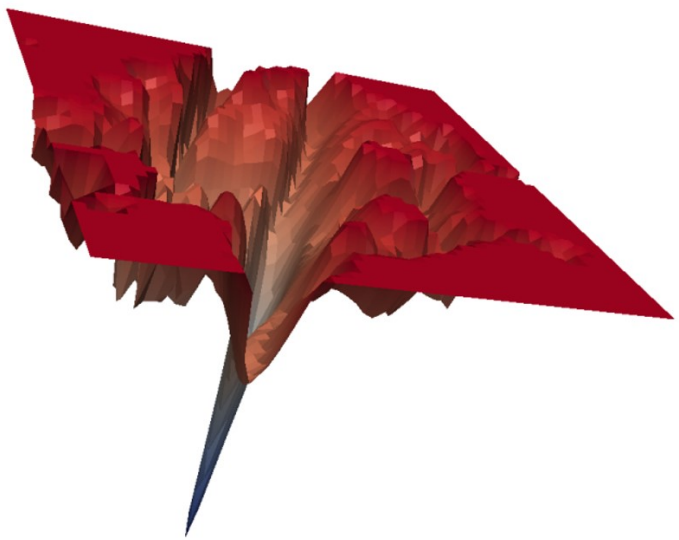
Can we achieve better accuracy along with improved privacy guarantees?

- **PATE** algorithm (Papernot et al., 2016) achieves this via a teacher-student framework.
- We ask: can we further improve the privacy/utility trade-offs of PATE?
  - We leverage the recently proposed **Sharpness-aware minimization (SAM)** (Foret et al., 2020) that **improves the generalization performance** of ML models
  - We propose **Sharp-PATE**: a **Sharp**ness-Aware **P**rivate **A**ggregation of **T**eacher **E**nsembles method

SAM

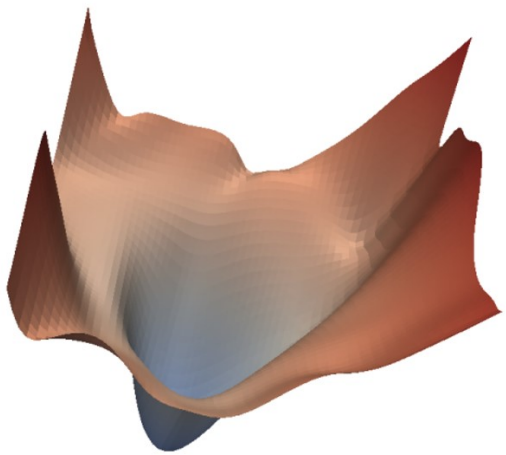
- SAM improves the generalization performance by leveraging the geometry of the loss landscape
- It has been shown that a wide minima achieves better generalization as compared to a sharp minima (Keskar et al. (2016))

Sharp minima -> degraded test performance



SGD

Flat minima -> Better test performance



SAM

SAM

- SAM improves the generalization performance by leveraging the geometry of the loss landscape
- It has been shown that a wide minima achieves better generalization as compared to a sharp minima (Keskar et al. (2016))
- SAM avoids sharp minima via the following optimization problem:

