

An Adaptive Technique for Neural Network Training with Private Features and Public Labels

Islam A. Monir¹, Muhamad I. Fauzan¹, Gabriel Ghinita², Mohamed M. Abdallah¹

¹College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar

²Computer Science Department, University of Massachusetts Boston, MA, USA

E-mail: {ismo58166, mufa68183, moabdallah}@hbku.edu.qa, gabriel.ghinita@umb.edu

Received 13 October 2024; received in revised form 29 October 2025; accepted 28 December 2025

Keywords. Differential Privacy, Machine Learning, Neural Networks

Abstract. Differentially-private stochastic gradient descent (DP-SGD) represents the de-facto standard for privacy-preserving training of neural networks (NNs) under the differential privacy (DP) model. Its canonical formulation assumes that both the input features and the corresponding labels of training instances require protection. Newer developments explore scenarios in which only the labels are private, while the features are public. Doing so decreases the amount of required noise, leading to improved model accuracy. We investigate a complementary and underexplored setting where labels are non-sensitive, but the input features contain private information. Instead of perturbing gradients, our proposed methodology for training private NNs adds noise at a designated *sanitization layer* within the network. We analyze key architectural and algorithmic trade-offs inherent in this design and demonstrate how modifying the network architecture to reflect these considerations can lead to improved predictive performance. We also devise two adaptive algorithm optimizations: the first one identifies early stopping conditions in the learning process in order to save privacy budget and boost the protection strength; the second customizes the clipping threshold at each learning iteration in order to improve accuracy. Extensive experiments on real data show that our approach significantly outperforms the DP-SGD baseline.

1 Introduction

The privacy threats that arise when neural networks (NNs) are trained on large-scale datasets containing individual-level information are well-understood [1]. Trained models are vulnerable to inference attacks, where sensitive personal attributes may be disclosed when models are queried. One way to address this threat is by employing Differential Privacy (DP) [2], which provides formal guarantees on individuals' data protection.

A common approach to training NN models is stochastic gradient descent (SGD), which adjusts model parameters by minimizing the discrepancy between predicted and true labels through back-propagation (see Section 2.2). Conventional SGD assumes both input

features and corresponding labels are publicly available [3]. Its differentially-private counterpart DP-SGD [4] addresses privacy risks as illustrated in Figure 1b. DP-SGD introduces two critical modifications: (1) *gradient clipping* to bound sensitivity, a core DP concept that influences the scale of injected noise (further discussed in Section 2.1), and (2) the addition of calibrated noise to the clipped gradients to ensure privacy.

The space of privacy-preserving NN training approaches can be illustrated using the quadrant representation of Figure 1a. The axes capture whether features and labels are sensitive or not. Conventional SGD and DP-SGD represent opposing cases of the privacy spectrum: SGD provides no privacy protection, whereas DP-SGD ensures the privacy of both input features and labels. A third category emerged recently, where input features are non-sensitive, but the labels are private [5]. The label information is obfuscated, based on the premise that while features might be observable or known to adversaries, the relationship to their labels is sensitive. This adjustment reduces the noise burden and can yield improved model accuracy.

Despite the progress in label-private learning, another relevant yet underexplored setting arises when the labels are public, but the input features require protection. For instance, consider an online platform where pseudonymous users assign ratings to products. The rating values (labels) are public, while users may wish to keep personal characteristics—such as age, gender, ethnicity, income, or education—confidential. Similarly, in clinical datasets documenting treatment outcomes (e.g., changes in blood sugar or quality-of-life scores), the labels may be intentionally shared, whereas sensitive features, including genetic profiles or lifestyle data, demand stringent privacy safeguards.

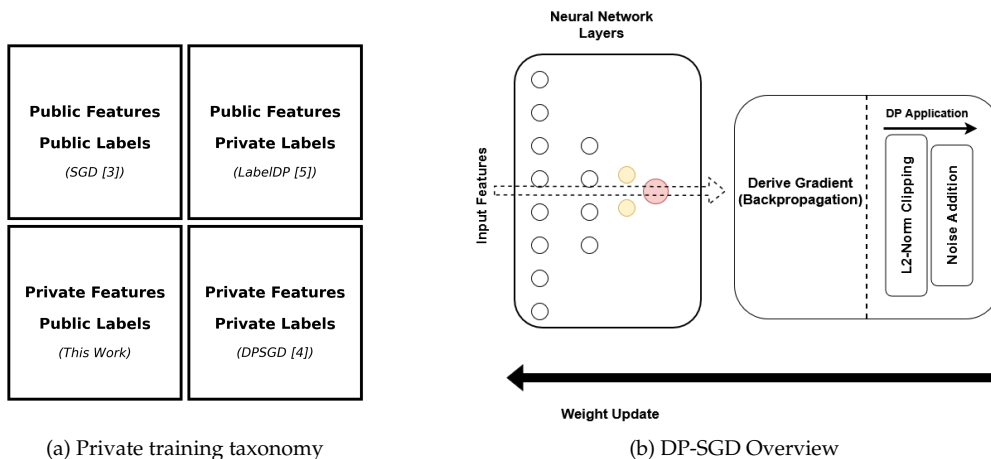


Figure 1: Private NN Learning Taxonomy and DP-SGD

We introduce a novel privacy-preserving technique that departs fundamentally from the conventional DP-SGD framework. While DP-SGD enforces privacy by injecting noise into the gradients—a critical juncture in the training pipeline—this approach often compromises utility by suppressing valuable learning signals. In contrast, our method introduces noise at an intermediate layer within the network, which we designate as the *sanitization layer*. This design permits greater retention of informative representations downstream.

We systematically analyze key architectural parameters, particularly the placement and dimensionality of the sanitization layer, which directly influence the privacy-utility trade-off. By exploring these factors, we propose a set of NN design adaptations aimed at enhancing model performance while adhering to differential privacy constraints.

Next, we focus on two adaptive optimizations that focus on reducing privacy budget consumption and improving accuracy¹. NN training is often performed over a large number of iterations, and each iteration requires expending the scarce privacy budget. In some cases, continuing training after a certain point does not significantly improve accuracy. Therefore, if one devises an appropriate training stop condition, significant privacy budget savings can be achieved. In DP-compliant computation, the privacy budget ϵ is a strict bound that must never be exceeded, such that the required amount of protection is guaranteed. Saving privacy budget is important for two reasons. First, building an accurate model requires more stages than just iterative SGD, such as data pre-processing and tuning. To obtain accurate models, such steps may also need to access the data, hence the privacy budget ϵ must be split among these stages and iterative SGD, according to the sequential composition property of DP [2]. If the iterative SGD step consumes less privacy budget, then more becomes available to the other stages, improving overall accuracy. Second, even when other stages are not required, or they do not consume budget (e.g., they are data independent), performing fewer iterations and stopping early means that the resulting privacy achieved by the algorithm is higher than the input bound, hence the resulting model provides a higher protection level than the one requested at the same level of accuracy, which is also desirable. Therefore, we focus on lowering budget consumption during iterative SGD using early stopping. Since the stop condition depends on the parameters of the current training step, one has to carefully devise the stopping strategy, such that it does not violate the privacy constraint. We propose such an adaptive stopping condition.

Furthermore, the amount of noise added in each iteration in order to achieve DP protection is important in improving accuracy. Specifically, in some iterations, the NN parameter values are large, hence they may be less affected by the noise magnitude, whereas in other iterations the amount of noise obliterates learning ability. We propose a customized strategy for clipping threshold adaptation, which uses information from previous rounds to infer an appropriate threshold to use in the current round. Adapting the noise magnitude can significantly improve the accuracy and convergence of NN training.

Our main contributions are as follows:

- We formalize a novel variant of the private neural network training problem, wherein labels are assumed to be public while input features remain sensitive. This completes the exploration of all four configurations in the feature/label privacy quadrant for NN training.
- We propose an alternative to the standard DP-SGD approach by introducing a privacy mechanism that applies calibrated noise to the outputs of a specifically designated *sanitization layer* within the network, rather than to the gradients.
- We investigate how architectural choices—particularly the location and dimensionality of the sanitization layer—impact model performance under differential privacy constraints.
- We devise a novel training stop condition that uses sanitized information from previous iterations to determine privately whether continuation of training is likely to significantly improve accuracy; if not, an early stopping condition is triggered, which provides significant privacy budget savings.

¹This submission is an extended version of our conference article published in [6]. The additional research content consists of the adaptive techniques for stopping and clipping threshold selection, including the corresponding analysis and experimental evaluation. Sections 5, 6.3 and 6.4 are entirely new.

- We propose an adaptive technique for clipping threshold selection which chooses at each iteration a value that is appropriate for that training step. The decision is made based on analyzing the parameter evolution from previous iterations in a DP-compliant fashion.
- We conduct comprehensive empirical evaluations on real-world datasets, demonstrating that our proposed technique, along with its adaptive architectural enhancements, consistently outperforms baseline approaches.

The remainder of this paper is organized as follows. Section 2 introduces the preliminary concepts and formal definitions required for our approach. Section 3 surveys related work in differential privacy and neural network training. Our proposed methodology is detailed in Section 4. Section 5 presents adaptive optimization strategies, including mechanisms for early stopping and clipping threshold selection. Section 6 provides a comprehensive empirical evaluation comparing our method against established benchmarks. Finally, Section 7 concludes the paper and outlines potential directions for future research.

2 Background

2.1 Differential Privacy

Differential privacy (DP) offers a formalism for statistical disclosure control, ensuring that the presence or absence of a single individual in a dataset has a provably limited influence on the outcome of a computation. This is achieved through the injection of calibrated randomness, thereby constraining adversarial inference.

Definition 1. A randomized algorithm $\mathcal{A} : \mathcal{D} \rightarrow \mathcal{R}$ satisfies (ϵ, δ) -differential privacy if for any two neighboring datasets $\mathcal{D}_1, \mathcal{D}_2 \in \mathcal{D}$ differing in at most one record, and for any measurable subset $\mathcal{S} \subseteq \mathcal{R}$, the following holds:

$$\Pr[\mathcal{A}(\mathcal{D}_1) \in \mathcal{S}] \leq e^\epsilon \Pr[\mathcal{A}(\mathcal{D}_2) \in \mathcal{S}] + \delta \quad (1)$$

This inequality bounds the multiplicative and additive differences in output distributions over neighboring inputs, thereby shielding individual-level information. The parameter $\epsilon > 0$ governs the privacy loss and is commonly referred to as *privacy budget*, while $\delta \in [0, 1]$ quantifies the permissible probability of failure of the privacy guarantee [2, 4, 7].

