USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

Sharpness-Aware Initialization: Improving Differentially Private Machine Learning from First Principles

Zihao Wang, Rui Zhu, and Dongruo Zhou, *Indiana University Bloomington;* Zhikun Zhang, *Zhejiang University;* XiaoFeng Wang, *Nanyang Technological University;* Haixu Tang, *Indiana University Bloomington*

https://www.usenix.org/conference/usenixsecurity25/presentation/wang-zihao

This paper is included in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

Sharpness-Aware Initialization: Improving Differentially Private Machine Learning from First Principles

Zihao Wang[1]*, Rui Zhu[2]*, Dongruo Zhou[2], Zhikun Zhang[3], XiaoFeng Wang[1], and Haixu Tang[2]
[1]*Nanyang Technological University* [2]*Indiana University Bloomington* [3]*Zhejiang University*

Abstract

Recent advances in privacy-preserving machine learning underscore the critical role of differential privacy (DP) in protecting individual data. However, the noise introduced during DP training often leads to significant performance degradation, creating a major challenge for differentially private machine learning (DPML).

In this work, we address this challenge by controlling the detrimental effects of DP noise. Specifically, we focus on enhancing a model's robustness to random perturbations, thereby mitigating their negative impact on convergence—a central factor in maintaining high utility under DP. To this end, we propose *sharpness-aware initialization* (SAI), a method for improving the accuracy of DPML algorithms by achieving a flatter loss landscape. Our approach employs a two-phase training framework: SAI followed by standard Differentially Private Stochastic Gradient Descent (DPSGD). This strategy capitalizes on the observation that loss-landscape flatness converges more rapidly than the training loss, enabling an early stop on flatness optimization to limit divergence risk, followed by a phase dedicated to training-loss optimization. Moreover, splitting the training into two distinct phases allows for different privacy budgets in each phase, aligning their respective optimization objectives and tolerance to DP noise, which further mitigates performance degradation. Our experimental results show that SAI substantially improves the accuracy of state-of-the-art DPML algorithms across a range of datasets and model architectures, achieving gains of over 6% on CIFAR-10 under $\varepsilon = 1$.

1 Introduction

Machine learning (ML) models may inadvertently reveal information about their training data due to limitations such as overfitting [64] and data memorization [12, 30]. Mitigating this privacy risk largely relies on Differential Privacy (DP) [25], which has been applied to protect diverse ML tasks [19, 44, 47, 79]. In differentially private machine learning (DPML), a randomized algorithm injects noise during training so that any single example in the dataset minimally affects the model's output distributions. This privacy guarantee is expressed as (ε, δ)-DP, where the *privacy budget* ε and δ determine the amount of noise added. Smaller ε and δ lead to heavier noise, making it more difficult for an adversary to infer the presence of a specific data point in the training set. DP models offer strong protections against membership inference [59, 62], training data extraction [12], and data poisoning [46].

Challenges in DPML. One of the most widely used DPML algorithms is Differentially Private Stochastic Gradient Descent (DPSGD) [1], which (1) clips per-example gradients to limit their sensitivity and (2) adds noise before parameter updates. Various methods aim to mitigate the performance loss from these steps by exploring new architectures [15, 67], loss functions [61], activation functions [55], and clipping functions [3, 8]. They target adverse effects from *gradient clipping* [3, 8, 55], *limited iteration rounds* [61], and *DP-unfriendly architectures* [15, 67]. Despite these efforts, achieving high DPML performance remains an open problem, primarily because most approaches do not directly address the negative impacts of DP noise.

This work seeks to fundamentally enhance DPML under its privacy guarantee by improving a model's *robustness to random perturbations*, thereby reducing the harm of DP noise. This direction focuses on the core of differential privacy—namely, the injected noise. Several pioneering studies have ventured down this path. Wang et al. [70] proposed DPAdapter, which uses sharpness-aware minimization (SAM) [29] on public data to improve parameter robustness and flatten the loss landscape before fine-tuning on private data. However, it relies on auxiliary data and can be computationally expensive. Park et al. [56] introduced Differentially Private Sharpness-Aware Training (DPSAT), integrating SAM into DPML without public data. Yet, it can get stuck in suboptimal solutions due to the divergence of robustness/flatness (see Section 3.1). Thus, leveraging parameter robustness

*The first two authors contributed equally to this work.

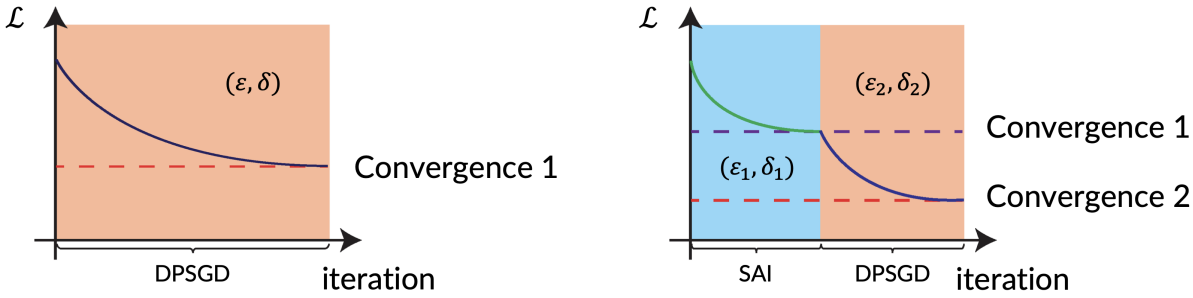

Figure 1: Overview of SAI-DPSGD. SAI-DPSGD utilizes a two-phase approach: the first phase focuses on finding a flatter loss landscape, while the second phase involves training with standard DPSGD for further convergence. The privacy budget (ε, δ) is allocated into $(\varepsilon_1, \delta_1)$ for the sharpness-aware initialization (SAI) and $(\varepsilon_2, \delta_2)$ for DPSGD, with $\varepsilon_1 \gg \varepsilon_2$.

and loss-landscape flatness without auxiliary data remains challenging. Throughout this paper, we use "flatness" to also imply parameter robustness, since a flatter landscape indicates stronger resistance to random perturbations.

Our Solution. A key insight from our work is that flatness typically converges *much earlier* than the training loss. Consequently, we can stop flatness optimization early and then focus on the training loss, avoiding flatness divergence while still achieving sufficient training-loss convergence. Building on this observation, we propose a two-phase approach: *Sharpness-Aware Initialization* (SAI) followed by standard DPSGD (see Figure 1).

By dividing the training process into two distinct phases, we can allocate separate privacy budgets that align with each phase's sensitivity to noise. Since flatter loss landscapes better tolerate noise [56], the DPSGD phase can handle higher noise once flatness has been established. Conversely, optimizing flatness is more sensitive to noise (Section 3.1), justifying a higher budget for the SAI phase. Although DPSGD subsequently runs with relatively higher noise, the landscape remains flat enough to ensure stable convergence. Our theoretical analysis further explains why flatness converges faster than the training loss and how this two-phase strategy improves the privacy-accuracy trade-offs of DPSGD.

The key contribution of SAI-DPSGD lies in its theoretically grounded two-phase framework. While prior research has explored pre-training on public data [70], none has examined a data-driven initialization phase using the same private dataset. Although initialization is crucial in deep neural networks [50], it remains largely unexplored in DPML. We are the first to show that DPML can benefit substantially from a dedicated data-driven initialization phase, which may inspire advanced initialization technologies for further improving DPML. Crucially, our design is not merely empirical; it is guided by rigorous theoretical insights (Section 4.3), providing a robust foundation for future work on even more effective initialization strategies.

Empirical Results. We implemented SAI and evaluated its performance on the MNIST, FashionMNIST, CIFAR-10, CIFAR-100, and SVHN datasets, using a variety of model architectures. These architectures include CNN-Tanh and DP-NASNet [15], both designed specifically for DPML, GNResNet with group normalization [73] (a common replacement for batch normalization in DPML [8,10,19,67,71]), and vision transformers (ViTs) [21] pretrained on ImageNet. Our results consistently show stable accuracy improvements compared to state-of-the-art approaches. For example, SAI-DPSGD improves the accuracy of CNN-Tanh with SELU on CIFAR-10 by over 6% for $\varepsilon = 1$, whereas most existing DPML enhancements achieve gains of less than 3% [71] under the same privacy budget.

Contributions. Our key contributions are summarized below:

- *New technique*. We propose SAI, a method that improves DPML accuracy by promoting a flatter loss landscape. The SAI-DPSGD framework achieves better privacy-accuracy trade-offs compared to existing approaches.
- *Theoretical understandings*. We analyze SAI theoretically to show why it is effective, revealing a fundamental trade-off between minimal flatness and training-loss convergence. This analysis also identifies a theoretical optimum range that guides hyperparameter choices.
- *Extensive empirical studies*. We conduct comprehensive experiments on SAI-DPSGD across diverse tasks and privacy budgets. Results show that SAI-DPSGD consistently improves the privacy-accuracy trade-off on standard DPML benchmarks and across various scenarios.

2 Background

2.1 Differentially Private Machine Learning

Differential privacy [24] (DP) is a mathematical framework designed to protect individuals' privacy when analyzing and sharing sensitive data. It allows for the extraction of useful statistical information from a population while minimizing the risk of exposing any individual's private information. The core concept of DP is to introduce a controlled amount of noise

or randomness to the data before releasing or analyzing it, thereby making it challenging to identify specific individuals within the dataset.

Formally, differential privacy can be defined as:

Definition 1 ((ε,δ)-Differential Privacy). *Given two neighboring datasets D and D' differing by one record, a randomized mechanism $\mathcal{M}$ satisfies (ε,δ)-differential privacy if*

$$Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot Pr[\mathcal{M}(D') \in S] + \delta,$$

where ε is the privacy budget, and δ is the failure probability.

Typically, a smaller ε indicates a higher privacy preserving level since it is more difficult for the attackers to distinguish between D and D'.

Composition Properties of DP. The following composition properties of DP are commonly used for building complex differentially private algorithms from simpler subroutines.

• *Sequential Composition.* Combining multiple subroutines that satisfy differential privacy for $\{\varepsilon_1, \cdots, \varepsilon_k\}$ results in a mechanism satisfying ε-differential privacy for $\varepsilon = \sum_i \varepsilon_i$.

• *Parallel Composition.* Given k algorithms working on disjoint subsets, each satisfying DP for $\{\varepsilon_1, \cdots, \varepsilon_k\}$, the result satisfies ε-differential privacy for $\varepsilon = \max\{\varepsilon_i\}$.

DPSGD. In the domain of privacy-preserving deep learning, *differentially private stochastic gradient descent* (DPSGD) [1] is the most widely adopted algorithm. It modifies the standard mini-batch gradient estimation in SGD by implementing a privatized version, wherein the gradient of each training instance is restricted to a maximum norm, known as the clipping norm. Subsequently, Gaussian noise proportional to the clipping norm is added to the summed clipped gradients. This addition effectively obscures the contribution of individual examples to the total gradient.

On a broader spectrum, the overall privacy budget of DPSGD is established by demonstrating the privacy budget of each iteration under specific (ε, δ) values, followed by the application of amplification by subsampling and composition throughout the iterations [11, 26, 27].