$$\min_{\theta} \max_{||\delta||_2 \leq \rho} L(\theta + \delta) \tag{1}$$

where, $\rho \geq 0$

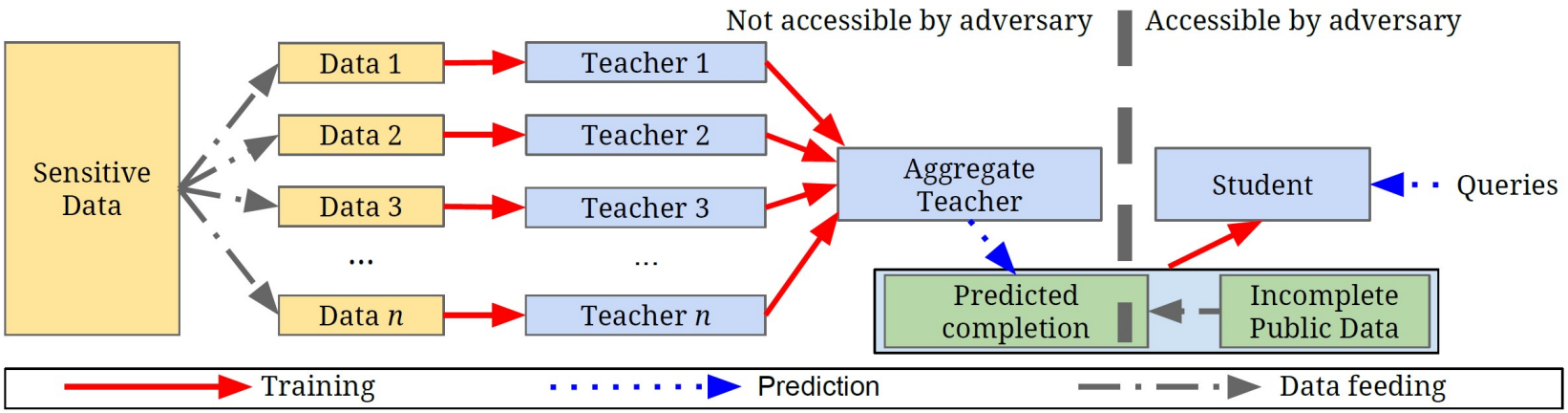
- δ is approximated as:

$$\hat{\delta} = \rho \frac{\nabla_\theta \mathcal{L}(\theta)}{||\nabla_\theta \mathcal{L}(\theta)||_2} \tag{2}$$

- Finally, we obtain our final gradient approximation:

$$\nabla_\theta \mathcal{L}(\theta)|_{\theta+\hat{\delta}} \tag{3}$$

PATE Algorithm



PATE Algorithm

Algorithm 1 PATE

Require: step sizes α, β, No. of teachers N, Teachers' data $\{X_i, Y_i\}_{i=1}^{N}$, Student's data X_s, noise ϵ
Initialize: teachers' models $\{\theta_i\}_{i=1}^{N}$, Student model θ_s, $t = 0$
for $i = 1, 2, \ldots,$ N **do**
    while not converged **do**
        Sample batch $\mathcal{B} = \{(X_i^1, Y_i^1), \ldots, (X_i^b, Y_i^b)\}$
        Compute gradient $g^t = \nabla_{\theta_i} \mathcal{L}_{\mathcal{B}}(\theta_i)$ using batch's training loss
        Update the weights: $\theta_i^{t+1} = \theta_i^t - \alpha g^t$
        $t = t + 1$
    end while
end for
Get student labels $Y_s = aggregated\_teacher(\{\theta_i\}_{i=1}^{N}, X_s, \epsilon)$
$t = 0$
while not converged **do**
    Sample batch $\mathcal{B}_s = \{(X_s^1, Y_s^1), \ldots, (X_s^b, Y_s^b)\}$
    Compute gradient $g_s^t = \nabla_{\theta_s} \mathcal{L}_{\mathcal{B}_s}(\theta_s)$ using batch's training loss
    Update the weights: $\theta_s^{t+1} = \theta_s^t - \beta g_s^t$
    $t = t + 1$
end while

PATE Algorithm

Algorithm 1 PATE

Require: step sizes α, β, No. of teachers N, Teachers' data $\{X_i, Y_i\}_{i=1}^{N}$, Student's data X_s, noise ϵ

Initialize: teachers' models $\{\theta_i\}_{i=1}^{N}$, Student model θ_s, $t = 0$

for $i = 1, 2, \ldots,$ N **do**

  while not converged **do**

    Sample batch $\mathcal{B} = \{(X_i^1, Y_i^1), \ldots, (X_i^b, Y_i^b)\}$

    Compute gradient $g^t = \nabla_{\theta_i} \mathcal{L}_{\mathcal{B}}(\theta_i)$ using batch's training loss

    Update the weights: $\theta_i^{t+1} = \theta_i^t - \alpha g^t$

    $t = t + 1$

  end while

end for

Get student labels $Y_s = aggregated\_teacher(\{\theta_i\}_{i=1}^{N}, X_s, \epsilon)$

$t = 0$

while not converged **do**

  Sample batch $\mathcal{B}_s = \{(X_s^1, Y_s^1), \ldots, (X_s^b, Y_s^b)\}$

  Compute gradient $g_s^t = \nabla_{\theta_s} \mathcal{L}_{\mathcal{B}_s}(\theta_s)$ using batch's training loss