Central to the design of DP mechanisms is the concept of *sensitivity*, which captures the worst-case effect of a single input perturbation on the function’s output.

Definition 2. Let $\phi : \mathcal{D} \rightarrow \mathbb{R}^d$ be a deterministic function. Its ℓ_2 -sensitivity, denoted $\Delta_2(\phi)$, is defined as:

$$\Delta_2(\phi) = \sup_{\mathcal{D}_1 \sim \mathcal{D}_2} \|\phi(\mathcal{D}_1) - \phi(\mathcal{D}_2)\|_2 \quad (2)$$

One of the most widely adopted DP mechanisms, particularly in gradient-based optimization, is the Gaussian mechanism [4]:

Definition 3. Let $\phi : \mathcal{D} \rightarrow \mathbb{R}^d$ have ℓ_2 -sensitivity Δ . The Gaussian mechanism perturbs the output of ϕ by adding isotropic Gaussian noise: $\mathcal{A}(\mathcal{D}) = \phi(\mathcal{D}) + \mathbf{z}$, where $\mathbf{z} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_d)$. Then \mathcal{A} satisfies (ϵ, δ) -DP provided:

$$\sigma \geq \frac{\Delta \sqrt{2 \ln(1.25/\delta)}}{\epsilon} \quad (3)$$

2.2 DP-SGD

Stochastic Gradient Descent (SGD) remains the dominant paradigm for training neural networks, particularly in large-scale supervised learning. It iteratively refines model parameters θ by computing the gradient of a loss function over mini-batches of the training set. The canonical update rule is:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla \mathcal{L}(\theta_t; \mathbf{x}^{(i)}, \mathbf{y}^{(i)}),$$

where η denotes the learning rate, and \mathcal{L} is the empirical loss over batch $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ [8, 3].

Differentially Private SGD (DP-SGD) [4] modifies this process by applying a two-step privacy-preserving transformation to each gradient: (1) *clipping*, to ensure bounded ℓ_2 -sensitivity; and (2) *noise addition*, via the Gaussian mechanism (Eq. (3)). This design is illustrated in Figure 1b, where privacy enforcement is localized at the gradient level—a bottleneck in the learning pipeline.

A major challenge in applying DP to iterative algorithms such as SGD stems from the cumulative nature of privacy loss across training steps. According to the standard *sequential composition theorem* [9], the overall privacy budget grows approximately as the sum of the per-iteration budgets, resulting in rapid budget exhaustion during long training processes. The cumulative privacy cost under this conventional composition can be approximated as:

$$\varepsilon_{\text{total}} \approx \sqrt{2 \sum_{t=1}^k \frac{\ln(1.25/\delta)}{\sigma_t^2}}, \quad (4)$$

where σ_t denotes the Gaussian noise scale at iteration t , and k is the total number of training iterations. This formulation reflects the linear accumulation of privacy loss and highlights the scalability limitation of naive composition.

To mitigate this issue, Abadi et al. [4] introduced the *Moments Accountant (MA)* framework, which provides a substantially tighter accounting of privacy loss by tracking higher-order moments of the privacy random variable. The MA framework exploits the randomness induced by *Poisson subsampling*—a process in which each record is independently included in a training batch with a fixed probability. This stochastic sampling reduces the expected contribution of any individual record to the gradient computation, thereby lowering the effective privacy loss per iteration. Combined with Gaussian noise addition, this yields significantly improved privacy bounds, allowing DP-SGD to achieve strong differential privacy guarantees even over many training epochs.

3 Related Work

The seminal contribution of Abadi et al. [4] introduced both the DP-SGD algorithm and the Moments Accountant framework for tighter privacy budget tracking. Since then, a range of follow-up studies have proposed refinements and extensions aimed at enhancing model utility under differential privacy constraints. The work in [10] introduces the Gradient Index Pruning (GIP) mechanism as a solution, which involves pruning gradients while preserving essential components to reduce their sizes. This strategic approach leads to a remarkable 12% accuracy boost over state-of-the-art methods. Several recent studies have explored adaptive strategies for privacy budget allocation and optimization in differentially private learning. Zhang et al. [11] and Koskela et al. [12] propose mechanisms

in which the magnitude of noise varies across training iterations. In particular, [11] introduces a decaying noise schedule in which Gaussian noise added to gradients diminishes over time, while [12] adjusts the learning rate dynamically in response to estimated errors, thereby improving training stability under DP constraints. Chen et al. [13] present a private backtracking line search algorithm that allocates privacy budget based on the estimated reliability of noisy gradients and loss values, mitigating the risk of over-consuming the budget. More recently, DiceSGD [14] addresses bias introduced by gradient clipping by incorporating an error-feedback mechanism to guide the adaptive selection of clipping thresholds, effectively reducing clipping-induced distortion.

A recent survey studied different adaptive techniques and their influence on the privacy-utility tradeoff [15]. Included in the study are different adaptation strategies for selecting clipping thresholds. One such approach was presented in [16], where the authors highlighted the difficulty of selecting a suitable clipping threshold C , especially for multiple layers. They proposed that the clipping bound for the current batch be scaled proportionally to the ℓ_2 norm of the previous batch, using a constant factor α . He et al. [17] proposed an adaptive clipping method using quantile estimation for per-layer clipping. A small fraction (1%-10%) of the privacy budget estimates the target quantile, setting thresholds C_1, \dots, C_K . The thresholds are adjusted based on the number of gradients clipped before each update.

The role of normalization techniques in improving model utility under differential privacy has also been investigated. Davody et al. [18] examine the effect of incorporating batch normalization within the DP-SGD framework and demonstrate that it can partially offset the loss in accuracy typically induced by noise injection. More recently, Nguyen et al. [19] propose Batch Clipping (BC) and Adaptive Layerwise Clipping (ALC) as alternative strategies to further enhance model accuracy and accelerate convergence. These methods adjust clipping norms either at the batch or layer level, enabling more fine-grained control over the noise-utility trade-off.

The studies in [20, 21] examine the influence of hyperparameters on model performance under DP-SGD and propose methods for their efficient tuning. The study in [20] systematically explores various hyperparameters, encompassing activation functions, learning rates, and normalization techniques. Notably, the authors scrutinize the impact of using Bounded RELU as an activation function in DP-SGD, revealing its potential to enhance utility without compromising privacy—a crucial finding in the privacy-utility trade-off landscape. In [21], the authors propose three automated hyperparameter tuning strategies—evolutionary search, Bayesian optimization, and reinforcement learning—for improving performance under DP-SGD. Experimental evaluations show that these methods yield significant accuracy gains compared to conventional grid search. Among them, evolutionary and Bayesian approaches strike the best balance, achieving high model utility while minimizing privacy leakage.

Cheng et al. [22] present a framework that combines differentially private learning with neural architecture search. Their approach automates the discovery of model architectures well-suited for private training by embedding DP-aware evaluation into the search process. Similarly, Remerscheid et al. [23] introduce SmoothNet, a specialized neural architecture designed to enhance learning under differential privacy. The study identifies architectural elements contributing to robust performance and demonstrates that SmoothNet outperforms standard models on benchmark datasets in both accuracy and stability.

Several recent works support individualized privacy requirements within the DP framework, particularly focusing on DP-SGD [24, 25]. Methods such as privacy-aware sampling, grouping, and individualized noise scaling allow for personalized differential pri-

vacy (PDP) [15], enabling the allocation of larger privacy budgets to less sensitive data points while preserving stricter privacy for others. This approach improves utility by tailoring privacy mechanisms to the sensitivity of individual data points, thereby enhancing model accuracy without compromising privacy.

More closely related to our approach, label differential privacy (label-DP) [5, 26] addresses settings in which only the labels of training data are considered sensitive. These works focus on training models that satisfy label-DP guarantees, often leveraging unsupervised or semi-supervised learning paradigms. By tailoring noise injection to the label space, they achieve the same level of privacy with reduced noise compared to standard DP-SGD, thereby enhancing the privacy-utility trade-off. In contrast, our work considers the inverse scenario: training on datasets where input features are private and labels are publicly known. We additionally investigate how architectural decisions—particularly the placement and design of intermediate layers—influence model accuracy under this privacy regime.

4 Proposed Approach

We propose a privacy-preserving training methodology tailored to settings where input features are private and labels are public. This design raises several key questions that we address in this section:

1. *Where should DP-compliant noise be applied in the NN pipeline?* DP-SGD adds noise at the gradient level, exploiting its role as a chokepoint to control sensitivity. While effective for protecting labels, this placement imposes unnecessary constraints when labels are public. We instead introduce a sanitization layer, allowing greater flexibility in managing the impact of noise on model performance.
2. *How should sensitivity clipping be implemented?* Traditional DP-SGD uses ℓ_2 -norm clipping on gradients, which causes all gradient elements to be scaled uniformly. This allows outliers to disproportionately influence updates. In contrast, we apply element-wise clipping at the sanitization layer, reducing the impact of outliers and introducing a trade-off between sensitivity and accuracy (see Section 4.3).
3. *How should the NN architecture be adapted for an optimal privacy-utility trade-off?* The flexibility in selecting both the sanitization layer’s location and its dimensionality introduces several design choices that significantly influence model performance. We explore these in detail.

4.1 Sanitization Layer

Before introducing our sanitization layer approach, we first provide a concise description of the serious limitation introduced by DP-SGD’s clipping approach. As noted in Section 2.2, DP-SGD enforces privacy by clipping and perturbing gradients, leveraging the gradient’s role as a chokepoint in the neural network (NN) pipeline. While this choice is convenient, as it is easy to quantify and control the sensitivity of the computation, and it does reflect for the influence of both features and labels, it is also quite inflexible. Specifically, DP-SGD performs global ℓ_2 -norm clipping, where all elements of the gradient vector are *uniformly scaled down* to limit the overall norm. This can lead to substantial accuracy degradation, especially when the gradient contains outliers.