2.2 Sharp and Flat Minima and Sharpness-Aware Minimization

Flat Minima. In modern machine learning, the loss function $\ell : \mathbb{R}^d \to \mathbb{R}$ typically has many local and global minima, and an important question is which of these minima yield good predictive performance. Among the numerous properties studied, the "flatness" of a minimum has attracted particular attention [13, 20, 28, 32, 33, 36, 49, 52, 60, 68, 75, 77].

A common way to quantify "flatness" is via the Hessian matrix $\mathbf{H} := \nabla^2 \ell(\boldsymbol{w})$ or its spectral norm $\|\mathbf{H}\|_2$, equivalent to the largest eigenvalue $\lambda_{\max}$. This norm captures the local curvature of the loss landscape [16]. When $\nabla^2 \ell(\boldsymbol{w})$ is positive semi-definite, flatter regions of the loss landscape are associated with smaller Hessian norms, often indicating better generalization to unseen data.

Sharpness-Aware Minimization. Foret et al. [29] proposed *Sharpness-Aware Minimization* (SAM), a state-of-the-art optimization method in many domains. SAM aims to minimize the worst-case perturbation within a radius ρ around the current parameters:

$$\min_{\boldsymbol{w}} \max_{\|\boldsymbol{\delta}^*\| \leq \rho} \ell(\boldsymbol{w} + \boldsymbol{\delta}^*). \tag{1}$$

Because identifying the optimal direction $\boldsymbol{\delta}^*$ is challenging, SAM approximates $\boldsymbol{\delta}$ via a first-order Taylor expansion and updates $\boldsymbol{w}$ in two steps:

$$\boldsymbol{w}_t^p = \boldsymbol{w}_t + \rho \frac{\nabla \ell(\boldsymbol{w}_t)}{\|\nabla \ell(\boldsymbol{w}_t)\|} \tag{2}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla \ell(\boldsymbol{w}_t^p), \tag{3}$$

where (2) is the ascent step to perturb $\boldsymbol{w}_t$, and (3) is the descent step to move toward $\nabla \ell(\boldsymbol{w}_t^p)$. SAM consistently outperforms standard stochastic gradient descent (SGD) by exploring flatter regions that often lead to better generalization.

Sharpness-Aware Minimization in DPML. Although SAM effectively finds flat minima, its two-step optimization can incur additional privacy costs, making it less suitable for differentially private machine learning (DPML). To address this issue, Park et al. [56] proposed *Differentially Private Sharpness-Aware Training* (DPSAT), which integrates SAM into DPML without extra privacy overhead. Specifically, DPSAT reuses the noisy gradient from the previous step $(t-1)$ for the current ascent step (t). Let $\boldsymbol{g}_{t-1}^p$ denote the privatized gradient obtained via Gaussian noise addition at step $(t-1)$. Then, the ascent step is given by:

$$\boldsymbol{w}_t^p = \boldsymbol{w}_t + \rho \frac{\boldsymbol{g}_{t-1}^p}{\|\boldsymbol{g}_{t-1}^p\|}. \tag{4}$$

They prove that DPSAT satisfies (ε, δ)-DP under the same privacy budget as DPSGD.

2.3 Batch Normalization and Network Divergence

Batch Normalization. Normalizing inputs to zero mean and unit variance has long been recognized as beneficial for neural network training [42]. Batch normalization extends this concept to deeper networks, normalizing intermediate-layer activations [34], which significantly stabilizes training. However, in DPML, batch normalization makes each sample's normalized value dependent on other samples in the batch, complicating per-sample privacy guarantees. As a result, group normalization [73] is commonly used in DPML to avoid these issues [8, 10, 19, 67, 71].

Network Divergence. Deep networks without batch normalization often exhibit divergent behavior, with loss and activations growing uncontrollably as network depth increases [6]. This "exploding" effect can be mitigated by reducing the learning rate, but overly small learning rates slow convergence and may lead to poorer local minima. Batch normalization solves this by repeatedly centering and scaling activations, maintaining zero mean and unit variance across layers [6]. This mechanism effectively curbs activation explosion, facilitating deeper and more stable training.

3 Motivation and Key Observations

3.1 Key Observations

Our key observation is that, when training with SAM, the flatness of the model converges during the early epochs, much earlier than the convergence of the training loss. If the training is not stopped early, this optimization leads to a divergence in flatness, which often results in worse flatness later on. This phenomenon becomes more pronounced as the DP budget decreases (i.e., as DP noise increases).

Specifically, as shown in Figure 2, we use DPSAT [56], which incorporates SAM into DPSGD as discussed in an earlier section, to train a DP model using the DPNASNet-MNIST architecture [15] on the MNIST dataset [41]. We record the changes in flatness, measured by the l_2 norm of the Hessian matrix $||\mathbf{H}||_2$ (where a larger value indicates a sharper loss landscape and a smaller value indicates a flatter one), and the training loss over the epochs. The left, middle, and right panels represent the model trained with privacy budgets ε ranging from large (∞) to medium ($\varepsilon = 8$) to small ($\varepsilon = 1$). The red lines depict the changes in sharpness/flatness, while the blue lines represent the changes in training loss. In all three panels, the red line converges earlier (reaches a minimum and then diverges). As the privacy budget decreases, we observe that the convergence rate of flatness increases and the extent of the divergence issue intensifies. These results provide strong evidence that DPSAT fails to fully realize the benefits of SAM in enhancing the performance of DPML.

3.2 Motivation and Challenges.

Prior work [56] shows that SAM-like optimization can enhance DPML by promoting a flatter loss landscape. However, their approach yields suboptimal results (Section 5.2). Our main challenge lies in identifying how to apply SAM effectively in DPML, grounded in theoretical foundations.

Our key observation reveals an issue within DPSAT: the flatness is not well-optimized in the later stages of training due to the divergence issue. A straightforward solution is to stop the training process as soon as the flatness converges, thereby eliminating the divergence issue. However, this strategy leads to insufficient convergence of the training loss. This is evidenced by the changes in training loss over the epochs shown in Figure 2, where the training loss is typically still far from convergence when the flatness converges.

Intuitively, to fully leverage the benefits of SAM in enhancing the training performance of DPML, both the flatness and training loss should converge to their respective minima simultaneously. The challenge, therefore, is to design an algorithm that achieves this dual convergence.

To achieve simultaneous convergence of flatness and training loss to their respective minima, it is crucial to address the divergence issue of flatness. If the divergence of flatness can be effectively resolved, the flatness could remain near its minimum level when the training loss converges. However, this is challenging because previous work has shown that optimizing flatness is often very unstable [37], and the divergence issue occurs frequently. One possible solution is batch normalization, which has shown a stable ability to overcome divergence issues [6]. However, as discussed earlier (Section 2.3), batch normalization relies on peer samples within a batch, disrupting sample independence and making it incompatible with DPML. As an alternative, many studies have used group normalization [73] in DPML [8, 10, 19, 67, 71]. However, group normalization also shows limited ability in overcoming the flatness divergence issue (Section 5.2).

3.3 Hypotheses

To address the identified divergence challenge, it is essential to better understand the core factors behind it. Our intuition is that the divergence issue of flatness primarily stems from the high instability of the flatness optimization process. Additionally, this divergence issue is further intensified by the interference of DP noise. Therefore, we propose two hypotheses in this section. *Hypothesis 1* targets the nature of the high instability of flatness optimization, while *Hypothesis 2* addresses the impact of DP noise on flatness optimization.

Hypothesis 1. Given that SAM-like algorithms have been identified as highly unstable [37], we consider using SAM with an early stop strategy. Instead of stopping the entire training process early, we propose to stop optimizing the flatness (i.e., using SAM-like algorithms) while continuing to optimize the training loss (i.e., using standard SGD). This approach may mitigate the divergence issue without hindering the convergence of the training loss.

> ***H_1*: Early Stopping SAM.** When flatness converges, switching from SAM to optimizing the training loss only may mitigate the flatness divergence issue while preserving the convergence of the training loss.

The intuition behind this hypothesis is that if we focus solely on optimizing the training loss, the changes in flatness tend to be much more stable than when we continue

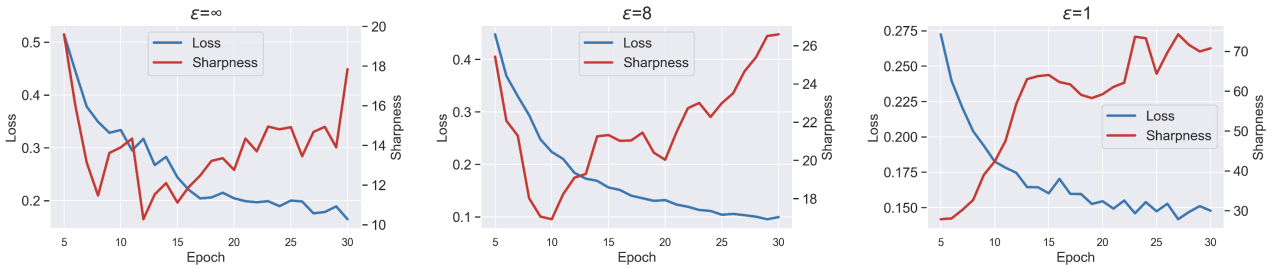

Figure 2: Convergence of CE Loss and Sharpness $\|\mathbf{H}\|_2$ of SAM by varying ε on MNIST.

optimizing flatness. This is because optimizing training loss and flatness are relatively independent processes [4, 53]. As a result, continuing to optimize the training loss does not significantly impact flatness, which would remain near a constant level since it is no longer being actively optimized. Consequently, the well-optimized flatness will be preserved with this early stopping strategy. Meanwhile, since the training loss is still being optimized, it will also sufficiently converge.

To verify this hypothesis, we conducted an experiment to explore the effects of early stopping on SAM in a non-DP setting, thereby masking the effects of DP noise. We used a DP-friendly architecture (typically without batch normalization) to maintain its adaptability to DP settings. Specifically, in this experiment, we configured the dataset to MNIST and the model architecture to DPNASNet-MNIST [15], the state-of-the-art architecture for DPML. The total number of epochs was set to 30. As shown in Figure 3, the red line represents the changes in sharpness (measured by $||\mathbf{H}||_2$) over epochs, while the blue line represents the changes in training loss (Cross-Entropy loss) over epochs. During the first 15 epochs, we used SAM to optimize flatness. After that, we applied two strategies: one continued optimizing using SAM (solid line) and the other switched to standard SGD (dashed line). We observed that continuing with SAM led to significant divergence in flatness, while switching to standard SGD effectively mitigated this issue, as indicated by its consistently lower curve compared to SAM, which aligns with our hypothesis. Additionally, the training loss converged better when switching to standard SGD in the later stages. This indicates that addressing the flatness divergence issue also helps improve the convergence of the training loss, reinforcing the original benefits of SAM [29].

Hypothesis 2. Hypothesis 1 suggests that to better leverage the benefits brought by SAM, the training process should be divided into two phases: an initial phase using SAM before flatness convergence, followed by a subsequent phase optimizing training loss only to mitigate the flatness divergence issue. However, Hypothesis 1 does not consider the impact of DP noise. As noted in Section 3.1, DP noise can cause earlier convergence of flatness and intensify the divergence issue. For instance, as shown in Figure 2, when the privacy budget is set to merely $\varepsilon = 1$, flatness can hardly converge. This makes solving the flatness divergence issue in DP settings more challenging. Therefore, given a targeted end-to-end privacy budget, it could be beneficial to allocate more budget to the initial phase, which is more sensitive to DP noise due to the instability of optimizing flatness compared to training loss, as identified in Figure 2.