  Update the weights: $\theta_s^{t+1} = \theta_s^t - \beta g_s^t$

  $t = t + 1$

end while

PATE Algorithm

Algorithm 1 PATE

Require: step sizes α, β, No. of teachers N, Teachers' data $\{X_i, Y_i\}_{i=1}^{N}$, Student's data X_s, noise ϵ

Initialize: teachers' models $\{\theta_i\}_{i=1}^{N}$, Student model θ_s, $t = 0$

for $i = 1, 2, \ldots,$ N **do**

  while not converged **do**

    Sample batch $\mathcal{B} = \{(X_i^1, Y_i^1), \ldots, (X_i^b, Y_i^b)\}$

    Compute gradient $g^t = \nabla_{\theta_i} \mathcal{L}_{\mathcal{B}}(\theta_i)$ using batch's training loss

    Update the weights: $\theta_i^{t+1} = \theta_i^t - \alpha g^t$

    $t = t + 1$

  end while

end for

1) Train N **teacher models** using ***private*** **data**

Get student labels $Y_s = aggregated\_teacher(\{\theta_i\}_{i=1}^{N}, X_s, \epsilon)$

$t = 0$

while not converged **do**

  Sample batch $\mathcal{B}_s = \{(X_s^1, Y_s^1), \ldots, (X_s^b, Y_s^b)\}$

  Compute gradient $g_s^t = \nabla_{\theta_s} \mathcal{L}_{\mathcal{B}_s}(\theta_s)$ using batch's training loss

  Update the weights: $\theta_s^{t+1} = \theta_s^t - \beta g_s^t$

  $t = t + 1$

end while

PATE Algorithm

Algorithm 1 PATE

Require: step sizes α, β, No. of teachers N, Teachers' data $\{X_i, Y_i\}_{i=1}^{N}$, Student's data X_s, noise ϵ

Initialize: teachers' models $\{\theta_i\}_{i=1}^{N}$, Student model θ_s, $t = 0$

for $i = 1, 2, \ldots,$ N **do**

  while not converged **do**

    Sample batch $\mathcal{B} = \{(X_i^1, Y_i^1), \ldots, (X_i^b, Y_i^b)\}$

    Compute gradient $g^t = \nabla_{\theta_i} \mathcal{L}_{\mathcal{B}}(\theta_i)$ using batch's training loss

    Update the weights: $\theta_i^{t+1} = \theta_i^t - \alpha g^t$

    $t = t + 1$

  end while

end for

Get student labels $Y_s = aggregated\_teacher(\{\theta_i\}_{i=1}^{N}, X_s, \epsilon)$

$t = 0$

while not converged **do**

  Sample batch $\mathcal{B}_s = \{(X_s^1, Y_s^1), \ldots, (X_s^b, Y_s^b)\}$

  Compute gradient $g_s^t = \nabla_{\theta_s} \mathcal{L}_{\mathcal{B}_s}(\theta_s)$ using batch's training loss

  Update the weights: $\theta_s^{t+1} = \theta_s^t - \beta g_s^t$

  $t = t + 1$

end while

1) Train N **teacher models** using ***private*** **data**

2) **Get student labels using ensemble of teachers**

PATE Algorithm

Algorithm 1 PATE

Require: step sizes α, β, No. of teachers N, Teachers' data $\{X_i, Y_i\}_{i=1}^{N}$, Student's data X_s, noise ϵ

Initialize: teachers' models $\{\theta_i\}_{i=1}^{N}$, Student model θ_s, $t = 0$

for $i = 1, 2, \ldots,$ N **do**

  while not converged **do**

    Sample batch $\mathcal{B} = \{(X_i^1, Y_i^1), \ldots, (X_i^b, Y_i^b)\}$

    Compute gradient $g^t = \nabla_{\theta_i} \mathcal{L}_{\mathcal{B}}(\theta_i)$ using batch's training loss

    Update the weights: $\theta_i^{t+1} = \theta_i^t - \alpha g^t$

    $t = t + 1$

  end while

end for

Get student labels $Y_s = aggregated\_teacher(\{\theta_i\}_{i=1}^{N}, X_s, \epsilon)$

$t = 0$

while not converged **do**

  Sample batch $\mathcal{B}_s = \{(X_s^1, Y_s^1), \ldots, (X_s^b, Y_s^b)\}$

  Compute gradient $g_s^t = \nabla_{\theta_s} \mathcal{L}_{\mathcal{B}_s}(\theta_s)$ using batch's training loss