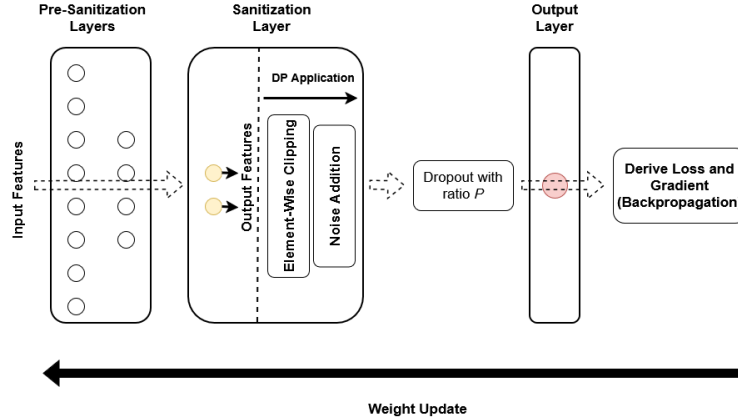


Figure 2: Proposed Sanitization Layer Approach

Specifically, a single gradient component with disproportionately large magnitude will dominate the norm. In such cases, ℓ_2 -norm clipping reduces all components by the same factor, equal to the ratio between the threshold C and the original norm, effectively suppressing all the information from the entries with smaller magnitudes, an aspect also noted in [4].

Our method adopts a different strategy. Rather than perturbing gradients, we apply clipping and noise addition earlier in the pipeline at a designated *sanitization layer*, as illustrated in Figure 2. This layer, placed prior to gradient computation, produces a set of sanitized values whose dimensionality equals the layer width. These values are subsequently propagated forward through the network, allowing for richer information flow during learning.

We also replace global ℓ_2 -norm clipping with *element-wise clipping*, where each component is independently bounded and perturbed. This prevents outliers from disproportionately influencing the entire layer output and contributes to improved stability and faster convergence.

A critical design decision is the placement of the sanitization layer. Applying noise too early in the network (e.g., near the input layer) can degrade performance, as noise compounds through successive layers. To mitigate this, we perform sanitization at the output of the penultimate layer. This placement offers two advantages: (i) it minimizes the propagation of injected noise and preserves the integrity of downstream computations, and (ii) it ensures the layer is sufficiently wide to dilute the effect of individual noisy components. The relationship between layer width, sensitivity, and accuracy is further analyzed in Section 4.3.

4.2 Bounding Sensitivity and Adding Noise

As outlined in Section 2.2, differential privacy (DP) must be enforced at each training epoch to protect individual contributions in the dataset. In our method, this is achieved by injecting calibrated noise into the output of a designated sanitization layer during each training iteration.

Directly computing the sensitivity of a neural network layer is analytically intractable due to its high dimensionality and non-linear structure. To address this, we follow the standard approach of imposing a deterministic upper bound on sensitivity via clipping. Specifically, let n denote the width of the sanitization layer, and let x represent its output.

We apply element-wise clipping with a threshold C , which ensures that each component of x is individually bounded. In contrast to DP-SGD’s global ℓ_2 -norm clipping of the gradient, this approach prevents any single component from disproportionately affecting the overall sensitivity.

Given element-wise clipping across n dimensions, the resulting ℓ_2 -sensitivity of the layer output becomes $C\sqrt{n}$. To satisfy (ϵ, δ) -DP, we add noise drawn from a Gaussian distribution with variance scaled accordingly:

$$\text{Noise} \sim \mathcal{N}(0, C^2\sigma^2n),$$

where σ is the noise multiplier determined by the desired privacy parameters.

This element-wise perturbation strategy offers two principal advantages over gradient-level DP-SGD:

1. **Noise robustness.** In contrast to DP-SGD, where noise directly perturbs the gradient and can distort the parameter update, our approach injects noise into the forward pass at the sanitization layer. This results in larger gradient magnitudes during back-propagation, prompting the model to adjust its parameters to compensate for the noise. Rather than degrading the update, the noise encourages the network to learn more robust representations, thereby enhancing convergence and stability.
2. **Outlier resilience.** By applying clipping to each element individually, our approach explicitly constrains the influence of outliers. In DP-SGD, a single outlier can skew the global gradient norm, forcing excessive downscaling of all components. In contrast, element-wise clipping localizes the impact, preserving a greater portion of the signal.

4.3 Design Choices and Tradeoffs

We examine the privacy-accuracy trade-offs inherent in our method and investigate how neural network architectural adaptations can be leveraged to enhance model accuracy under differential privacy constraints.

4.3.1 Choosing Sanitization Layer Width.

Selecting an appropriate width n for the sanitization layer is critical for achieving an optimal privacy-utility balance. Increasing n initially improves accuracy, as it allows richer feature representations to flow into gradient computation while reducing the effect of individual outliers due to element-wise clipping. However, since the sensitivity of the layer output grows proportionally to \sqrt{n} under element-wise clipping, larger values of n also increase the magnitude of noise required for differential privacy. Excessive noise can degrade the model’s ability to detect meaningful patterns, potentially resulting in overfitting or underfitting.

We observe that n must be sufficiently large to retain enough feature information for the model to converge effectively. At the same time, it must be constrained to avoid excessive noise amplification. To balance these factors, we adopt a funnel-shaped architecture, wherein a series of fully connected layers gradually reduce dimensionality to the target width n of the sanitization layer. Specifically, we include layers with widths $8n$, $4n$, $2n$, and n , respectively. The output layer is then sized according to the number of target classes in the dataset.

This architectural design, illustrated in Figure 3, serves two purposes: it concentrates feature abstraction as the network deepens, and it enables the application of DP noise to a compact, informative feature set. The progressive narrowing of the feature space ensures that the information retained at the sanitization layer is both concise and expressive, while helping to mitigate the adverse impact of noise on model accuracy.

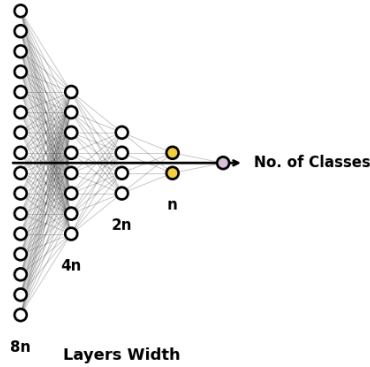


Figure 3: NN Architecture Customization with Funnel-shape Structure of Layers

4.3.2 Choosing Clipping Threshold.

Determining an appropriate clipping threshold C is essential to balancing privacy and model utility. A low threshold constrains the sensitivity of the sanitization layer output, which in turn reduces the scale of the Gaussian noise required for differential privacy. However, this benefit comes at the cost of aggressive truncation of feature values, potentially discarding meaningful information and impairing model accuracy.

On the other hand, a higher clipping threshold allows more of the original layer output to pass through unaltered, thereby preserving representational richness. Yet, this increases the l_2 -sensitivity, resulting in larger noise magnitudes that may degrade learning performance.

Table 1 illustrates this trade-off using representative examples. With a low threshold (e.g., $C = 1.0$), the original values $[2.8, 0.9, 1.5]$ are clipped to $[1.0, 0.9, 1.0]$, which significantly limits sensitivity and leads to smaller added noise (standard deviation 0.5), yielding outputs such as $[1.23, 0.44, 0.95]$. In contrast, with a higher threshold ($C = 5.0$), the same inputs remain unclipped, but the noise scale increases (standard deviation 2.5), producing more perturbed outputs like $[1.92, 3.21, 0.07]$.

These examples underscore the need to calibrate C carefully: overly aggressive clipping sacrifices information, while loose clipping inflates noise and risks performance degradation.

Table 1: Comparison of Low vs. High Clipping Threshold Effects on Sanitization Output

| Parameter | Low Threshold ($C = 1.0$) | High Threshold ($C = 5.0$) |
|------------------------|-----------------------------|------------------------------|
| Original Values | $[2.8, 0.9, 1.5]$ | $[2.8, 0.9, 1.5]$ |
| Clipped Output | $[1.0, 0.9, 1.0]$ | $[2.8, 0.9, 1.5]$ |
| Noise Std. Dev. | 0.5 | 2.5 |
| Noisy Output (Example) | $[1.23, 0.44, 0.95]$ | $[1.92, 3.21, 0.07]$ |

4.3.3 Regularization with Dropout.

Injecting noise during training can serve a dual purpose: enforcing privacy and acting as an implicit regularizer. Moderate noise levels introduce variability that discourages the model from overfitting to specific training instances, thereby improving generalization to unseen data. However, excessive noise can have the opposite effect—if the model starts to overfit to noise, treating it as signal, it may achieve low training error while exhibiting poor test performance.

To mitigate this risk, we apply a dropout layer immediately after the sanitization layer during training. Dropout is a widely used regularization technique that randomly disables a fraction p of the units in a layer during each forward pass. This prevents the model from becoming overly reliant on any particular set of features, thereby enhancing its robustness.

In the context of our method, dropout also affects the sensitivity analysis for differential privacy. Without dropout, sensitivity is scaled by \sqrt{n} , where n is the width of the sanitization layer. However, when a proportion p of the outputs are dropped, the effective number of active units becomes $n(1 - p)$. Consequently, the sensitivity—and thus the required noise magnitude—is reduced. The noise added to the clipped layer outputs then follows a Gaussian distribution:

$$\mathcal{N}(0, C^2 \sigma^2 n(1 - p)),$$

This adjustment ensures that differential privacy guarantees are preserved while also promoting better generalization during training.

Algorithm 1 outlines the complete procedure for our proposed training approach, integrating noise injection, clipping, and architectural adaptations to ensure differential privacy with public labels and private features.