> ***H_2*: Uneven Allocation of Privacy Budget.** The privacy budget allocated to the flatness optimization phase should be greater than that allocated to the training loss optimization phase.

Allocating more privacy budget to the flatness optimization phase can help mitigate the impact of DP noise on the convergence of flatness (as identified in Figure 2). By the time the training loss optimization phase begins, the flatness has already converged. As a result, although a relatively small budget is allocated to the training loss optimization phase due to this strategy, the impact of increased DP noise on the convergence of training loss can be limited. This is because previous work has shown that a flatter loss landscape is more robust to DP noise [56].

Moreover, increasing the privacy budget for the flatness optimization phase can benefit both flatness and training loss optimization during this phase. Since the flatness has not yet converged in this phase, the training loss is relatively more sensitive to DP noise compared to the subsequent training loss optimization phase [56]. It is important to note that training loss optimization is also crucial during this phase, as it serves as the starting point for the subsequent training loss optimization phase.

4 Sharpness-Aware Initialization

4.1 Methodology Overview

To address the previously identified flatness divergence challenge and fully leverage the benefits of SAM in enhancing the training performance of DPML, in this paper, we propose Sharpness-Aware Initialization (SAI).

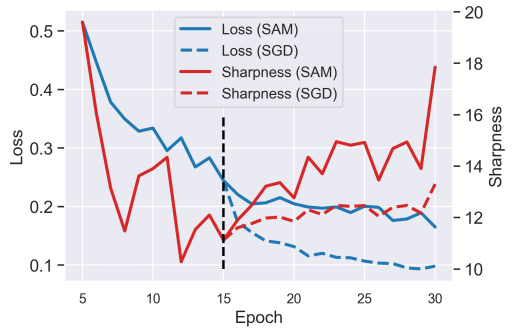

Figure 3: CE Loss and Sharpness $\|\mathbf{H}\|_2$ of SAM and SAM with early stopping on MNIST.

Figure 1 illustrates the overall workflow of the proposed SAI-DPSGD. To fully leverage the benefits of sharpness-aware training, SAI-DPSGD incorporates three key components, as described below.

• *Sharpness-Aware Initialization* (SAI). As an initial stage, SAI employs a SAM-like strategy to find a flatter loss landscape. Specifically, we reuse the perturbed gradient from the previous step to adjust the ascent direction of DP-SAM in the current step. This approach helps to avoid the additional privacy cost associated with SAM's two-step optimization, as suggested by DPSAT [56]. SAI serves as the foundation of the proposed SAI-DPSGD, as it identifies a flatter loss landscape, which is crucial for mitigating the negative impacts of clipping and noise addition.

• *Standard DPSGD.* Following the SAI stage, we further optimize the training loss using standard DPSGD. This stage commences once the flatness has converged. The transition can be managed automatically by monitoring the convergence of flatness or manually by setting it as a hyperparameter. Notably, in the automatic case, the switch decision relies solely on the spectral norm of the Hessian, which is computed from DP-protected outputs. Therefore, it doesn't access raw sensitive data and incurs no additional privacy cost. In this phase, since the flatness is no longer being optimized, issues related to divergence in flatness are resolved. Meanwhile, this stage allows standard DPSGD to achieve sufficient convergence on the training loss.

• *Budget Allocation.* The final crucial component is the strategic allocation of the privacy budget across different training phases. We propose allocating a relatively small budget to the standard DPSGD training phase, as the flatness has already converged by the time this phase begins, making it more tolerant to DP noise. The budget savings from this allocation can reduce the DP noise added during the SAI phase, thereby protecting both the optimization of flatness and the training loss. The specific budget allocation serves as a hyperparameter, and its optimal value depends on the dataset and model architecture used.

Remark. It is important to highlight a key distinction between our approach and DPSAT [56]. Specifically, DPSAT applies SAM throughout the entire training process, which can potentially conflict with other DP enhancements, as we have observed that SAM within DP is highly unstable (Figure 2). In contrast, our approach confines SAM to an initialization phase, isolating it from the impact of other enhancements. Consequently, the remaining DPSGD phase remains highly compatible with existing DP enhancements, as these are primarily designed for DPSGD. This separation serves as a significant advantage of the proposed two-phase training framework.

4.2 Design Details

Given a targeted end-to-end privacy budget (ε, δ), we first need to allocate a suitable $(\varepsilon_1, \delta_1)$ for the Sharpness-Aware Initialization (SAI) phase. The optimal value for this allocation can be identified through hyperparameter search. Then, the privacy budget for the standard DPSGD phase $(\varepsilon_2, \delta_2)$ is determined by the remaining budget, calculated from the given (ε, δ) and $(\varepsilon_1, \delta_1)$.

Given a total number of iterations T, we allocate T_1 iterations for the SAI phase. The value of T_1 can be determined through automatic early stopping, such as terminating the SAI phase if the flatness does not decrease over several consecutive iterations, or manually stopping based on a predefined number of iterations, as suggested by widely-used early stopping strategies [58]. The remaining number of iterations, T_2, for the standard DPSGD phase is calculated as $T_2 = T - T_1$.

Later in Section 4.3, we provide theoretical guidance on how to determine T_1 and T_2 for optimal performance.

During the SAI phase, we integrate DPSGD with SAM following the guidance of DPSAT [56]. Specifically, for each iteration $t \in [1, 2, \ldots, T_1]$, we select a batch of data I_t and then calculate the gradient as follows:

$$\boldsymbol{\delta}_t = \rho \boldsymbol{g}_{t-1}^p / (\|\boldsymbol{g}_{t-1}^p\| + \tau) \tag{5}$$

where ρ is the radius of the worst-case perturbation, $\boldsymbol{g}_{t-1}^p$ is the previous perturbed gradient, and τ is a small constant used to prevent division by zero.

Then, similar to DPSGD, we apply gradient clipping and add DP noise to this gradient, obtaining the DP-perturbed gradient:

$$\boldsymbol{g}_t^p = \frac{1}{|I_t|} \sum_{i \in I_t} \text{clip}(\nabla \ell_i(\boldsymbol{w}_t + \boldsymbol{\delta}_t), C_1) + \mathcal{N}(\mathbf{0}, C_1^2 \sigma_1^2 \mathbf{I}) \tag{6}$$

where $\nabla \ell_i(\boldsymbol{w}_t + \boldsymbol{\delta}_t)$ is the gradient of the loss function ℓ_i for the i-th data point at parameter $\boldsymbol{w}_t + \boldsymbol{\delta}_t$, C_1 is the clipping threshold for SAI, and $\mathcal{N}(\mathbf{0}, C_1^2 \sigma_1^2 \mathbf{I})$ is the Gaussian noise added for differential privacy, with σ_1 controlling the noise scale. Note that σ_1 is determined by $(\varepsilon_1, \delta_1)$.

Algorithm 1 SAI-DPSGD

Require: Initial parameter $\boldsymbol{w}_0$, learning rate η_1 for SAI, learning rate η_2 for DPSGD, clipping threshold C_1 for SAI, clipping threshold C_2 for DPSGD, radius ρ, constant τ to prevent zero division, privacy budget (ε, δ), and number of iterations T.

Initialize: $\boldsymbol{g}_0^p = \mathbf{0}$
Allocate T into T_1 and T_2
Allocate privacy budget (ε, δ) into $(\varepsilon_1, \delta_1)$ and $(\varepsilon_2, \delta_2)$
Determine noise variances σ_1^2 and σ_2^2 using the PRV accountant [31] on $(\varepsilon_1, \delta_1)$ and $(\varepsilon_2, \delta_2)$
# Sharpness-Aware Initialization
for $t = 1, 2, \ldots, T_1$ **do**
  Construct a mini-batch I_t
  $\boldsymbol{\delta}_t = \rho \boldsymbol{g}_{t-1}^p / (\|\boldsymbol{g}_{t-1}^p\| + \tau)$
  $\boldsymbol{g}_t^p = \frac{1}{|I_t|} \sum_{i \in I_t} \text{clip}(\nabla \ell_i(\boldsymbol{w}_t + \boldsymbol{\delta}_t), C_1) + \mathcal{N}(\mathbf{0}, C_1^2 \sigma_1^2 \mathbf{I})$
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_1 \boldsymbol{g}_t^p$
end for
# DPSGD
for $t = 1, 2, \ldots, T_2$ **do**
  Construct a mini-batch I_t
  $\boldsymbol{g}_t = \frac{1}{|I_t|} \sum_{i \in I_t} \text{clip}(\nabla \ell_i(\boldsymbol{w}_t), C_2) + \mathcal{N}(\mathbf{0}, C_2^2 \sigma_2^2 \mathbf{I})$
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_2 \boldsymbol{g}_t$
end for

Note that, in our implementation, we employ the *private random variable* (PRV) accountant [31] to determine noise variances. Alternatives such as the Rényi Differential Privacy (RDP) accountant [48] or other advanced accountants can also be used.

Finally, we update the model parameters as follows:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_1 \boldsymbol{g}_t^p \tag{7}$$

where η_1 is the learning rate for the SAI phase, and $\boldsymbol{g}_t^p$ is the DP-perturbed gradient calculated in the previous step.

Following the SAI phase, we continue to optimize the model using standard DPSGD for an additional T_2 iterations. During this phase, the focus shifts from optimizing flatness to ensuring the convergence of the training loss, leveraging the improved flatness achieved during the SAI phase.

The complete algorithm for the proposed SAI-DPSGD framework is detailed in Algorithm 1.

4.3 Theoretical Understanding

In this section, we provide a theoretical perspective that clarifies why *Sharpness-Aware Initialization (SAI)* followed by a standard gradient-based method yields a favorable convergence profile. We begin by outlining the necessary definitions and assumptions for our analysis (followed by [2]), then present a two-phase convergence result indicating that a sharper-focus phase can locate a flat local minimum in relatively few iterations, after which a standard optimization phase can efficiently reduce the loss further. Finally, we discuss how these insights motivate our design of SAI and guide the choice of hyperparameters T_1 and T_2.

$$\ell(\boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{n} \ell_i(\boldsymbol{w}) := \frac{1}{n} \sum_{i=1}^{n} \ell\big(p_i(\boldsymbol{w}), y_i\big), \tag{8}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ denotes the model parameters, $p_i(\boldsymbol{w})$ is the model's prediction for the i-th data point, and y_i is its ground-truth label. We define

$$W^* = \{\boldsymbol{w} \in \mathbb{R}^d : p_i(\boldsymbol{w}) = y_i \text{ for all } i = 1, \ldots, n\},$$

the set of global minima, and assume $\nabla p_i(\boldsymbol{w}) \neq 0$ for any $\boldsymbol{w} \in W^*$.

Gradient flow. A fundamental notion in our analysis is the gradient flow. For a continuous-time process $\boldsymbol{w}(t)$ that follows

$$\dot{\boldsymbol{w}}(t) = -\nabla \ell\big(\boldsymbol{w}(t)\big), \quad \boldsymbol{w}(0) = \boldsymbol{w},$$

we define its limiting point:

Definition 2 (Gradient flow). *For any initial point $\boldsymbol{w}$, let $\Phi(\boldsymbol{w})$ be the limit of the gradient flow on ℓ starting at $\boldsymbol{w}$. Formally, if $\boldsymbol{w}(t)$ follows the above differential equation, then*

$$\Phi(\boldsymbol{w}) = \lim_{t \to \infty} \boldsymbol{w}(t).$$

Flat local minima. Next, we define a notion of $(\varepsilon, \varepsilon')$-flat local minima that captures both proximity to a global minimizer and a bounded measure of curvature (via the trace of the Hessian).