  Update the weights: $\theta_s^{t+1} = \theta_s^t - \beta g_s^t$

  $t = t + 1$

end while

1) Train N **teacher models** using ***private*** **data**

2) **Get student labels using ensemble of teachers**

3) Train **Student Model** using **public data**

Sharp-PATE Algorithm

Algorithm 2 Sharp-PATE. (Colored lines reflect changes to the original)

Require: step sizes α, β, No. of teachers N, Teachers' data $\{X_i, Y_i\}_{i=1}^{N}$, Student's data X_s, noise ϵ, SAM perturbation radius ρ

Initialize: teachers' models $\{\theta_i\}_{i=1}^{N}$, Student model θ_s, $t = 0$

for $i = 1, 2, \ldots,$ N do

TEACHERS' training loop

  while not converged do

    Sample batch $\mathcal{B} = \{(X_i^1, Y_i^1), \ldots, (X_i^b, Y_i^b)\}$

    Compute gradient $\nabla_{\theta_i} \mathcal{L}_{\mathcal{B}}(\theta_i)$ using batch's training loss

    Compute $\hat{\delta}$ per equation (2)

    Compute gradient approximation for SAM objective (equation (3)): $\mathbf{g}^t = \nabla_{\theta_i} \mathcal{L}_{\mathcal{B}}(\theta_i)|_{\theta_i + \hat{\delta}}$

    Update the weights: $\theta_i^{t+1} = \theta_i^t - \alpha g^t$

    $t = t + 1$

  end while

end for

Get student labels $Y_s = aggregated\_teacher(\{\theta_i\}_{i=1}^{N}, X_s, \epsilon)$

$t = 0$

while not converged do

Student's training loop

  Sample batch $\mathcal{B}_s = \{(X_s^1, Y_s^1), \ldots, (X_s^b, Y_s^b)\}$

  Compute gradient $\nabla_{\theta_s} \mathcal{L}_{\mathcal{B}_s}(\theta_s)$ using batch's training loss

  Compute $\hat{\delta}$ per equation (2)

  Compute gradient approximation for SAM objective (equation (3)): $g_s^t = \nabla_{\theta_s} \mathcal{L}_{\mathcal{B}_s}(\theta_s)|_{\theta_s + \hat{\delta}}$

  Update the weights: $\theta_s^{t+1} = \theta_s^t - \beta g_s^t$

  $t = t + 1$

end while

Sharp-PATE Algorithm

Algorithm 2 Sharp-PATE. (Colored lines reflect changes to the original)

Require: step sizes α, β, No. of teachers N, Teachers' data $\{X_i, Y_i\}_{i=1}^{N}$, Student's data X_s, noise ϵ, SAM perturbation radius ρ

Initialize: teachers' models $\{\theta_i\}_{i=1}^{N}$, Student model θ_s, $t = 0$

for $i = 1, 2, \ldots,$ N **do**

TEACHERS' training loop

  while not converged **do**

    Sample batch $\mathcal{B} = \{(X_i^1, Y_i^1), \ldots, (X_i^b, Y_i^b)\}$

    Compute gradient $\nabla_{\theta_i} \mathcal{L}_{\mathcal{B}}(\theta_i)$ using batch's training loss

    Compute $\hat{\delta}$ per equation (2) → $\hat{\delta} = \rho \frac{\nabla_\theta \mathcal{L}(\theta)}{||\nabla_\theta \mathcal{L}(\theta)||_2}$

    Compute gradient approximation for SAM objective (equation (3)): $\mathbf{g}^t = \nabla_{\theta_i} \mathcal{L}_{\mathcal{B}}(\theta_i)|_{\theta_i + \hat{\delta}}$

    Update the weights: $\theta_i^{t+1} = \theta_i^t - \alpha g^t$

    $t = t + 1$

  end while

end for

Get student labels $Y_s = aggregated\_teacher(\{\theta_i\}_{i=1}^{N}, X_s, \epsilon)$

$t = 0$

while not converged do

Student's training loop

  Sample batch $\mathcal{B}_s = \{(X_s^1, Y_s^1), \ldots, (X_s^b, Y_s^b)\}$

  Compute gradient $\nabla_{\theta_s} \mathcal{L}_{\mathcal{B}_s}(\theta_s)$ using batch's training loss