Algorithm 1 Differentially Private Learning with Public Labels and Private Features

Require: Sampling ratio q , dropout probability p , sanitization layer width n , clipping threshold C , noise multiplier σ

Require: Training dataset X

- 1: Partition X into mini-batches $\{B_i\}_{i=1}^m$ using sampling ratio q
 - 2: **for** each mini-batch B_i **do**
 - 3: Forward propagate the features in B_i up to the sanitization layer
 - 4: **for** each activation vector $x \in B_i$ **do**
 - 5: Clip activations: $x \leftarrow \text{clip}(x, C)$
 - 6: Add Gaussian noise: $x \leftarrow x + \mathcal{N}(0, C^2 \sigma^2 n(1 - p))$
 - 7: **end for**
 - 8: Apply dropout with probability p
 - 9: Forward propagate the sanitized activations through the remaining network layers
 - 10: **end for**
-

5 Adaptive Training Techniques

Training a model using Algorithm 1 can lead to situations where the model ceases to improve accuracy over further iterations, and thus it unnecessarily consumes privacy budget. In addition, prolonged inefficacy in learning may also lead to performance degradation

due to overfitting. Throughout training, the outputs of the fully connected layers in the neural network may exhibit a gradual increase or decrease in value, influenced by factors such as weight initialization, activation functions (e.g., ReLU unbounded positive inputs), dynamic learning rates, and the application of regularization. The direction and magnitude of these changes reflect the network’s adaptation to the training data and optimization process. However, when employing static clipping as discussed in Section 4, there is a risk that all the outputs from fully connected layers may be clipped before sanitization, as they exceed the threshold C . When repeated over several iterations, this trend results in stagnant values that can no longer derive useful patterns, thereby preventing effective learning. We propose two adaptive techniques that address these limitations and improve the performance of the proposed approach, namely: (1) *early stopping* and (2) *dynamic clipping thresholds*.

The early stopping technique introduces a *stopping flag* mechanism (defined in Section 5.1) to detect when the model’s learning ability has stagnated, and thus further training offers no significant benefits. By raising this flag, we can halt the training process, thereby conserving the remaining privacy budget and boosting the privacy constraint.

The *dynamic clipping threshold* approach leverages on changes in the outputs of the sanitization layer to perform informed threshold adjustments. Since the weights of any given layer influence its output, a general increase or decrease in these weights determines a similar change in the layer’s outputs, as we discuss in Section 5.2. Our approach inspects the ℓ_2 -norm of the weights in the sanitization layer after each epoch and dynamically adjusts the element-wise clipping threshold C , aligning the clipping threshold with the changing dynamics of the layer’s outputs throughout training.

Sections 5.1 and 5.2 discuss in details each of these adaptive approaches.

5.1 Early Stopping Condition

Figure 4 reinforces our motivation for devising an early stopping condition. Assuming the noise magnitude does not change over time (which is the case for our baseline approach from Section 4) privacy budget consumption increases linearly with the number of epochs. However, after a certain point, the accuracy no longer improves significantly. For instance, if one chooses to activate an early stopping flag at epoch 40, the accuracy obtained is roughly 55% with a privacy budget of 0.8, whereas continuing further until epoch 100 consumes an additional 0.3 privacy budget (an increase of close to 40%) while the accuracy improves by only 5%.

Next, we introduce our strategy for raising early stopping flags. When all original outputs of a layer are clipped, resulting in each output being equal to the value C , the standard deviation and variance after clipping is 0 [27, 28]. Following Algorithm 1, after clipping, noise is added with a fixed magnitude. Denote the standard deviations of sanitization layer outputs after clipping and noise as S_1 and S_2 respectively. Let S_3 be the standard deviation of the noisy outputs. When adding two independent random variables X_1 and X_2 , each with standard deviations S_1 and S_2 respectively, the standard deviation of their sum $X_3 = X_1 + X_2$ is given by:

$$S_3^2 = S_1^2 + S_2^2, \quad (5)$$

where this relationship arises from the additivity of variances for independent variables, since the variance of a sum of independent variables equals the sum of their variances [27].

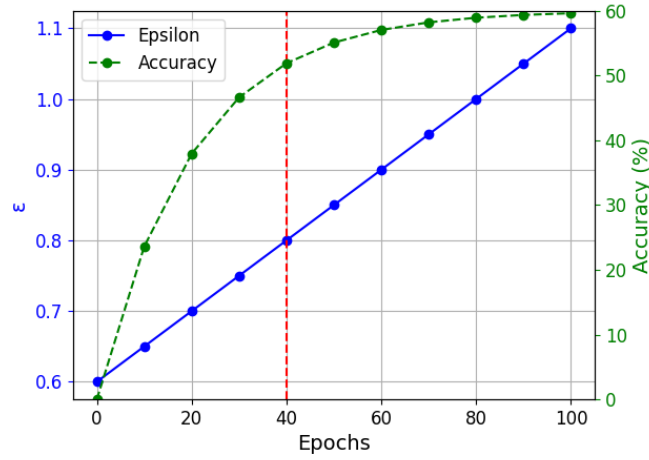


Figure 4: Motivation for Early Stopping of Training Process

Now, if all values of X_1 are clipped, then their standard deviation becomes zero: $S_1 = 0$. Substituting into the equation above, we get:

$$S_3^2 = 0 + S_2^2 = S_2^2, \quad (6)$$

and hence:

$$S_3 = S_2. \quad (7)$$

Equation (7) shows that the standard deviation of the sanitized output equals the standard deviation of the injected noise when all outputs are clipped. This results in a constant standard deviation across all outputs and batches over time. To illustrate this behavior, we plot in Figure 5 the average standard deviation across batches over time while training on the CIFAR-10 dataset. We observe that after a certain number of epochs, the value gets to be almost constant as a result of having all original outputs exceeding the clipping threshold.

We implement an early stop flag mechanism based on the standard deviation trend. Following noise addition at the sanitization layer output, the algorithm computes the average standard deviation for each batch, and aggregates these values to obtain an average standard deviation for each epoch. To address outliers that can be incurred in a single epoch, we consider a sliding window of len consecutive epochs, and determine the standard deviation trend within the window. The standard deviation values at the beginning and end of the window are utilized to calculate the slope γ of an imaginary line connecting these points. The closer the slope approaches zero, the stronger the indication of a constant standard deviation. The computed slope is compared to a fixed threshold T_s , and whenever the current slope falls below the threshold, a flag is raised to signal early stopping.

Figure 5 illustrates the progression of the standard deviation along with the slope lines connecting the sliding window endpoints for $len = 5$ (the results shown in the graph are for the MNIST dataset, please see Section 6 for experimental settings). As observed, flags are triggered at the terminal epochs when the slope approaches zero. The modifications to the basic algorithms required to support this behavior are marked in Algorithm 2.

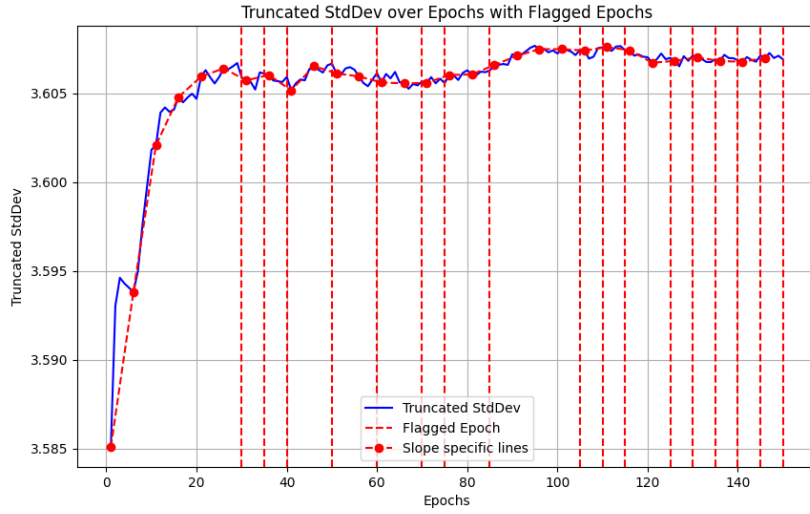


Figure 5: Computation of Early Stopping Flags

While parameters len and T_s do not directly affect the overall accuracy of the model, they significantly influence the *sensitivity* and *precision* of the proposed stopping condition based on the standard deviation of layer outputs across training epochs.

The effect of the parameters len and T_s on the behavior of the stopping condition is illustrated in Figure 6. Each subplot in the figure corresponds to a unique combination of these two parameters, showing the evolution of output standard deviation over 100 training epochs. Vertical red lines mark the epochs where the stopping condition is satisfied.

Shorter window lengths result in a highly sensitive mechanism that raises multiple early stopping flags due to transient or local fluctuations in output variability. This behavior, observable in the top row of Figure 6, indicates that small window sizes tend to overreact to noise, thus reducing the precision of the stopping decision.

Conversely, longer windows yield smoother slope estimates and fewer stopping events, improving robustness and ensuring that flags are raised only during genuine periods of output stabilization. Similarly, smaller slope thresholds enforce stricter convergence criteria and delay flagging, while larger thresholds increase sensitivity but may introduce false positives by reacting to temporary oscillations.

5.2 Dynamic Clipping Threshold

The output of the sanitization layer is computed as the weighted sum of the input values plus a bias term, followed by the application of an activation function. Let $\mathbf{x} \in \mathbb{R}^v$ be the input vector to the layer, $\mathbf{W} \in \mathbb{R}^{n \times v}$ the weight matrix, and $\mathbf{b} \in \mathbb{R}^n$ the bias vector. The output of the layer, $\mathbf{y} \in \mathbb{R}^n$, is given by:

$$\mathbf{y} = \phi(\mathbf{W}\mathbf{x} + \mathbf{b})$$

where ϕ represents the activation function applied element-wise [29].