Definition 3 (Flat local minima). *Let $\boldsymbol{w}^* = \Phi(\boldsymbol{w})$ be the limiting point of the gradient flow from $\boldsymbol{w}$. Then $\boldsymbol{w}$ is an $(\varepsilon, \varepsilon')$-flat local minimum if*

$$\|\boldsymbol{w} - \boldsymbol{w}^*\| \leq \varepsilon \quad \text{and} \quad \big\| \big[\nabla \big(\text{tr} \circ \nabla^2 \ell(\Phi)\big)\big](\boldsymbol{w}^*) \big\| \leq d\,\varepsilon'.$$

Remark. In Definition 3, ε indicates how close $\boldsymbol{w}$ is to a minimizer $\boldsymbol{w}^*$ of ℓ, and ε' measures how flat the local region around $\boldsymbol{w}^*$ is, in terms of the Hessian trace (tr).

Two-Phase Convergence. We now describe a two-phase approach: first, run a *sharpness-aware* method (analogous to SAI) for T_1 iterations to reach an $(\varepsilon^{1.5}, \sqrt{\varepsilon})$-flat region, then switch to a standard gradient-based method for T_2 iterations to refine the loss while retaining approximate flatness. Concretely:

- **Phase 1 (Sharpness-aware method)**: After at most

$$T_1 = O\big(d^{-1} \varepsilon^{-2} \max\{1, \tfrac{1}{\delta^3 \varepsilon^4}\}\big), \tag{9}$$

  the model reaches an $\big(O(\varepsilon^{1.5}), \sqrt{\varepsilon}\big)$-flat local minimum with high probability, provided $\boldsymbol{w}_0$ is suitably close to W^*.

- **Phase 2 (Gradient descent)**: From this $\big(O(\varepsilon^{1.5}), \sqrt{\varepsilon}\big)$-flat point, running a standard gradient-based procedure with step size $\eta = O(\varepsilon)$ achieves an $\big(\varepsilon, \sqrt{\varepsilon}\big)$-flat local minimum within

$$T_2 = O\big(\varepsilon^{-1} \log(1/\varepsilon)\big) \tag{10}$$

  iterations.

Focusing on sharpness early (Phase 1) can rapidly secure a flatter region, while continuing standard updates afterward (Phase 2) then further reduces the loss.

Trade-Off in Iteration Allocation. If we denote the total iteration budget by $T_1 + T_2$, overextending the sharpness-aware phase gives diminishing returns, since the additional gains require increasingly many iterations. Conversely, insufficient sharpness-aware optimization leaves the model in a suboptimal, sharper region that standard updates alone may struggle to improve. A balanced split between the two phases mitigates these issues.

Theorem 1 (Informal)**.** *Follow the same assumptions and conditions of Theorem 2 from [2], let the target accuracy* $\varepsilon > 0$ *be chosen sufficiently small, and* $\delta \in (0,1)$*. Suppose* $\boldsymbol{w}_0$ *is suitably close to* W^**, and the model reaches an* $\left(O(\varepsilon^{1.5}), \sqrt{\varepsilon}\right)$*-flat minimum after* T_1 *(in Equation 9) iterations of sharpness-aware optimization. Then gradient descent with* $\eta = O(\varepsilon)$ *finds a* $\left(\varepsilon, \sqrt{\varepsilon}\right)$*-flat minimum after at most* T_2 *(in Equation 10) more iterations. Let* ν *defined as* $\min\left\{d, \varepsilon^{-1/3}\right\}$*, d is the dimension of the domain. Then the ratio of* $\frac{T_1}{T_2}$ *is suppose to bounded by:*

$$\frac{1}{d\,\varepsilon\log(1/\varepsilon)} \;\leq\; \frac{T_1}{T_2} \;\leq\; \frac{\delta}{\varepsilon^3\nu^3}. \tag{11}$$

Remark on the theorem. According to [23], due to over-parameterization, the weight updates are small, and all weight vectors remain within a small neighborhood of their initialization. Therefore, under suitable conditions (and with high probability), $\boldsymbol{w}_0$ is close to W^*. In practice, pre-training on public data can help steer the parameters toward favorable regions. For simplicity, we provide the theorem statement here in informal form. The formal theorem and its proof can be found in Appendix B. This result describes how to allocate training iterations between the SAI-like phase and the standard optimization phase to achieve both low training loss and near-flat Hessian measurements.

Connection to SAI. These observations motivate *Sharpness-Aware Initialization*, where T_1 is the number of iterations for seeking a flatter solution, and T_2 is the subsequent refinement stage via standard (DP) optimization. The bound above suggests that once the model parameters reach a sufficiently flat region, additional focus on sharpness yields diminishing benefits; switching to a simpler training-loss-driven method can be more effective. On the other hand, neglecting the initial push toward flatness deprives the subsequent stage of its robust starting point. Hence, SAI's two-phase structure emerges as a natural solution to achieve both flatness and strong convergence, all under a limited iteration (and privacy) budget.

5 Evaluation

5.1 Experimental Setup

DPML Algorithms. We conduct experiments on three DPML algorithms: DPSGD [1], DPSAT [56], and the proposed SAI-DPSAT, using two different base optimizers: SGD and Adam [38]. We use the implementations provided by the authors and the optimal hyperparameters identified in their papers for fair comparison [56]. All algorithms are implemented using PyTorch [57] and Opacus [78], and the experiments are run on 4 NVIDIA Tesla A100 GPUs.

Datasets. We adopt four datasets for our experiments, including (1) MNIST [41] that contains 60,000 handwritten digits from 0 to 9, with 50,000 training images and 10,000 testing images; (2) FashionMNIST [74] that consists of 70,000 grayscale images of 10 fashion categories, with 60,000 training images and 10,000 testing images; (3) CIFAR-10 [39] that contains 60,000 color images in 10 classes, with 50,000 training images and 10,000 testing images; and (4) SVHN [51] that represents digits from house numbers in Google Street View, with 73,257 training images and 26,032 testing images. These datasets are widely used in evaluating privacy-preserving machine learning methods [1, 54, 56, 71, 80, 81].

Model Architectures. We focus on four model architectures, including GNResNet-10 (Group Norm ResNet-10) and DPNASNet-MNIST [15] for MNIST and FashionMNIST; and CNN-Tanh with SELU [55] and DPNASNet-CIFAR [15] for CIFAR-10 and SVHN. Particularly, DPNASNet architectures are state-of-the-art architectures for training with DPSGD.

Additionally, we evaluated our approach on Vision Transformers (ViT) [21], including CrossViT_tiny_240 [14], which is designed for efficient processing with fewer parameters; CrossViT_small_240 [14], which offers a balance between model size and performance; and CrossViT_18_240 [14], which is a larger model aimed at maximizing accuracy, for CIFAR-10 and CIFAR-100. The evaluation on these architectures explores the use of fine-tuning and transfer learning using PyTorch Image Models [72], which has been identified as a better approach when training DP-ViTs [8].

DPML Configurations. We utilize different privacy budgets: $\varepsilon = \{0.5, 1, 2, 3, 5, 8, \infty\}$. The clipping threshold C for DPSGD, DPSAT, and the DPSGD in SAI-DPSGD (C_2) is fixed at 0.1. We adopt cross-entropy as the loss function and utilize SGD as the base optimizer with a momentum of 0.9. The batch size is set to 2048 for all the algorithms. The learning rate η for DPSGD, DPSAT, and SAI (η_1) is fixed at 2.0, while the learning rate for the DPSGD in SAI-DPSGD (η_2) is fixed at 0.1. The total number of epochs is set to 40 for MNIST and FashionMNIST, and to 30 for CIFAR-10 and SVHN. To ensure a fair comparison across all baselines, we conduct a grid search to determine the hyperparameters specific to SAI. Specifically, we search over the radius $\rho \in \{0.03, 0.05, 0.1, 0.3\}$, the number of epochs for SAI

Table 1: Test accuracy of DPSGD, DPSAT, and SAI-DPSGD on the MNIST, FashionMNIST, CIFAR-10, and SVHN datasets. We also report the Error Reduction Rate (ERR) when trained with SAI-DPSGD, in comparison to DPSAT.

Datasets	Model	Privacy budget	Optimizers			ERR (%)
			DPSGD	DPSAT	SAI-DPSAT	
MNIST	GNResNet-10	$\varepsilon = 1, \delta = 10^{-8}$	95.02±0.05	95.66±0.22	**96.59±0.22**	21.43%
		$\varepsilon = 1, \delta = 10^{-5}$	95.15±0.17	96.00±0.21	**97.29±0.05**	32.25%
		$\varepsilon = 2, \delta = 10^{-5}$	96.68±0.27	97.35±0.14	**97.55±0.14**	7.55%
		$\varepsilon = 3, \delta = 10^{-5}$	97.30±0.14	97.83±0.10	**98.12±0.14**	13.36%
	DPNAS-MNIST	$\varepsilon = 1, \delta = 10^{-8}$	97.18±0.04	97.38±0.08	**97.90±0.13**	19.85%
		$\varepsilon = 1, \delta = 10^{-5}$	97.77±0.13	97.96±0.08	**98.39±0.02**	21.08%
		$\varepsilon = 2, \delta = 10^{-5}$	98.60±0.06	98.71±0.09	**98.78±0.13**	5.81%
		$\varepsilon = 3, \delta = 10^{-5}$	98.70±0.12	98.93±0.02	**98.96±0.06**	2.80%
FashionMNIST	GNResNet-10	$\varepsilon = 1, \delta = 10^{-8}$	79.88±0.14	81.09±0.07	**82.50±0.16**	7.46%
		$\varepsilon = 1, \delta = 10^{-5}$	80.57±0.25	81.33±0.45	**82.81±0.27**	7.95%
		$\varepsilon = 2, \delta = 10^{-5}$	82.71±0.35	84.53±0.41	**85.06±0.29**	3.39%
		$\varepsilon = 3, \delta = 10^{-5}$	84.55±0.17	85.91±0.22	**86.38±0.10**	3.34%
	DPNAS-MNIST	$\varepsilon = 1, \delta = 10^{-8}$	82.72±0.19	83.33±0.33	**85.00±0.43**	9.99%
		$\varepsilon = 1, \delta = 10^{-5}$	84.62±0.19	85.92±0.35	**86.36±0.16**	3.12%
		$\varepsilon = 2, \delta = 10^{-5}$	86.99±0.57	87.75±0.24	**88.40±0.13**	5.31%
		$\varepsilon = 3, \delta = 10^{-5}$	87.97±0.17	88.60±0.04	**89.05±0.03**	3.95%
CIFAR-10	CNN-Tanh with SELU	$\varepsilon = 1, \delta = 10^{-8}$	41.05±0.50	41.30±0.38	**49.52±0.21**	14.00%
		$\varepsilon = 1, \delta = 10^{-5}$	45.24±0.42	45.78±0.48	**51.79±0.66**	11.08%
		$\varepsilon = 2, \delta = 10^{-5}$	56.90±0.33	58.35±0.55	**60.44±0.40**	5.02%
		$\varepsilon = 3, \delta = 10^{-5}$	61.84±0.48	63.51±0.40	**64.30±0.12**	2.16%
	DPNAS-CIFAR10	$\varepsilon = 1, \delta = 10^{-8}$	54.85±0.30	54.34±0.27	**60.22±0.76**	12.87%
		$\varepsilon = 1, \delta = 10^{-5}$	59.42±0.38	60.13±0.34	**62.76±0.40**	6.60%
		$\varepsilon = 2, \delta = 10^{-5}$	66.30±0.27	67.23±0.12	**67.91±0.17**	2.08%
		$\varepsilon = 3, \delta = 10^{-5}$	68.43±0.43	69.86±0.49	**70.19±0.14**	1.11%
SVHN	DPNAS-CIFAR10	$\varepsilon = 1, \delta = 10^{-8}$	78.82±0.29	79.01±0.32	**84.34±0.23**	25.37%
		$\varepsilon = 1, \delta = 10^{-5}$	82.25±0.15	83.09±0.54	**86.27±0.12**	18.81%
		$\varepsilon = 2, \delta = 10^{-5}$	86.85±0.33	87.68±0.13	**88.00±0.06**	2.60%
		$\varepsilon = 3, \delta = 10^{-5}$	88.18±0.23	88.74±0.18	**88.89±0.16**	1.33%

$e_1 \in \{10, 15, 20\}$, the portion of the total budget allocated to SAI $p \in \{0.8, 0.9\}$, and the clipping threshold for SAI $C_1 \in \{0.1, 0.2\}$. Note that the optimal radius, epoch count, and budget allocation may vary depending on the total privacy budget used in DP training. Please refer to Appendix A for more details of the experimental settings.