  Compute $\hat{\delta}$ per equation (2) → $\hat{\delta} = \rho \frac{\nabla_\theta \mathcal{L}(\theta)}{||\nabla_\theta \mathcal{L}(\theta)||_2}$

  Compute gradient approximation for SAM objective (equation (3)): $g_s^t = \nabla_{\theta_s} \mathcal{L}_{\mathcal{B}_s}(\theta_s)|_{\theta_s + \hat{\delta}}$

  Update the weights: $\theta_s^{t+1} = \theta_s^t - \beta g_s^t$

  $t = t + 1$

end while

Rensselaer

Experiments: Settings

- Datasets: SVHN and MNIST
- Metrics: Student's test accuracy and (ϵ, δ)-privacy
- Baselines:
  1. Non-private model
  2. PATE

Experiments: Results

Algorithm	Student Accuracy	ε	δ
PATE (reproduced)	95.80	6.69	1e-5
Non-Private Baseline	99.18	-	-
Sharp-PATE (Teacher)	95.89	6.70	1e-5
Sharp-PATE (Student)	95.95	6.69	1e-5
Sharp-PATE (Both)	96.10	6.67	1e-5
aSHARP-PATE (Both)	96.30	6.68	1e-5

Figure. Results on MNIST

aSharp-PATE uses the **Adaptive-SAM optimizer (Kwon et al., 2021) instead of the original SAM optimizer

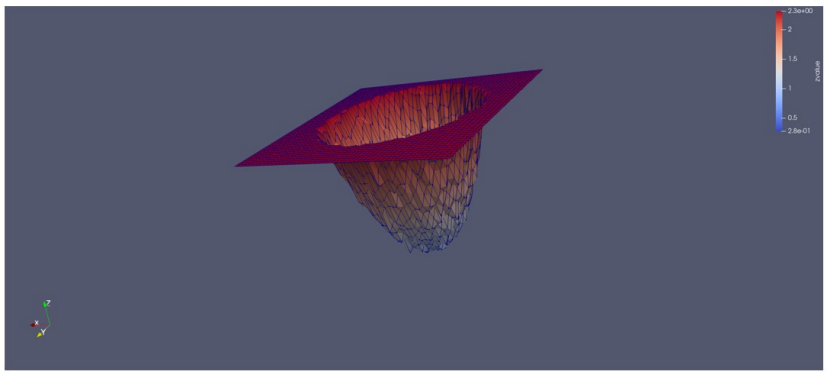
Experiments: Results

Algorithm	Student Accuracy	ε	δ
PATE (reproduced)	71.40	7.16	1e-6
Non-Private Baseline	92.80	-	-
Sharp-PATE (Teacher)	73.30	7.15	1e-6
Sharp-PATE (Student)	74.30	7.14	1e-6
Sharp-PATE (Both)	74.52	7.15	1e-6
aSHARP-PATE (Both)	75.19	7.16	1e-6

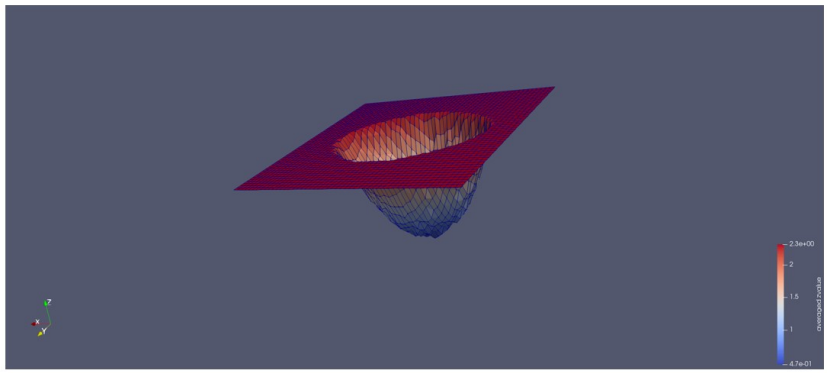
Figure. Results on SVHN

aSharp-PATE uses the **Adaptive-SAM optimizer (Kwon et al., 2021) instead of the original SAM optimizer

Experiments: Results



PATE



Sharp-PATE (both)

Figure. Loss landscapes for PATE and Sharp-PATE (both)