The ℓ_2 norm of the weight matrix \mathbf{W} is a measure of the magnitude of the weights in the layer, and is given by:

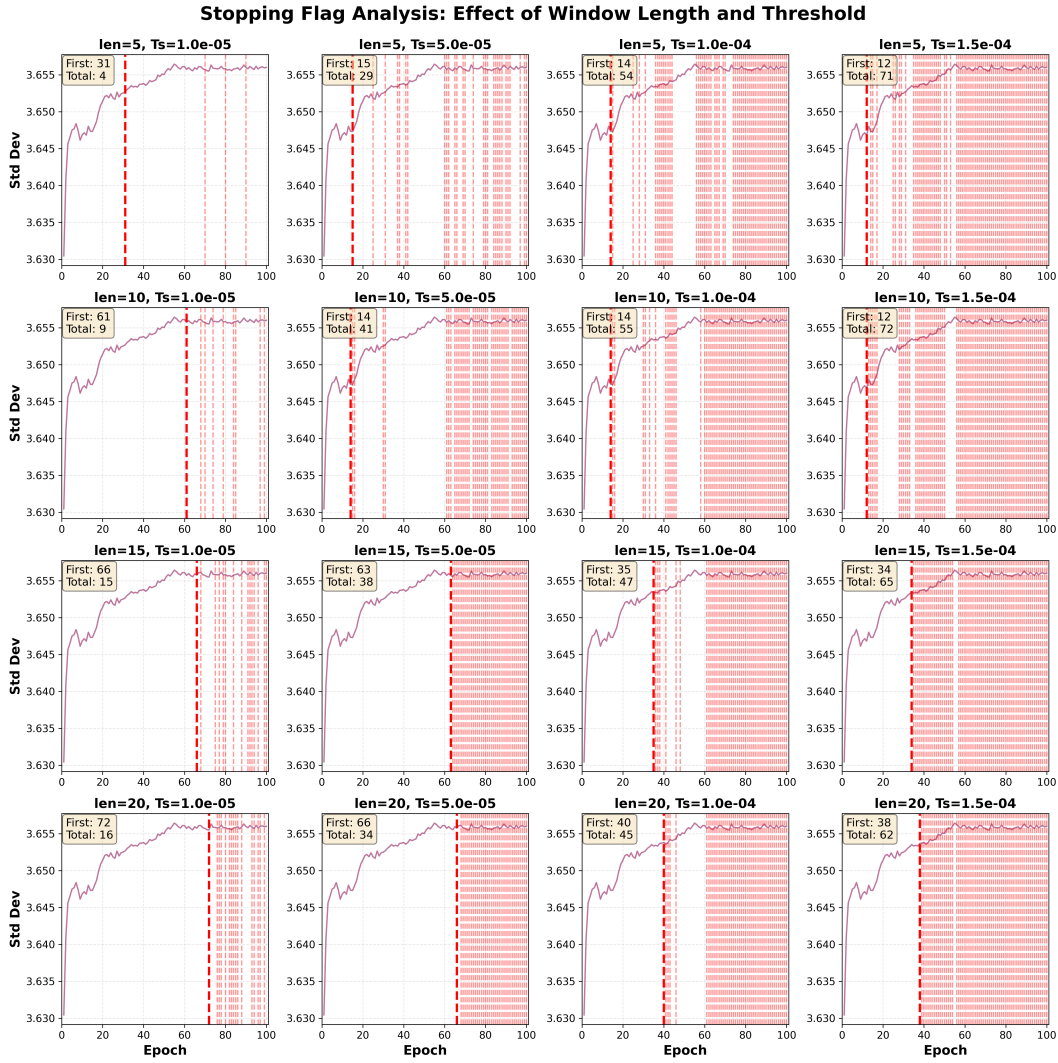


Figure 6: Effect of window length (len) and slope threshold (T_s) on stopping flags.

$$\|\mathbf{W}\|_2 = \sqrt{\sum_{i=1}^u \sum_{j=1}^v W_{ij}^2}$$

An increase in the l_2 norm $\|\mathbf{W}\|_2$ suggests that the magnitude of the weights is increasing. Since the output y is a function of the product $\mathbf{W}\mathbf{x}$, an increase in the weight norms implies a potential increase in the magnitude of the output values, assuming the input vector \mathbf{x} remains relatively constant.

If the l_2 norm of the weight matrix \mathbf{W} increases, the product $\mathbf{W}\mathbf{x}$ is likely to yield larger values, which subsequently affects the output y . Specifically, if $\|\mathbf{W}\|_2$ increases, then:

$$\|\mathbf{W}\mathbf{x}\|_2 \approx \|\mathbf{W}\|_2 \|\mathbf{x}\|_2$$

Given that $y = \phi(\mathbf{W}\mathbf{x} + \mathbf{b})$, larger values of $\|\mathbf{W}\|_2$ lead to larger inputs to the activation

Algorithm 2 Training with Early Stopping Flags

Require: Sampling ratio q , Dropout ratio p
 Sanitization layer width n , Clipping threshold C , Noise multiplier σ , Epoch counter k ,
 Window length len

Require: Training set X
 Sample X into mini-batches B_i for $i = 1, 2, \dots, m$ (number of batches)

for batch B_i **do**
 Feed forward the batch till the target layer output.
 $S_i \leftarrow 0$ Initialize average standard deviation for batch i as 0
for each output x in B_i **do**
 $x \leftarrow \text{clip}(x, C)$
 $x \leftarrow x + \sim \mathcal{N}(0, C^2 \sigma^2 n(1-p))$
 $S_x \leftarrow$ standard deviation of x
end for
 $S_i \leftarrow \text{Average}(S_x)$
 mask \leftarrow random binary mask with probability p
 $B_i \leftarrow B_i \odot \text{mask}$ {Element-wise multiplication for applying Dropout}
 Propagate noisy B_i forward in the network
end for
 $s_{epoch} \leftarrow \text{Average}(S_i)$ {Avg stddev for current epoch j }
 $k \leftarrow k + 1$
if $k = len$ **then**
 $\gamma \leftarrow \text{slope}(s_{epoch}, s_{epoch-k})$
if $\gamma \leq T_s$ **then**
 RAISE STOPPING FLAG
 $k \leftarrow 0$
end if
end if

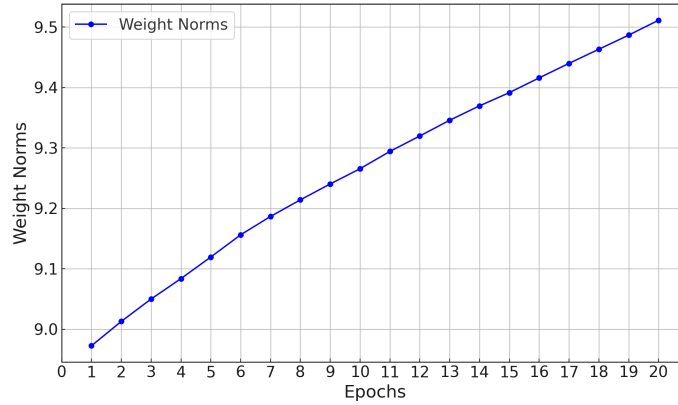
function ϕ , thereby potentially increasing the output values. During training, the weights in \mathbf{W} are adjusted to minimize the loss function. These weights can vary in both positive and negative directions, leading to changes in the ℓ_2 -norm of the weight matrix $\|\mathbf{W}\|_2$. Since the weights control the outputs of the layer, an increase in the ℓ_2 -norm of \mathbf{W} may reflect a corresponding increase in the values of the layer's outputs. Larger weight magnitudes can lead to larger activation function values, thus monitoring the ℓ_2 -norm of the weights is a good proxy for understanding how the outputs of the layer evolve during training. Figure 7 illustrates how $\|\mathbf{W}\|_2$ changes during training on the CIFAR-10 dataset (for detailed experimental settings please see Section 6).

Since in our approach we perform element-wise clipping to the sanitization layer outputs, an increase in $\|\mathbf{W}\|_2$ indicates that all elements are likely to be clipped, if the clipping threshold C is kept constant. As a result, further learning is impeded. To mitigate this, we dynamically adapt the clipping threshold as a function of the ℓ_2 norm of the weights:

$$C = t \times \|\mathbf{W}\|_2$$

where t is a constant. The pseudocode in Algorithm 3 summarizes the clipping threshold adjustment process.

The constant t controls how reactive the adaptation of C is to changes in $\|\mathbf{W}\|_2$. Figure 8

Figure 7: Evolution of ℓ_2 -norm of \mathbf{W} during training**Algorithm 3** Adapting Clipping Threshold Based on ℓ_2 -Norm of Weights**Input:** W : Initial weights**Input:** t : Constant for scaling the clipping threshold**Input:** E : Total number of epochs**Output:** Trained model with adapted clipping threshold**for** $epoch = 1$ to E **do**

Update weights using private training process

following two steps are performed on sanitized data (post-processing)

$$\|W\|_2 \leftarrow \sqrt{\sum_{i=1}^v W_i^2}$$

$$C \leftarrow t \times \|W\|_2$$

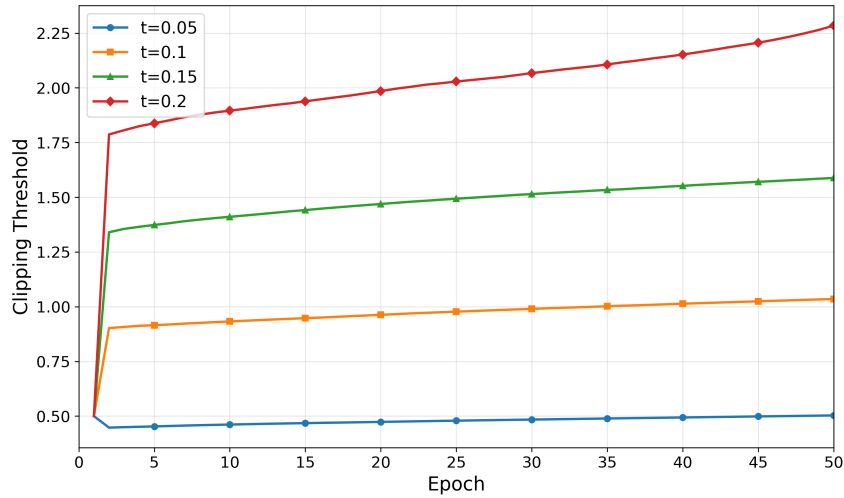
 Apply the clipping threshold C during next epoch's training**end for**

shows the effect of different values of t on the C values. Larger t values lead to faster C threshold changes, which prevents the situation where a large proportion of the outputs are clipped. However, more noise is injected to preserve DP, as the C value is part of the noise magnitude as discussed in Section 4.2.