In practice, public data can be leveraged for hyperparameter selection without incurring additional privacy costs. As shown in Section 5.8, SAI-DPSGD remains effective when its hyperparameters are selected using public datasets.

Metrics. We use accuracy (Acc) to evaluate the models' utility. Additionally, we analyze the Hessian matrix of the loss function, $\mathbf{H} = \mathbf{H}_{\boldsymbol{w}} := \nabla^2 \ell(\boldsymbol{w})$, and its *sharpness*, which is defined as its spectral norm $\|\mathbf{H}\|_2$ (or the largest eigenvalue λ_{max}). A lower $\|\mathbf{H}\|_2$ indicates a flatter loss landscape. We also report the Error Reduction Rate (ERR) when trained with SAI-DPSGD, in comparison to DPSAT:

$$\text{ERR} = \frac{\text{ACC}_{M_{SAI\text{-}DPSGD}} - \text{ACC}_{M_{DPSAT}}}{1 - \text{ACC}_{M_{DPSAT}}} \tag{12}$$

where $M_{SAI\text{-}DPSGD}$ and M_{DPSAT} represent private models trained by SAI-DPSGD and DPSAT. The ERR value indicates the proportion of accuracy recovered by SAI.

5.2 Effectiveness of SAI

We conducted a performance comparison of DPSGD, DPSAT, and SAI-DPSGD as presented in Table 1. Overall, the proposed SAI-DPSGD demonstrates superior performance compared to DPSGD and DPSAT across all scenarios.

Take the most challenging scenario with a privacy budget of $(\varepsilon = 1, \delta = 10^{-8})$ as an example. When using the CIFAR-10 dataset and the CNN-Tanh architecture, SAI-DPSGD boosts accuracy to 49.52%. In contrast, DPSAT achieves only 41.30%, resulting in an accuracy improvement of 8.22% and an ERR of 14%. Similarly, with the SVHN dataset and the DPNASNet-CIFAR architecture, SAI-DPSGD increases accuracy from 79.01% to 84.34%, marking an improvement of 5.33% and an ERR of 25.37%. Moreover, the performance improvements are particularly pronounced in larger models,

Figure 4: Difference of accuracy for SAI-DPSGD and DPSAT w.r.t DPSGD. $\varepsilon = \infty$ indicates non-DP settings.

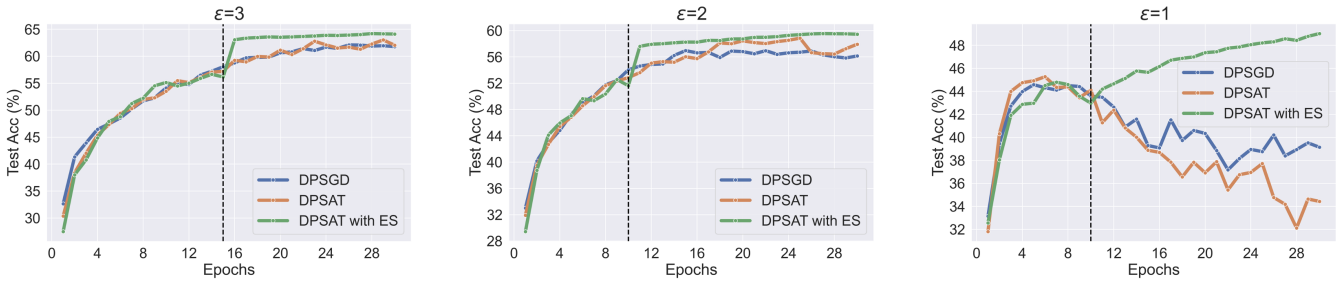

Figure 5: The test accuracy of DPSGD, DPSAT, and DPSAT with Early Stopping (ES) by varying ε on CIFAR-10.

such as GNResNet-10. For the widely-used MNIST benchmark, SAI-DPSGD achieves an average ERR of 18.65%. Notably, when the privacy budget is set to $(\varepsilon = 1, \delta = 10^{-5})$, SAI-DPSGD achieves an ERR of 32.25%. Large models with complex architectures often face challenges in generalizing well in DP settings due to their susceptibility to perturbations [67]. The advanced flatness provided by SAI-DPSGD proves notably advantageous in these cases. Furthermore, it is important to note that GNResNet-10 leverages group normalization [73], a widely used replacement for batch normalization in DPML [8, 10, 19, 67, 71]. However, group normalization cannot effectively address the divergence issue of flatness, as evidenced by the significant gap between DPSAT and SAI-DPSGD despite the use of group normalization. This demonstrates the irreplaceable value of the proposed SAI-DPSGD strategy.

To visualize the differences between optimization methods, we plot the accuracy difference with respect to DPSGD in Figure 4. We maintain $\delta = 10^{-5}$ and test various privacy budgets, $\varepsilon \in \{0.5, 1, 2, 3, 5, 8\}$, including the non-DP setting ($\varepsilon = \infty$). Consistent with previous results, SAI-DPSGD shows the best performance among the methods. Furthermore, as the privacy budget ε increases, gradually approaching the non-DP settings, the gap between the proposed SAI-DPSGD and DPSAT diminishes. This indicates that SAI-DPSGD primarily leverages the positive effects of flatness in DP models rather than in non-DP settings. Additionally, we observe that, in general, the lower the privacy budget, the larger the gap between SAI-DPSGD and DPSGD, even when the privacy budget is reduced to extremely low values, such as $\varepsilon = 0.5$. However, this trend is not observed with DPSAT. The peak gap between DPSAT and DPSGD occurs around $\varepsilon = 2.0$, and as the privacy budget is further reduced, the gap between DPSAT and DPSGD diminishes. This can be attributed to the divergence issue in flatness, which leads to a sharper loss landscape in the later stages of training. Recall that a sharper loss landscape has been proven to increase the detrimental effects of noise addition [56]. As a result, the lower the privacy budget ε (i.e., the larger the noise), the more pronounced these detrimental effects become, ultimately resulting in worse performance. In contrast, SAI-DPSGD avoids this issue by early stopping the optimization of flatness, thereby achieving remarkably good results even in cases with extremely low privacy budgets.

5.3 DPSAT Requires Early Stop

Setup. In this experiment, we investigate the impact of early stopping when applying worst-case perturbations in DPSAT. Specifically, we evaluate "DPSAT with early stopping", a simplified version that stops the sharpness-aware phase early while using an even privacy budget throughout. In contrast, SAI-DPSGD reallocates the privacy budget to allocate more to the sharpness-aware phase.

We maintain $\delta = 10^{-5}$ and investigate three privacy budgets $\varepsilon \in \{1, 2, 3\}$. The early stopping epoch is set to 15 for $\varepsilon = 3$, and 10 for both $\varepsilon = 2$ and $\varepsilon = 1$. The total number of epochs is fixed at 30. The dataset used is CIFAR-10, and the model architecture is CNN-Tanh with SELU.

Observations. The results, shown in Figure 5, reveal the influence of early stopping on the effectiveness of DPSAT. Overall, early stopping enhances test accuracy across all configura-

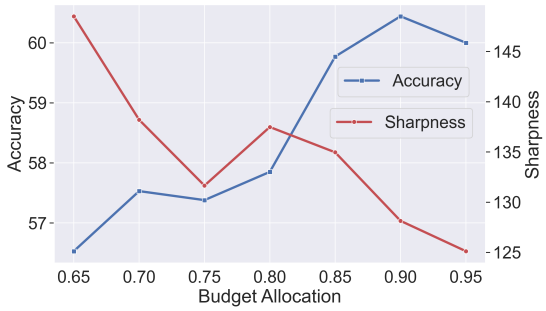

Figure 6: The impact of Budget Allocation.

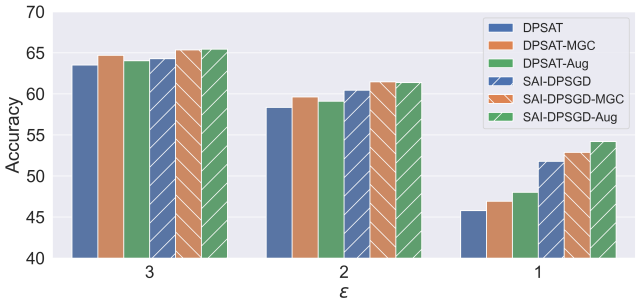

Figure 7: Compatibility of SAI with MGC and Self-Aug.

tions, demonstrating that SAI-DPSGD leverages the positive effects of early stopping effectively. Consistent with the results from the previous section, we observe that the lower the privacy budget, the larger the gap between DPSAT and DPSAT with early stopping. Notably, when the privacy budget is set to $\varepsilon = 1$, although DPSAT achieves better optimal test accuracy than DPSGD at an early stage, its performance is worse than DPSGD if we consider the final epoch rather than the best epoch. This could be due to the severe flatness divergence in the later stage of training when the privacy budget is extremely low. This also explains why DPSAT does not achieve good results with extremely low privacy budgets, as observed in the previous section.

5.4 Impact of Budget Allocation

Setup. Recall that in SAI-DPSGD, the allocation of the privacy budget to SAI is crucial for fully leveraging the optimized flatness it provides. In this section, we investigate the impact of the budget portion allocated to SAI on the performance of SAI-DPSGD. We explore seven allocation strategies, with portions allocated to SAI set at {65%, 70%, 75%, 80%, 85%, 90%, 95%}. The dataset used is CIFAR-10, and the model architecture is CNN-Tanh with SELU. The total number of epochs is fixed at 30, with early stopping set at epoch 10. The total budget is set to $(\varepsilon = 2, \delta = 10^{-5})$.