**landscapes are generated using the ParaView software following the filter normalization method proposed in (Li et al., 2018)

Rensselaer

Experiments: Results

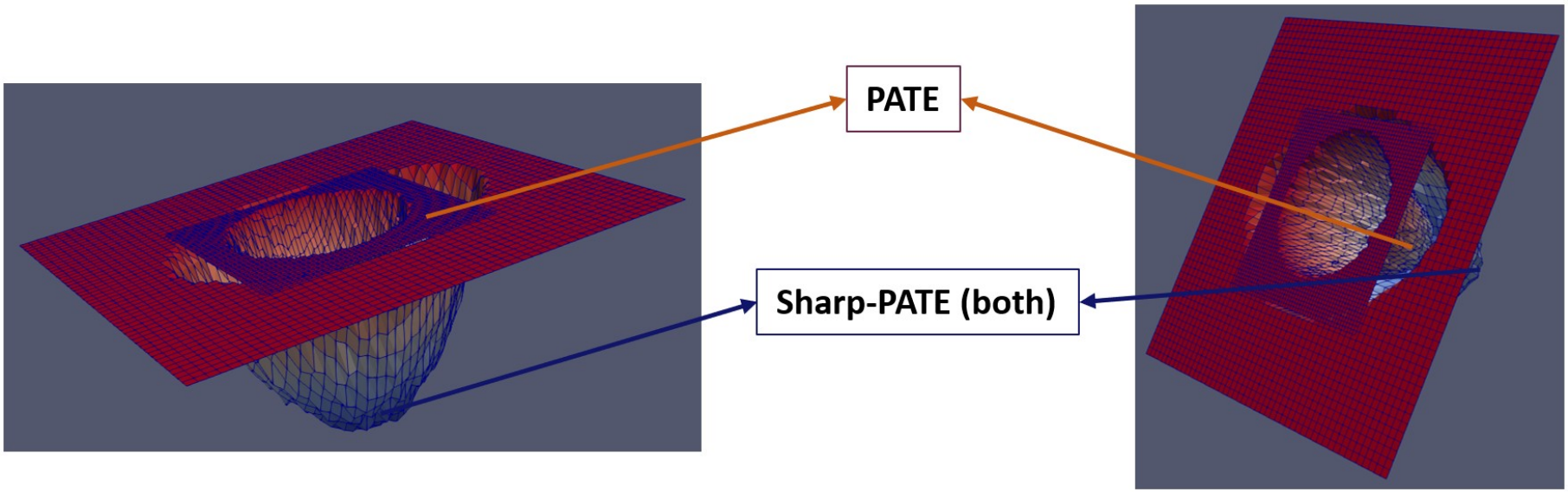


Figure. Loss Landscapes (Top View)

**landscapes are generated using the ParaView software following the filter normalization method proposed in (Li et al., 2018)

Key-Takeaways

- **Sharp-PATE improves the student's test accuracy** as compared to PATE
- The **aSharp-PATE further improves the student's test accuracy** as compared to PATE
- Sharp-PATE does **not have a significant effect on the privacy cost**
- **Loss landscapes justify the improved test performance** of Sharp-PATE as compared to PATE

Conclusion and Future Work

- Given the success of SAM, we proposed Sharp-PATE that leverages SAM to improve the original PATE
- Sharp-PATE and its variant improved the performance of the original PATE in terms of the test accuracy
- We further proposed a new variant, aSharp-PATE, that further improves the test accuracy
- Future directions:
  - We use the Laplacian noise to the aggregate answers used to train the student. It would be interesting to see the affect of other types of noises e.g. the Gaussian noise
  - We have so far evaluated only on simple classification tasks like MNIST and SVHN. It would be interesting to investigate the utility of our method when applied to larger-scale learning tasks and real-world datasets
- **CODE: https://github.com/mominabbass/Sharp-PATE**

Thank you!
Questions?

Reference I

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *Proc. International Conference on Learning Representations*, Addis Ababa, Ethiopia, April 2020.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450338325. doi:

Reference II

10.1145/2810103.2813677. URL https://doi.org/10.1145/2810103.2813677.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, pages 5905–5914. PMLR, 2021.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Reference III

Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1310–1321, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450338325. doi: 10.1145/2810103.2813687. URL https://doi.org/10.1145/2810103.2813687.