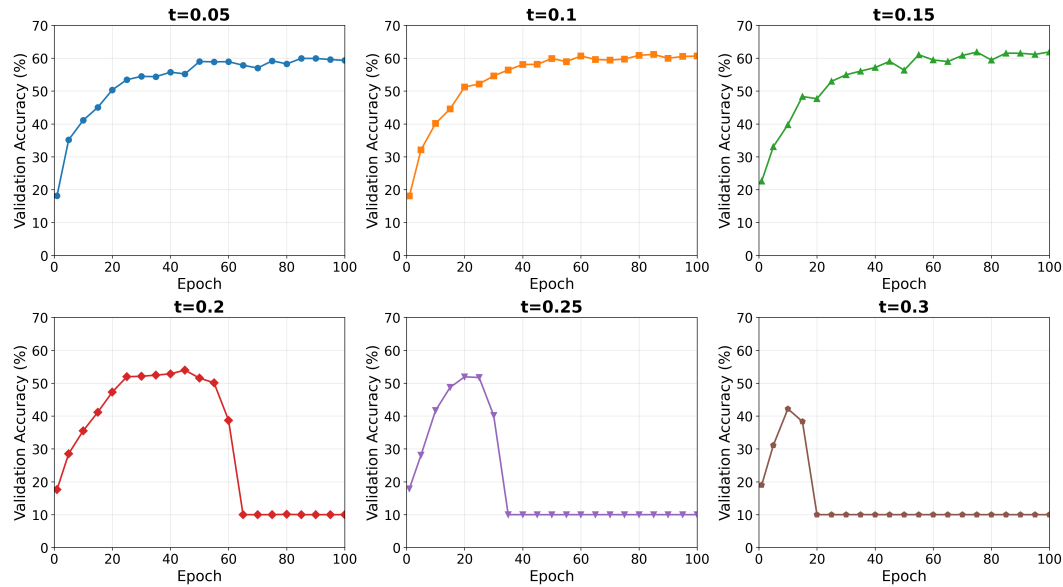
A clear trade-off emerges when setting the value of t . Increasing t results in larger values for C , which in turn leads to a more aggressive increase in the noise magnitude. This can negatively affect accuracy by introducing excessive noise into the training process. Conversely, smaller values of t reduce the noise magnitude but also increase the risk of a higher proportion of output elements being clipped. This may hinder the network's ability to capture certain patterns in the data.

Figure 9 illustrates the model's performance across training epochs for varying values of t , using the CNN architecture described in Section 6. The results indicate that moderate increases in t initially lead to improved performance. However, as t becomes too large, the model experiences higher levels of noise, which induces overfitting and hinders effective learning. Consequently, after a few initial epochs, models trained with excessively large t values fail to generalize and exhibit a significant drop in performance. Therefore, balancing the value of t is crucial to ensure the noise injection is sufficient to sanitize the model without compromising accuracy, while simultaneously limiting excessive clipping.

Privacy Discussion. According to the properties of DP-compliant iterative training [9], the computation invariant states that each iteration ends with a sanitized output, hence the

Figure 8: Influence of Constant t on Clipping Thresholds

model parameters are safe to disclose after each step. Specifically, the process starts with a random set of model parameters, hence data-independent and private. At each iteration, the effect of the input data is clipped and noised using either a gradient-based (in the case of conventional DP-SGD) or a sanitization-layer based (our case) approach. Our adaptive threshold algorithm computes the new clipping threshold to be used in iteration $(i + 1)$ based on the constant t and the model weights computed after step i , which are already sanitized. Hence, the computation qualifies as post-processing [2] according to DP, and thus is DP-compliant without requiring additional privacy budget consumption.

Figure 9: Influence of Constant t on model performance during training

Adaptive clipping and early stopping. Earlier in Section 5.1 we introduced an early stopping condition which worked under the assumption that the clipping threshold C is con-

stant. Specifically, the standard deviation of the added Gaussian noise was given by:

$$S_2 = \sigma \cdot \sqrt{n} \cdot C \quad (8)$$

By substituting this expression into Equation (7), it resulted that:

$$S_3 = \sigma \cdot \sqrt{n} \cdot C \quad (9)$$

However, when C is dynamically adapted, S_3 no longer remains constant, so a new technique for early stopping must be derived.

Note that, if we replace $\sigma \cdot \sqrt{n}$ by a constant μ , we obtain:

$$S_3 = \mu \cdot C \quad (10)$$

Since both σ and n are constant, μ is also constant. As a result, S_3 becomes a linear function of C when all output elements of the sanitization layer are clipped. This means that as C dynamically adapts, the standard deviation S_3 will increase or decrease linearly with C . Our new objective is to devise a new stopping condition that relies on detecting deviation from linear trends in the standard deviation S_3 . Same as in the non-adaptive case, we employ a sliding window of size len epochs. For each window, we fit the standard deviations across the len epochs to a line, and the slope of this line provides an indication of the rate of change in S_3 .

Once the slope is computed for each window, we compare the slopes of consecutive windows. If the difference between the slopes of two consecutive windows falls within a predefined small tolerance parameter τ , we conclude that a linear trend has been detected. The tolerance parameter τ ensures that minor fluctuations in the slopes do not trigger false positives, allowing the model to adapt more reliably to the changing values of S_3 .

This approach allows us to track the behavior of the standard deviation S_3 over time and trigger the stopping flag when a stable linear trend is observed, preventing unnecessary consumption of privacy budget under excessive output clipping. The overlapping windows help smooth out short-term variations, ensuring that the stopping condition is robust and reliable. Algorithm 4 shows the pseudocode for early stopping with adaptive clipping threshold.

Algorithm 4 Detecting Linear Trend in Standard Deviation

Input: S_3 : List of standard deviation values over epochs
Input: len : Window size (e.g., 5 epochs)
Input: τ : Tolerance parameter for slope comparison
Input: E : Total number of epochs
Output: $stop_flag$ (True or False)

Initialize $stop_flag = False$
Initialize $slopes = []$
for $i = 1$ to $E - len + 1$ **do**
 $current_window = S_3[i : i + len - 1]$
 Fit a line to $current_window$
 Compute the slope of the fitted line and store in $slopes[i]$
end for
for $j = 2$ to $length(slopes)$ **do**
 if $|slopes[j] - slopes[j - 1]| \leq \tau$ **then**
 $stop_flag = True$
 else
 $stop_flag = False$
 end if
end for
Return $stop_flag$

6 Experimental Evaluation

6.1 Experimental Settings

We evaluated our proposed approach against several prominent benchmarks [13, 22, 11, 10, 23]. Following established practices in the literature, we conducted experiments on two widely used datasets—MNIST and CIFAR-10—using two representative neural network architectures: a Convolutional Neural Network (CNN) and a Residual Network (ResNet-18). Below, we summarize the design of each model.

The CNN is a generic and configurable architecture commonly applied to image classification tasks on MNIST and CIFAR-10. Our implementation consists of three convolutional layers and three max-pooling layers, followed by four fully connected layers forming a funnel-shaped structure (as described in Section 4.3), ending with an output layer. As discussed earlier, DP noise is added to the penultimate layer, which acts as the sanitization layer. During training, a dropout layer is placed before this sanitization layer to mitigate overfitting and improve robustness to noise. The dropout ratio p is treated as a hyperparameter that balances the trade-off between noise scaling (via effective sensitivity reduction) and generalization performance.

The ResNet-18 architecture [30], composed of convolutional and residual blocks, is a deep residual network widely adopted for image classification tasks. In this model, the sanitization layer corresponds to the output of the Global Average Pooling (GAP) layer—a 1D tensor with 512 elements. These features capture high-level semantic information, such as patterns or textures extracted from the image, and serve as the input to the final classifica-

tion layer.

In the context of image classification, the 512-dimensional tensor in ResNet-18 serves as a compact, high-level representation of the input image. Each element captures discriminative features that contribute to distinguishing between classes, effectively summarizing the most relevant visual patterns identified by the network. In our experiments, DP noise is applied directly to this 512-dimensional output tensor, with $n = 512$.

6.2 Static Approach Evaluation

We begin by evaluating the privacy protection strength of our approach, which follows the (ϵ, δ) -differential privacy model with budget accounting handled via the Moments Accountant method. This technique takes as input the noise multiplier (σ), dataset size, and number of training epochs, and computes the resulting privacy loss ϵ . We explored a range of noise multiplier values from 1 to 5 and considered multiple settings for δ (commonly set as the inverse of the dataset size).

For example, on the MNIST dataset with $\sigma = 1$ and $\delta = 10^{-5}$, our method incurred a privacy cost of $\epsilon = 1.0$. Figures 10a and 10b show the privacy budget consumption on MNIST and CIFAR-10 using the CNN architecture after 100 epochs of training. In most configurations, our method requires an ϵ value below 0.5—substantially better than several benchmark methods, some of which consume budgets as high as 1 or even 10. The highest privacy cost in our setup was $\epsilon = 1.6$, occurring under minimal noise and large δ .