Observations. Figure 6 illustrates the relationship among the portion of the privacy budget allocated to SAI, the lowest sharpness (highest flatness) achieved during the entire training process (measured by $\|\mathbf{H}\|_2$), and the performance of SAI-DPSGD. We observe that, in general, the higher the portion allocated to SAI, the greater the peak flatness achieved during the SAI-DPSGD training process. This occurs because increased DP noise has a more significant influence on the convergence of flatness, a widely observed phenomenon in DPML [9]. However, SAI-DPSGD reaches its peak performance when 90% of the privacy budget is allocated to SAI. This could be because, when the budget allocated to DPSGD is extremely low, the DP noise becomes too high, and a much flatter loss landscape no longer brings additional benefits to DPSGD's convergence.

Table 2: The impact of dataset and model architecture.

Model	Datasets	Optimizers		ERR (%)
		DPSAT	SAI-DPSAT	
CrossViT_tiny_240 (#Params: 6.7M)	CIFAR-10	88.76±0.28	**89.73±0.25**	8.63%
	CIFAR-100	59.75±0.40	**60.48±0.21**	1.81%
CrossViT_small_240 (#Params: 26.3M)	CIFAR-10	94.15±0.28	**94.45±0.27**	5.13%
	CIFAR-100	71.38±0.23	**71.77±0.19**	1.36%
CrossViT_18_240 (#Params: 42.6M)	CIFAR-10	95.31±0.13	**95.52±0.15**	4.48%
	CIFAR-100	74.52±0.19	**74.88±0.20**	1.41%

5.5 Compatibility with Other Enhancements

Setup. Recall that we have demonstrated the compatibility of the proposed SAI-DPSGD with DP-friendly model architectures [15] and activation functions [55] (Section 5.2), showing that we can further improve test accuracy based on their implementations. In this experiment, we examine the compatibility of SAI-DPSGD with advanced clipping strategies and self-augmentation techniques. Specifically, for clipping, we adopt mixed ghost clipping (MGC), a state-of-the-art enhancement method introduced in recent work [8]. For self-augmentation, we follow De et al. [19], who demonstrated that simple data augmentations—such as horizontal flips and random cropping—can improve DP training performance. We denote methods that incorporate these augmentations with the suffix "-Aug". In this experiment, we evaluate three privacy budgets ($\varepsilon \in \{1, 2, 3\}$). The dataset used is CIFAR-10, and the model architecture is CNN-Tanh with SELU. The total number of epochs is fixed at 30, with early stopping set at epoch 10.

Observations. As shown in Figure 7, both MGC and self-augmentation improve test accuracy compared to not using these enhancements. When combined with the proposed SAI, test accuracy improves further, demonstrating the strong compatibility of SAI-DPSGD with both techniques. Notably, at a low privacy budget of $\varepsilon = 1$, SAI alone yields a significantly greater improvement than either MGC or self-augmentation alone. At higher budgets ($\varepsilon = 2$ and $\varepsilon = 3$), the performance gains from all three methods become comparable.

Table 3: Test accuracy of DPSGD, DPAdam, DPSAT, DPSAT (Adam), SAI-DPSGD and SAI-DPAdam on the MNIST, Fashion-MNIST, and CIFAR-10 datasets.

Datasets	Privacy budget ε ($\delta = 10^{-5}$)	Optimizers					
		DPSGD	DPAdam	DPSAT	DPSAT (Adam)	SAI-DPSGD	SAI-DPAdam
MNIST	$\varepsilon = 1$	97.77±0.13	97.94±0.15	97.96±0.08	98.29±0.20	**98.39±0.02**	**98.34±0.08**
	$\varepsilon = 2$	98.60±0.06	98.51±0.11	98.71±0.09	98.72±0.11	**98.78±0.13**	**98.80±0.08**
	$\varepsilon = 3$	98.70±0.12	98.69±0.03	98.93±0.02	98.83±0.09	**98.96±0.06**	**98.93±0.06**
FashionMNIST	$\varepsilon = 1$	84.62±0.19	84.92±0.14	85.92±0.35	85.51±0.30	**86.36±0.16**	**85.70±0.12**
	$\varepsilon = 2$	86.99±0.57	87.09±0.55	87.75±0.24	87.40±0.26	**88.40±0.13**	**87.67±0.31**
	$\varepsilon = 3$	87.97±0.17	87.94±0.22	88.60±0.04	88.25±0.27	**89.05±0.03**	**88.43±0.22**
CIFAR-10	$\varepsilon = 1$	59.42±0.38	62.01±0.59	60.13±0.34	62.56±0.34	**62.76±0.40**	**63.34±0.14**
	$\varepsilon = 2$	66.30±0.27	66.50±0.33	67.23±0.12	66.56±0.14	**67.91±0.17**	**68.12±0.11**
	$\varepsilon = 3$	68.43±0.43	68.75±0.20	69.86±0.49	69.23±0.33	**70.19±0.14**	**70.73±0.16**

5.6 Impact of Dataset and Model Architecture

Setup. In this experiment, we explore the adaptability of the proposed SAI-DPSGD on Vision Transformers (ViTs) [21] and more complex classification tasks such as CIFAR-100. We evaluate three model architectures: CrossViT_tiny_240, CrossViT_small_240, and CrossViT_18_240, using two datasets: CIFAR-10 and CIFAR-100. The evaluation in this section follows a fine-tuning (transfer learning) approach using the pre-trained models provided by PyTorch Image Models [72]. We use Adam as the base optimizer with a learning rate of 0.002. The models are trained for 5 epochs with a batch size of 1000 and a mini-batch size of 100. Early stopping is set at epoch 2. The total privacy budget is set to ($\varepsilon = 2$, $\delta = 10^{-5}$), with 90% allocated to SAI. The radius ρ is set to 0.002 for CIFAR-10 and 0.0005 for CIFAR-100.

Observations. We conducted a performance comparison of DPSAT and SAI-DPSGD, as presented in Table 2. The pre-trained ViT models, which have well-generalizing latent spaces, show much higher fine-tuning accuracy than other CNN models on CIFAR-10 as demonstrated in the previous section. Generally, SAI-DPSGD achieves the highest accuracy across all configurations, showcasing its adaptability. For instance, SAI-DPSGD boosts the accuracy of CrossViT_tiny_240 on CIFAR-10 with an ERR of 8.63% compared to DPSAT. It is important to note that the relatively small gap between DPSAT and SAI-DPSGD in this experiment could be due to the relatively short training period.

5.7 Impact of Base Optimizer

Setup. Recall that in our experiments, we utilize SGD as the default base optimizer for all the DPML algorithms: DPSGD, DPSAT, and SAI-DPSGD. In this experiment, we explore the impact of different base optimizers on these DPML algorithms. We investigate six DPML settings: DPSGD, DPAdam, DPSAT, DPSAT(Adam), SAI-DPSGD, and SAI-DPAdam. A recent study [66] shows that DP-Adam is essentially DP-SGD with momentum unless bias correction is applied. Therefore, we adopt DP-AdamBC (DPAdam with bias correction) [66] for the DPAdam, DPSAT(Adam), and SAI-DPAdam settings. In these settings, the learning rate is set to 0.002, while all other hyperparameters follow those used in the SGD-based configurations. In this experiment, we maintain $\delta = 10^{-5}$ and evaluate three privacy budgets ($\varepsilon \in \{1, 2, 3\}$) and three datasets: MNIST, FashionMNIST, and CIFAR-10. The model architecture used is DPNASNet-CIFAR for CIFAR-10 and DPNASNet-MNIST for MNIST and FashionMNIST.

Observations. The results, as illustrated in Table 3, demonstrate the influence of the base optimizer on different DPML algorithms. We observe that the difference in performance between SGD and Adam is marginal. In all cases, algorithms incorporating SAI consistently achieve the highest accuracy, regardless of the base optimizer used.

5.8 Impact of Hyper-parameter Selection

Setup. In previous sections, we used grid search to ensure a fair comparison across all baselines. Here, we evaluate the effectiveness of SAI-DPSGD when its hyperparameters are selected using public datasets. We set the total privacy budget to ($\varepsilon = 1$, $\delta = 10^{-5}$). For CIFAR-10 and SVHN, we use DPNASNet-CIFAR, and for MNIST and FashionMNIST, we use DPNASNet-MNIST.

Observations. The results in Table 4 highlight how the choice of dataset for hyperparameter selection affects the performance of different DPML algorithms. Each row corresponds to the dataset used for grid search, while each column shows the dataset on which the selected hyperparameters are applied. Diagonal entries represent the ideal case—hyperparameters selected and applied on the same dataset. Off-diagonal entries reflect settings where public datasets are used for hyperparameter tuning. We observe that across all settings, SAI-DPSGD consistently outperforms DPSGD and DPSAT. Even when public datasets lead to suboptimal hyperparameters,

Table 4: The impact of the dataset used for hyperparameter selection.

Public \ Private	MNIST		FashionMNIST		CIFAR-10		SVHN	
MNIST	DPSGD	97.77±0.13	DPSGD	84.62±0.19	DPSGD	58.51±0.54	DPSGD	81.30±0.57
	DPSAT	97.96±0.08	DPSAT	85.00±0.49	DPSAT	58.35±0.33	DPSAT	80.97±0.31
	SAI-DPSGD	**98.39±0.02**	SAI-DPSGD	**86.36±0.16**	SAI-DPSGD	**60.12±0.23**	SAI-DPSGD	**83.02±0.22**
FashionMNIST	DPSGD	97.77±0.13	DPSGD	84.62±0.19	DPSGD	58.51±0.54	DPSGD	81.30±0.57
	DPSAT	97.89±0.12	DPSAT	85.92±0.35	DPSAT	58.58±0.46	DPSAT	81.47±0.23
	SAI-DPSGD	**98.39±0.02**	SAI-DPSGD	**86.36±0.16**	SAI-DPSGD	**60.12±0.23**	SAI-DPSGD	**83.02±0.22**
CIFAR-10	DPSGD	97.76±0.09	DPSGD	84.32±0.29	DPSGD	59.42±0.38	DPSGD	82.25±0.15
	DPSAT	97.85±0.14	DPSAT	85.02±0.22	DPSAT	60.13±0.34	DPSAT	82.29±0.17
	SAI-DPSGD	**98.02±0.10**	SAI-DPSGD	**85.15±0.13**	SAI-DPSGD	**62.76±0.40**	SAI-DPSGD	**85.09±0.14**
SVHN	DPSGD	97.76±0.09	DPSGD	84.32±0.29	DPSGD	59.42±0.38	DPSGD	82.25±0.15
	DPSAT	97.89±0.10	DPSAT	85.09±0.15	DPSAT	59.88±0.23	DPSAT	83.09±0.54
	SAI-DPSGD	**98.08±0.19**	SAI-DPSGD	**85.19±0.15**	SAI-DPSGD	**60.84±0.18**	SAI-DPSGD	**86.27±0.12**

SAI-DPSGD maintains strong and stable performance, showing robustness to hyperparameter variation. We also observe strong transferability between MNIST and FashionMNIST, and between CIFAR-10 and SVHN—likely due to similarities between the dataset pairs. This indicates that, in practice, selecting hyperparameters using public data that closely resembles private data can further improve performance.

6 Discussion

Moreover, while our discussion and implementation focus on a single-party scenario, it is important to highlight that our framework is adaptable to both single-party and multi-party contexts. In a multi-party setting, multiple decentralized devices or servers with local data samples can collaboratively learn a model through federated learning [43]. DPSGD is particularly effective in such cases, as it adds noise to the gradients during training, ensuring that the contribution of any single data point is not easily identifiable, thereby protecting the privacy of individual data owners [1]. Consequently, the proposed enhancements to DPSGD are well-suited for multi-party scenarios as well.

Although SAM has proven very successful in practice in DPML, there is a notable divide between the established motivation for SAM of finding a flat loss landscape and the empirical gains achieved. Fundamentally, the work we present here aims to justify the usage of SAM by appealing to a different set of principles than those originally used to derive the algorithm. Specifically, we show that training with SAM and incorporating early stopping can effectively resolve the divergence issue of flatness. This provides a natural perspective to better leverage the benefits brought by SAM in DPML. Our results demonstrate that this approach leads to more robust and reliable training outcomes, providing a more cohesive theoretical foundation for the practical successes observed with SAM in DPML.

7 Related Work

Sharpness Minimization Strategies. The notion that flat minima can enhance generalization dates back to the work of Hochreiter and Schmidhuber [33], inspiring numerous methods aimed at optimizing for more robust minima. These methods range from random perturbations such as dropout [65] and Entropy-SGD [13] to worst-case perturbations such as SAM [29] and its variations [22, 40, 82], which indicates that SAM is particularly effective for vision transformers on large-scale datasets like ImageNet, as standard transformers tend to converge to very sharp minima. Parallel research on the implicit bias of SGD suggests that it implicitly minimizes certain hidden complexity measures related to the flatness of minima [35, 36, 63, 76]. For example, Izmailov et al. [35] propose averaging weights during SGD to enhance generalization, motivated by a reduction in sharpness. Smith et al. [63] derive an implicit regularization term for SGD based on the gradient norm. Sharpness-related metrics based on the Hessian have also been extensively studied. For instance, Cohen et al. [16], Arora et al. [5], and Damian et al. [18] empirically and theoretically examine the regime of full-batch gradient descent, where the maximum eigenvalue of the Hessian is inversely proportional to the learning rate. Blanc et al. [7], Li et al. [45], and Damian et al. [17] find that label-noise SGD implicitly minimizes the trace of the Hessian, serving as a proxy for standard SGD.

Loss Landscape of DPML. Recent studies have investigated the differences between the loss landscapes of DPSGD and standard SGD. Bu et al. [9] analyzed the convergence of DP training, focusing on different clipping methods and noise addition. Wang et al. [69] first highlighted the issue of DPSGD getting stuck in local minima due to training instability. They suggested that averaging the gradients of neighborhoods in the parameter space can achieve a smoother loss landscape and improved performance, but this approach incurs signifi-

cant computational cost. Instead of averaging, Shamsabadi et al. [61] proposed that using loss functions with smaller norms can reduce the impact of clipping, thereby creating a smoother loss function. Park et al. [56] proposed DPSAT, which incorporates a single-step SAM into DPSGD to reduce the negative effects of per-example gradient clipping and noise addition. This method is currently the state-of-the-art in this area.

Wang et al. [70] proposed DPAdapter, which applies SAM on public data to improve parameter robustness and flatten the loss landscape before fine-tuning on private data, thereby mitigating the negative effects of DP noise addition. However, this method can be computationally expensive, as it applies SAM during the pre-training stage, where data volume is typically much larger than in fine-tuning. Moreover, it employs a two-step SAM procedure, doubling the computational cost compared to our single-step SAM approach. Additionally, DPAdapter is tailored for transfer learning and relies on the transferability of sharpness—a property influenced by the similarity between upstream and downstream datasets—which limits its effectiveness when such similarity is low. In contrast, our method applies SAM directly on the same data, thus avoiding this limitation.

8 Conclusion

In this work, we introduced *sharpness-aware initialization* (SAI), a technique that achieves a flatter loss landscape and addresses the challenges posed by Differential Privacy (DP) noise in machine learning models. By adopting a two-phase strategy—sharpness-aware initialization (SAI) followed by standard DPSGD—SAI effectively reduces the adverse effects of DP noise. Our extensive evaluations on MNIST, FashionMNIST, CIFAR-10/100, and SVHN demonstrate that SAI significantly enhances the accuracy of DPML algorithms across a range of model architectures. These promising results suggest that SAI can serve as an important tool for future developments in privacy-preserving machine learning.

Ethical Considerations

We adhered to recognized ethical guidelines and best practices throughout this study. All evaluations took place in a local environment using publicly available datasets. Because this research solely relies on public data and does not involve human participants, it was not classified as human subjects research by our Institutional Review Board (IRB).

Open Science

The implementation of SAI-DPSGD is available at: https://zenodo.org/records/15490109.

References

[1] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 308–318, 2016.

[2] K. Ahn, A. Jadbabaie, and S. Sra. How to escape sharp minima. *arXiv preprint arXiv:2305.15659*, 2023.

[3] G. Andrew, O. Thakkar, B. McMahan, and S. Ramaswamy. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems*, 34:17455–17466, 2021.

[4] M. Andriushchenko and N. Flammarion. Towards understanding sharpness-aware minimization. In *International Conference on Machine Learning*, pages 639–668. PMLR, 2022.

[5] S. Arora, Z. Li, and A. Panigrahi. Understanding gradient descent on the edge of stability in deep learning. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 948–1024. PMLR, 2022.

[6] J. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger. Understanding batch normalization. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7705–7716, 2018.

[7] G. Blanc, N. Gupta, G. Valiant, and P. Valiant. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In J. D. Abernethy and S. Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 483–513. PMLR, 2020.

[8] Z. Bu, J. Mao, and S. Xu. Scalable and efficient training of large convolutional neural networks with differential privacy. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[9] Z. Bu, H. Wang, Z. Dai, and Q. Long. On the convergence and calibration of deep learning with differential

privacy. *Transactions on machine learning research*, 2023, 2023.

[10] Z. Bu, Y. Wang, S. Zha, and G. Karypis. Automatic clipping: Differentially private deep learning made easier and stronger. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[11] M. Bun and T. Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[12] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In N. Heninger and P. Traynor, editors, *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 267–284. USENIX Association, 2019.

[13] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. T. Chayes, L. Sagun, and R. Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[14] C.-F. R. Chen, Q. Fan, and R. Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 357–366, 2021.

[15] A. Cheng, J. Wang, X. S. Zhang, Q. Chen, P. Wang, and J. Cheng. Dpnas: Neural architecture search for deep learning with differential privacy. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 6358–6366, 2022.

[16] J. Cohen, S. Kaur, Y. Li, J. Z. Kolter, and A. Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[17] A. Damian, T. Ma, and J. D. Lee. Label noise SGD provably prefers flat global minimizers. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 27449–27461, 2021.

[18] A. Damian, E. Nichani, and J. D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[19] S. De, L. Berrada, J. Hayes, S. L. Smith, and B. Balle. Unlocking high-accuracy differentially private image classification through scale. *CoRR*, abs/2204.13650, 2022.

[20] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1019–1028. PMLR, 2017.

[21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[22] J. Du, D. Zhou, J. Feng, V. Y. F. Tan, and J. T. Zhou. Sharpness-aware training for free. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[23] S. S. Du, X. Zhai, B. Póczos, and A. Singh. Gradient descent provably optimizes over-parameterized neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[24] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 265–284, 2006.

[25] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[26] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[27] C. Dwork, G. N. Rothblum, and S. Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual*

Symposium on Foundations of Computer Science, pages 51–60. IEEE, 2010.

[28] G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In G. Elidan, K. Kersting, and A. Ihler, editors, *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017.

[29] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[30] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In I. Ray, N. Li, and C. Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1322–1333. ACM, 2015.

[31] S. Gopi, Y. T. Lee, and L. Wutschitz. Numerical composition of differential privacy. *Advances in Neural Information Processing Systems*, 34:11631–11642, 2021.

[32] H. He, G. Huang, and Y. Yuan. Asymmetric valleys: Beyond sharp and flat local minima. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 2549–2560, 2019.

[33] S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Comput.*, 9(1):1–42, 1997.

[34] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[35] P. Izmailov, D. Podoprikhin, T. Garipov, D. P. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. In A. Globerson and R. Silva, editors, *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 876–885. AUAI Press, 2018.

[36] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[37] H. Kim, J. Park, Y. Choi, and J. Lee. Stability analysis of sharpness-aware minimization. *arXiv preprint arXiv:2301.06308*, 2023.

[38] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[39] A. Krizhevsky. Learning multiple layers of features from tiny images. Tech. report, University of Toronto, 2009.

[40] J. Kwon, J. Kim, H. Park, and I. K. Choi. ASAM: adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5905–5914. PMLR, 2021.

[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[42] Y. LeCun, L. Bottou, G. B. Orr, and K. Müller. Efficient backprop. In G. Montavon, G. B. Orr, and K. Müller, editors, *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pages 9–48. Springer, 2012.

[43] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.*, 37(3):50–60, 2020.

[44] X. Li, F. Tramèr, P. Liang, and T. Hashimoto. Large language models can be strong differentially private learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

[45] Z. Li, T. Wang, and S. Arora. What happens after SGD reaches zero loss? –a mathematical framework. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[46] Y. Ma, X. Zhu, and J. Hsu. Data poisoning against differentially-private learners: Attacks and defenses. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence,*

IJCAI 2019, Macao, China, August 10-16, 2019, pages 4732–4738. ijcai.org, 2019.

[47] H. Mehta, A. Thakurta, A. Kurakin, and A. Cutkosky. Large scale transfer learning for differentially private image classification. *CoRR*, abs/2205.02973, 2022.

[48] I. Mironov. Rényi differential privacy. In *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*, pages 263–275. IEEE Computer Society, 2017.

[49] R. Mulayoff and T. Michaeli. Unique properties of flat minima in deep networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7108–7118. PMLR, 2020.

[50] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone. A review on weight initialization strategies for neural networks. *Artif. Intell. Rev.*, 55(1):291–322, 2022.

[51] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2843–2851, 2011.

[52] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5947–5956, 2017.

[53] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

[54] N. Papernot, M. Abadi, Ú. Erlingsson, I. J. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[55] N. Papernot, A. Thakurta, S. Song, S. Chien, and Ú. Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 9312–9321. AAAI Press, 2021.

[56] J. Park, H. Kim, Y. Choi, and J. Lee. Differentially private sharpness-aware training. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, pages 27204–27224, 2023.

[57] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.

[58] L. Prechelt. Early stopping - but when? In G. Montavon, G. B. Orr, and K. Müller, editors, *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pages 53–67. Springer, 2012.

[59] M. A. Rahman, T. Rahman, R. Laganière, and N. Mohammed. Membership inference attack against differentially private deep learning model. *Trans. Data Priv.*, 11(1):61–79, 2018.

[60] L. Sagun, U. Evci, V. U. Güney, Y. N. Dauphin, and L. Bottou. Empirical analysis of the hessian of overparametrized neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018.

[61] A. S. Shamsabadi and N. Papernot. Losing less: A loss for differentially private deep learning. *Proc. Priv. Enhancing Technol.*, 2023(3):307–320, 2023.

[62] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 3–18. IEEE Computer Society, 2017.

[63] S. L. Smith and Q. V. Le. A bayesian perspective on generalization and stochastic gradient descent. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[64] C. Song and V. Shmatikov. Auditing data provenance in text-generation models. In A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 196–206. ACM, 2019.

[65] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[66] Q. Tang, F. Shpilevskiy, and M. Lécuyer. Dp-adambc: Your dp-adam is actually DP-SGD (unless you apply bias correction). In M. J. Wooldridge, J. G. Dy, and S. Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 15276–15283. AAAI Press, 2024.