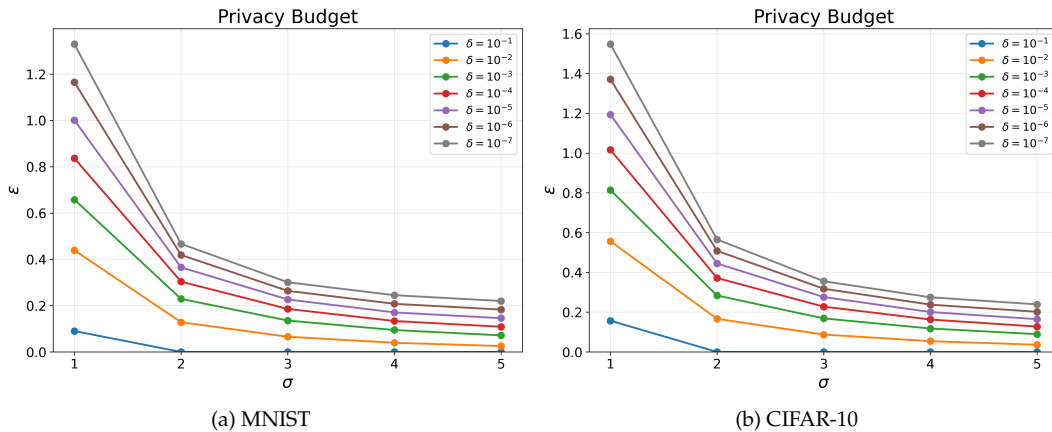
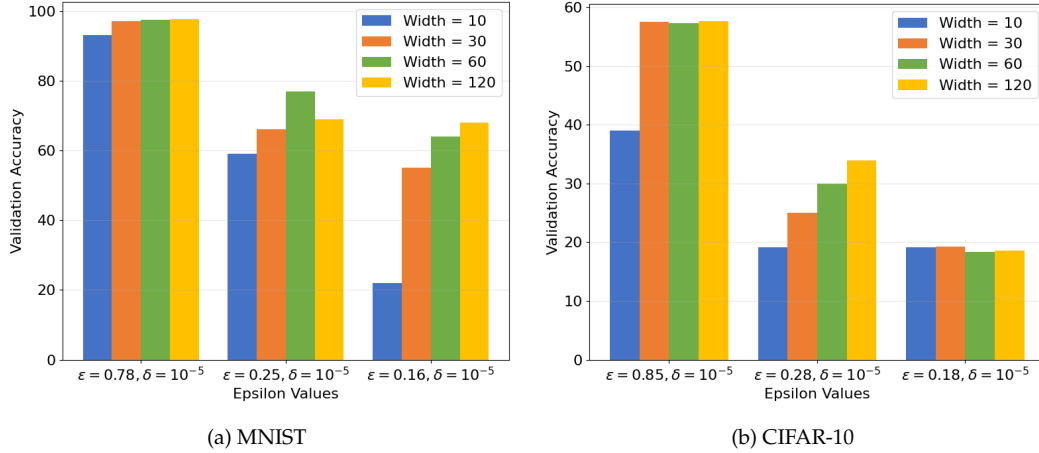


Figure 10: Privacy Budget Consumption for Proposed Approach

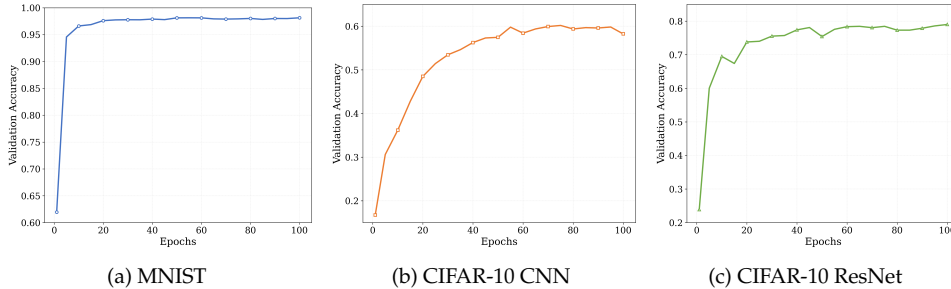
We also evaluated validation accuracy under varying sanitization layer widths. Figures 11a and 11b present the accuracy results for MNIST and CIFAR-10 on the CNN architecture. As discussed in Section 4.3, increasing the width n of the sanitization layer improves feature retention and thus accuracy, up to a point. However, since sensitivity scales with \sqrt{n} under element-wise clipping, excessive increases in n lead to larger noise magnitudes, ultimately degrading accuracy. Our results confirm this trade-off: accuracy improves as n increases from 10 to 60, but deteriorates beyond that. The optimal range was found to be $n = 40$ – 60 for both datasets. Our approach achieved peak validation accuracies of 98% on MNIST and nearly 60% on CIFAR-10—substantially outperforming conventional DP-SGD. Benchmark comparisons are provided in Tables 2 and 3.

We also investigated training convergence for both CNN and ResNet architectures. As



(a) MNIST (b) CIFAR-10
Figure 11: Accuracy with Varying Sanitization Layer Width

shown in Figure 12, both models exhibit stable convergence behavior. For ResNet-18 on CIFAR-10, we set $\sigma = 1$ and clipping threshold $C = 0.5$, using a batch size of 250. Under this configuration, the model reached a validation accuracy of 78.37% while satisfying ($\epsilon = 4, \delta = 10^{-5}$) after 82 epochs, and 79% with ($\epsilon = 4.46, \delta = 10^{-5}$) after 100 epochs. Full results are shown in Table 4.



(a) MNIST (b) CIFAR-10 CNN (c) CIFAR-10 ResNet
Figure 12: Convergence on CNN and ResNet-18 Architectures

Comparison with Benchmarks. Tables 2, 3, and 4 compare our results with leading benchmark approaches across both datasets and architectures.

Conventional DP-SGD [4] underperforms relative to our approach, lagging by approximately 18% on MNIST and 15% on CIFAR-10. Chen et al. [13], which introduced adaptive privacy budget allocation across epochs, achieved 90% on MNIST and 45% on CIFAR-10.

With privacy guarantees of ($\epsilon = 1, \delta = 10^{-5}$), the DPNASNET approach [22] achieved 97.22% accuracy on MNIST and 52.95% on CIFAR-10. Under identical privacy constraints, our method outperforms DPNASNET by approximately 1% on MNIST and 10% on CIFAR-10. A recent adaptive privacy-preserving framework [11] mitigates the impact of noise by decaying the noise magnitude over time during training. At ($\epsilon = 1.19, \delta = 10^{-5}$), it reached 97.7% accuracy on MNIST.

The study in [20] analyzed the impact of different activation functions within DP-SGD, finding that bounded ReLU and \tanh functions yield the best outcomes, with 96.02% ac-

curacy on MNIST and 44.42% on CIFAR-10 under $\epsilon = 2.0$.

We further evaluated our method on the ResNet-18 architecture and compared results with recent benchmarks. In [23], ResNet-18 trained with DP-SGD on CIFAR-10 achieved 59.8% accuracy, while their custom SmoothNets model reached 73.5%, though both models required a high privacy budget of $\epsilon = 7$. Our technique outperforms these models while maintaining significantly tighter privacy guarantees.

Finally, the ModelMix framework [10] addressed DP-SGD limitations through iterative perturbation and random aggregation of intermediate models. When tested on CIFAR-10 with ResNet-20, it reached 79.1% accuracy at ($\epsilon = 6.1, \delta = 10^{-5}$). Our method achieves nearly equivalent accuracy with substantially stronger privacy guarantees.

6.3 Adaptive Clipping Evaluation

To evaluate the effect of adaptive clipping threshold C on training accuracy, we perform experiments on the more challenging CIFAR-10 dataset, with both CNN and ResNet-18 architectures. First, we varied the sanitization layer width and investigated its effect on the model performance for different privacy constraints, as illustrated in Figure 13. The proposed adaptive clipping approach exhibits a similar trend as its non-adaptive counterpart, namely, increasing the sanitization layer width initially results in higher accuracy, but after a certain point, the increase in sensitivity caused by the growing layer width leads to an accuracy drop.

The adaptive approach clearly outperforms the static one for the entire parameter value range, but the gap closes when the width of the sanitization layer grows very large. This fact is explained as follows: when sanitization layer width n increases, the sensitivity grows as \sqrt{n} (please see Section 4.3 for details). In the case of the adaptive approach, which increases the clipping threshold to follow the value distribution, the increased threshold coupled with higher sensitivity leads to an increase in noise magnitude, in order to satisfy the privacy constraint. The static approach is subject to the same increase in sensitivity, but the C threshold value does not increase, so the magnitude of the noise added stays lower.

Furthermore, increasing the sanitization layer also increases the ℓ_2 -norm of the weights vector, leading to an increase in the adaptive clipping threshold value, which is computed as:

$$C = t \times \|\mathbf{W}\|_2$$

This also leads to amplification of the noise magnitude. This effect was particularly pronounced when the sanitization layer is increased to 100 or higher (not shown on the graph), as nearly all values of ϵ led to the accuracy dropping to a constant 10%. This emphasizes the critical balance between the noise magnitude and the width of the sanitization layer. Simply increasing the layer width does not always improve performance, as excessive width can cause the noise to dominate the results, leading to a significant drop in accuracy.

In our next experiment, we evaluate the effect of the clipping constant t . We consider two distinct number of training epochs, namely 50 and 100. The results in Figure 14 show that as long as t is within a certain range it improves model performance, as it leads to a suitable selection of C values. However, excessive values of t lead to a sharp drop in accuracy, as low as 10%. This is explained by the theoretical dependency between t and C captured in Figure 8, where an increase in t leads to larger clipping threshold values, which in turn cause the noise magnitude to increase dramatically. Figure 14 shows that a value range of t from 0.05 to 0.1 increases accuracy. With $t = 0.2$ the amount of noise being added is

almost double that of $t = 0.1$ in all training stages, producing accuracy of 51.5% at epoch 50. Additionally, the adaptation causes an increase in the noise magnitude as the training goes on, triggering a deterioration in the performance with accuracy dropping sharply at epoch 100 (discussed in section 5.2).

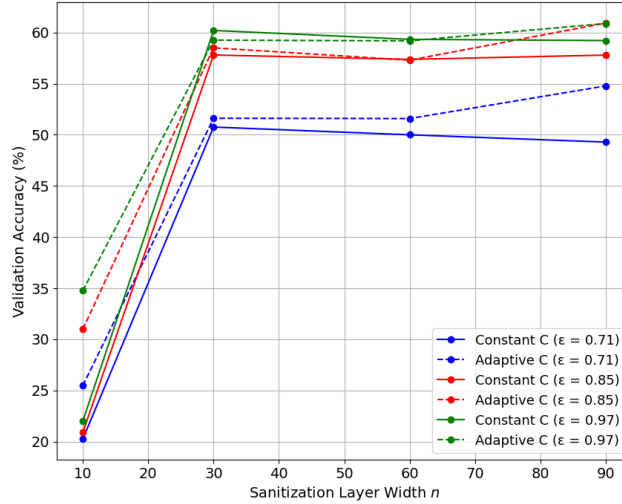


Figure 13: Adaptive Clipping Threshold Approach: Varying Sanitization Layer Width

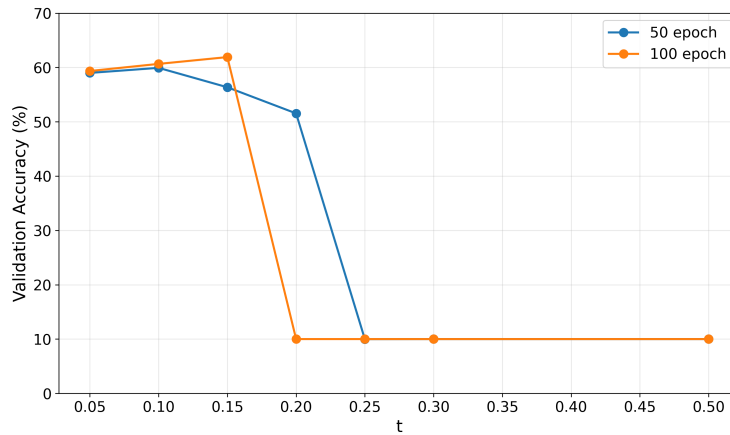


Figure 14: Adaptive Clipping Threshold Approach: Varying Clipping Constant t

6.4 Early Stopping Evaluation

Early Stopping with Non-Adaptive Threshold C. Using the same setup used in Section 6.2, we ran experiments on the CIFAR-10 dataset using the early stopping algorithm from Section 5.1. Specifically, we measure the model accuracy when stopping flags are raised, and we also determine the privacy budget savings as a result of early stopping.