[67] F. Tramèr and D. Boneh. Differentially private learning needs better features (or much more data). In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[68] Y. Tsuzuku, I. Sato, and M. Sugiyama. Normalized flat minima: Exploring scale invariant definition of flat minima for neural networks using pac-bayesian analysis. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9636–9647. PMLR, 2020.

[69] W. Wang, T. Wang, L. Wang, N. Luo, P. Zhou, D. Song, and R. Jia. Dplis: Boosting utility of differentially private deep learning via randomized smoothing. *Proc. Priv. Enhancing Technol.*, 2021(4):163–183, 2021.

[70] Z. Wang, R. Zhu, D. Zhou, Z. Zhang, J. Mitchell, H. Tang, and X. Wang. Dpadapter: Improving differentially private deep learning through noise tolerance pre-training. In *33st USENIX Security Symposium (USENIX Security 24)*, 2024.

[71] C. Wei, M. Zhao, Z. Zhang, M. Chen, W. Meng, B. Liu, Y. Fan, and W. Chen. Dpmlbench: Holistic evaluation of differentially private machine learning. *CoRR*, abs/2305.05900, 2023.

[72] R. Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

[73] Y. Wu and K. He. Group normalization. *Int. J. Comput. Vis.*, 128(3):742–755, 2020.

[74] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[75] Z. Xie, I. Sato, and M. Sugiyama. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[76] C. Xing, D. Arpit, C. Tsirigotis, and Y. Bengio. A walk with SGD. *CoRR*, abs/1802.08770, 2018.

[77] Z. Yao, A. Gholami, Q. Lei, K. Keutzer, and M. W. Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4954–4964, 2018.

[78] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, G. Cormode, and I. Mironov. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*, 2021.

[79] D. Yu, S. Naik, A. Backurs, S. Gopi, H. A. Inan, G. Kamath, J. Kulkarni, Y. T. Lee, A. Manoel, L. Wutschitz, S. Yekhanin, and H. Zhang. Differentially private fine-tuning of language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

[80] D. Yu, H. Zhang, W. Chen, and T.-Y. Liu. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[81] Y. Zhu, X. Yu, M. Chandraker, and Y. Wang. Private-knn: Practical differential privacy for computer vision. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11851–11859, 2020.

[82] J. Zhuang, B. Gong, L. Yuan, Y. Cui, H. Adam, N. C. Dvornek, S. Tatikonda, J. S. Duncan, and T. Liu. Surrogate gap minimization improves sharpness-aware training. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Table 5: Hyperparameters for training on MNIST, FashionMNIST, CIFAR-10, and SVHN.

Dataset		MNIST		FashionMNIST		CIFAR-10		SVHN
Architecture		GNResNet-10	DPNAS-MNIST	GNResNet-10	DPNAS-MNIST	CNN-Tanh with SELU	DPNAS-CIFAR10	DPNAS-CIFAR10
Optimizer		SGD	SGD	SGD	SGD	SGD	SGD	SGD
Total Epoch		40	40	40	40	30	30	30
Batch size		2048	2048	2048	2048	2048	2048	2048
Momentum		0.9	0.9	0.9	0.9	0.9	0.9	0.9
η_1		2	2	2	2	2	2	2
η_2		0.1	0.1	0.1	0.1	0.1	0.1	0.1
C_2		0.1	0.1	0.1	0.1	0.1	0.1	0.1
C_1	$\varepsilon = 1$	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	$\varepsilon = 2$	0.2	0.1	0.2	0.2	0.2	0.2	0.2
	$\varepsilon = 3$	0.2	0.2	0.2	0.2	0.2	0.2	0.2
ε_1	$\varepsilon = 1$	0.8	0.8	0.8	0.8	0.8	0.8	0.8
	$\varepsilon = 2$	1.8	1.8	1.8	1.8	1.8	1.8	1.8
	$\varepsilon = 3$	2.7	2.7	2.7	2.7	2.7	2.7	2.7
e_1	$\varepsilon = 1$	15	15	15	15	10	10	10
	$\varepsilon = 2$	15	15	15	15	10	10	10
	$\varepsilon = 3$	20	20	20	20	15	15	15
ρ	$\varepsilon = 1$	0.05	0.05	0.03	0.05	0.3	0.1	0.05
	$\varepsilon = 2$	0.05	0.05	0.03	0.05	0.05	0.05	0.05
	$\varepsilon = 3$	0.03	0.05	0.03	0.03	0.05	0.05	0.05

A Hyperparameters

The clipping threshold C_2 is fixed at 0.1. We adopt cross-entropy as the loss function and utilize SGD as the base optimizer with a momentum of 0.9. The batch size is set to 2048. The learning rate η_1 is fixed at 2.0, while the learning rate η_2 is fixed at 0.1. The total number of epochs is set to 40 for MNIST and FashionMNIST, and to 30 for CIFAR-10 and SVHN. We conducted a hyperparameter search on clipping threshold C_1 on $\{0.1, 0.2\}$, radius $\rho = \{0.03, 0.05, 0.1, 0.3\}$, epoch for SAI $e_1 = \{10, 15, 20\}$, and the portion of budget allocated for SAI on $\{0.8, 0.9\}$.

Note that the optimal clipping threshold C_1, radius ρ, epoch for SAI e_1, and budget allocated for SAI can vary according to the total privacy budget of DP training. For a detailed explanation of DPNAS architectures as described in [15], please refer to their paper for further details. For DPSAT and DPSGD, we use the implementations provided by the authors and the optimal hyperparameters identified in the DPSAT paper for fair comparison [56].

B Proof of Theorem 1

This section establishes a formal proof of Theorem 1. We begin with the assumptions for our analysis, present the theorem in full detail, and then provide its proof.

Assumption 1 (Followed by [2]). *There exists a constant $\xi > 0$ such that within the ξ-neighborhood of the set of local minima W^*, the following properties hold:*

1. *ℓ is four-times continuously differentiable.*
2. *The limit map under gradient flow Φ (Definition 2) is well-defined, is twice Lipschitz-differentiable, and $\Phi(\boldsymbol{w}) \in W^*$. Moreover, the gradient flow starting at $\boldsymbol{w}$ remains in the ξ-neighborhood of W^*.*
3. *A local Polyak-Łojasiewicz (PL) condition is satisfied:*

$$\ell(\boldsymbol{w}) - \ell\big(\Phi(\boldsymbol{w})\big) \;\leq\; \frac{1}{2\alpha}\big\|\nabla\ell(\boldsymbol{w})\big\|^2.$$

Theorem 1 (Formal). *Let Assumption 1 hold, and suppose each ℓ_i is four-times continuously differentiable with β-Lipschitz gradients in the ξ-neighborhood of W^*. Let $\varepsilon > 0$ be sufficiently small and $\delta \in (0,1)$. Assume $\boldsymbol{w}_0$ is ξ-close to W^*. A sharpness-aware (SAM-like) method finds an $\big(O(\varepsilon^{1.5}), \sqrt{\varepsilon}\big)$-flat minimum with probability at least $1 - O(\delta)$ in T_1 epochs. From this point, a standard gradient descent method with step size $\eta = O(\varepsilon)$ reaches a $\big(\varepsilon, \sqrt{\varepsilon}\big)$-flat minimum in T_2 additional epochs. Then:*

$$\frac{1}{d\,\varepsilon\log(1/\varepsilon)} \;\leq\; \frac{T_1}{T_2} \;\leq\; \frac{\delta}{\varepsilon^3\nu^3},$$

where $\nu = \min\{d, \varepsilon^{-1/3}\}$ and d is the dimension of the parameter space.

Proof. **Outline.** We analyze the iteration complexities T_1 and T_2 for the two-phase procedure. The sharpness-aware phase concludes within

$$T_1 \;=\; O\Big(d^{-1}\varepsilon^{-2}\max\big\{1,\, \tfrac{1}{\delta^3\varepsilon^4}\big\}\Big),$$

while the subsequent gradient-descent phase requires

$$T_2 \;=\; O\big(\varepsilon^{-1}\log\tfrac{1}{\varepsilon}\big)$$

to refine the flatness and loss further. We derive lower/upper bounds on each expression and then combine them to establish the ratio $\frac{T_1}{T_2}$.

Step 1: Bounds for T_2. By definition,

$$T_2 \;=\; O\Big(\varepsilon^{-1}\log\big(\tfrac{1}{\varepsilon}\big)\Big).$$

Hence, there is an upper bound

$$T_2 \;\le\; \frac{\log\big(\frac{1}{\varepsilon}\big)}{\varepsilon}.$$

On the other hand, standard arguments about local refinement and curvature yield a lower bound of the form

$$T_2 \;\ge\; \frac{1}{d\,\varepsilon^3\,\nu^3\,\delta^4},$$

where $\nu = \min\{\,d,\,\varepsilon^{-1/3}\}$ and d is the parameter dimension. Combining these, we have:

$$\frac{1}{d\,\varepsilon^3\,\nu^3\,\delta^4} \;\le\; T_2 \;\le\; \frac{\log\big(\frac{1}{\varepsilon}\big)}{\varepsilon}.$$

Step 2: Bounds for T_1. For the sharpness-aware phase,

$$T_1 \;=\; O\Big(d^{-1}\,\varepsilon^{-2}\max\big\{1,\,\tfrac{1}{\delta^3\varepsilon^4}\big\}\Big).$$

In the *worst-case* scenario, where $\delta^3\varepsilon^4 < 1$, we get:

$$T_1 \;\le\; O\Big(\tfrac{1}{d\,\delta^3\varepsilon^6}\Big),$$

while in the *best-case* scenario, where $\delta^3\varepsilon^4 \ge 1$, it follows that:

$$T_1 \;\ge\; O\Big(\tfrac{1}{d\varepsilon^2}\Big).$$

Step 3: Combining the bounds for $\frac{T_1}{T_2}$. Consider

$$\frac{T_1}{T_2} \;=\; \frac{O\big(d^{-1}\,\varepsilon^{-2}\max\{1,\,\frac{1}{\delta^3\varepsilon^4}\}\big)}{O\big(\varepsilon^{-1}\log(1/\varepsilon)\big)}.$$

Using the *upper* bound for T_1 and the *lower* bound for T_2:

$$\frac{T_1}{T_2} \;\le\; \frac{\frac{1}{d\,\delta^3\varepsilon^6}}{\frac{1}{d\,\varepsilon^3\nu^3\,\delta^4}} \;=\; \frac{\delta}{\varepsilon^3\nu^3}.$$

Using the *lower* bound for T_1 and the *upper* bound for T_2:

$$\frac{T_1}{T_2} \;\ge\; \frac{\frac{1}{d\varepsilon^2}}{\frac{\log(1/\varepsilon)}{\varepsilon}} \;=\; \frac{1}{d\,\varepsilon\log(1/\varepsilon)}.$$

Hence,

$$\frac{1}{d\,\varepsilon\log(1/\varepsilon)} \;\le\; \frac{T_1}{T_2} \;\le\; \frac{\delta}{\varepsilon^3\nu^3}.$$

These inequalities complete the derivation of the claimed bounds on the ratio of the two phases, confirming that an appropriate balance between the sharpness-aware stage and the subsequent gradient-based refinement is crucial for achieving a $\big(\varepsilon,\sqrt{\varepsilon}\big)$-flat minimum in both theory and practice.

□