We experimented with both the basic CNN and RESNET-18 architectures. Figure 15 illustrates the convergence of the training process on the RESNET-18 architecture, whereas

Tables 3 and 4 summarize the accuracy results obtained for the CNN and RESNET-18 architectures, respectively.

While training the basic CNN architecture, we set window length $len = 10$ and stopping threshold $T_s = 1.2e^{-4}$. The first flag was reported at epoch 50 with 57.38% testing accuracy guaranteeing $(\epsilon = 0.85, \delta = 10^{-5}) - DP$. This result achieves savings of 0.15 of the privacy budget compared with the conventional approach that does not employ early stopping. The recorded drop in accuracy due to early stopping was only 2.62%. For the RESNET-18 case, we set $len = 10$ and $T_s = 1.5e^{-5}$, which resulted in the stopping flag being raised at epoch 30 with 76.80% testing accuracy guaranteeing $(\epsilon = 2.34, \delta = 10^{-5}) - DP$. This achieves savings of 1.66 in privacy compared to the base case without stopping, with an only 1.57% drop in accuracy.

Figure 15 shows the slight difference in convergence between the two cases (with and without early stopping). We observe that after 30 epochs, the adaptive version is having higher accuracy improvement rate resulting from the fewer clipped elements compared to the original approach. After the first 30 epochs both start to have high convergence rate but the adaptive version's rate is slightly slower, having a more pronounced tendency for improvement than the basic approach. For the RESNET-18 architecture ($t = 0.02$), the early stopping approach reached an accuracy of 76.8% with privacy guarantees of $(\epsilon = 2.34, \delta = 10^{-5}) - DP$.

Early Stopping with Adaptive Threshold C In this experiment, we test the revised stopping condition formulation that supports adaptive thresholds (introduced in Section 5.2). We set the tolerance parameter τ to $\tau = 2 \times 10^{-5}$ for the CNN architecture and $\tau = 3 \times 10^{-5}$ for the RESNET-18 architecture with $len = 5$. For the CNN architecture, the stopping flag was first raised at epoch 57. At this point, the model achieved a validation accuracy of 60.33%, satisfying differential privacy with $(\epsilon = 0.88, \delta = 10^{-5}) - DP$. Notably, this method resulted in only a 0.7% decrease in accuracy compared to training without early stopping, while saving 0.216 in privacy budget. The accuracy obtained matched that of the case without early stopping.

For the RESNET-18 architecture, the stopping flag was raised at epoch 51, with a reported validation accuracy of 78.9%, satisfying $(\epsilon = 3.09, \delta = 10^{-5}) - DP$. The accuracy was just 0.6% lower than the adaptive training without early stopping, but the method saved 1.35 in privacy budget.

Table 2: Comparison with Benchmarks on CNN Architecture - MNIST

| Dataset | MNIST | |
|--------------------|---------------------------------------|--------------|
| | Study | Accuracy (%) |
| Chen et. al. [13] | ADAPTIVE | 90% |
| DPNASNet [22] | $(1, 10^{-5}) - DP$ | 97.22% |
| Ayoub et. al. [20] | $(1.43, 10^{-5}) - DP$ | 96.02% |
| DP-SGD | $(1, 10^{-5}) - DP$ | 80% |
| Our Study | $(1, 10^{-5}) - DP$ | 98% |

7 Conclusion

We proposed a novel approach for differentially private training of neural networks in settings where the input features are private but the labels are public. Our method intro-

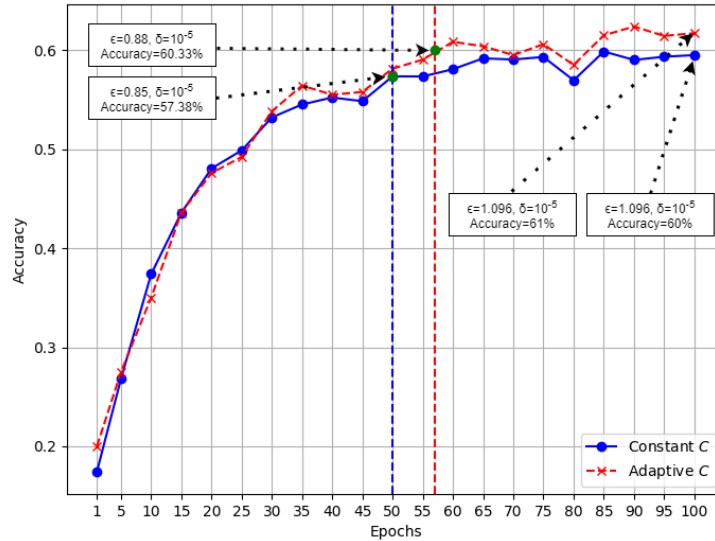


Figure 15: Effect of Early Stopping Condition with Fixed and Adaptive C

Table 3: Comparison with Benchmarks on CNN Architecture - CIFAR-10

| Dataset | CIFAR-10 | |
|--|---|---------------|
| | (ϵ, δ) -DP | Accuracy (%) |
| Chen et. al. [13] | ADAPTIVE | 45% |
| DPNASNet [22] | $(1, 10^{-5})$ -DP | 52.95% |
| Ayoub et. al. [20] | $(2.20, 10^{-5})$ -DP | 44.42% |
| DP-SGD | $(1.096, 10^{-5})$ -DP | 45% |
| Our Study | $(1.096, 10^{-5})$-DP | 60% |
| Our Study + Stopping | $(0.85, 10^{-5})$-DP | 57.38% |
| Our Study + Adaptive C | $(1.096, 10^{-5})$-DP | 61% |
| Our Study + Adaptive C + Stopping | $(0.88, 10^{-5})$-DP | 60.33% |

duces a more flexible alternative to conventional DP-SGD by rethinking where and how to inject differential privacy into the training process. Instead of applying ℓ_2 -norm clipping and adding noise at the gradient level, we perform element-wise clipping and inject noise at a dedicated sanitization layer within the network. This design choice allows for better insulation against the impact of outliers and enables more precise control over the privacy-utility trade-off. We also explored key architectural modifications that further enhance model accuracy, demonstrating that careful placement of noise and thoughtful network design can yield substantial improvements in performance under strong privacy guarantees. Additionally, we investigated dynamic approaches for adjusting clipping thresholds along with private strategies for raising early stopping flags of the training process when

Table 4: Comparison with Benchmarks on ResNet-18 Architecture (CIFAR-10)

| Study | (ϵ, δ) -DP | Accuracy | Model |
|--------------------------------|--|---------------|------------|
| Xiao et. al. [10] | $(6.1, 10^{-5})$ -DP | 79.1% | ResNet-20 |
| Remerscheid et. al. [23] | $(\epsilon = 7)$ -DP | 59.8% | ResNet-18 |
| Remerscheid et. al. [23] | $(\epsilon = 7)$ -DP | 73.5% | SmoothNets |
| Our Study | $(4, 10^{-5})$-DP | 78.37% | ResNet-18 |
| | $(4.46, 10^{-5})$-DP | 79% | |
| + Stopping | $(2.34, 10^{-5})$-DP | 76.80% | ResNet-18 |
| + Adaptive C | $(4.46, 10^{-5})$-DP | 79.5% | ResNet-18 |
| + Stopping + Adaptive C | $(3.09, 10^{-5})$-DP | 78.9% | ResNet-18 |

an accuracy plateau occurs. Extensive experimental results demonstrate that our approach consistently outperforms existing benchmarks across multiple datasets and neural network architectures. In future work, we plan to investigate avenues for additional accuracy gains, through a fine-tuned allocation of privacy budget across learning stages. We also plan to investigate adaptation of other parameters, such as learning rates.

References

- [1] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, 2017 IEEE Symposium on Security and Privacy (SP) (2016) 3–18.
- [2] C. Dwork, Differential privacy, in: International Colloquium on Automata, Languages and Programming (ICALP), Vol. 4052, 2006, pp. 1–12.
- [3] Y. Tian, Y. Zhang, H. Zhang, Recent advances in stochastic gradient descent in deep learning, Mathematics 11 (2023).
- [4] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proc. of ACM SIGSAC Conference on Computer and Communications Security, 2016, p. 308–318.
- [5] B. Ghazi, N. Golowich, R. Kumar, P. Manurangsi, C. Zhang, Deep learning with label differential privacy, Advances in neural information processing systems 34 (2021) 27131–27145.
- [6] I. A. Monir, G. Ghinita, Differentially-private neural network training with private features and public labels, in: Big Data Analytics and Knowledge Discovery - 26th International Conference, DaWaK 2024, Naples, Italy, August 26-28, 2024, Proceedings, Vol. 14912 of Lecture Notes in Computer Science, Springer, 2024, pp. 208–222.
- [7] I. Mironov, Rényi differential privacy, in: 2017 IEEE 30th Computer Security Foundations Symposium (CSF), 2017, pp. 263–275.
- [8] R. M. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin, P. Richtárik, SGD: General analysis and improved rates, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Intl Conf on Machine Learning (ICML), Vol. 97, 2019, pp. 5200–5209.
- [9] M. Gong, Y. Xie, K. Pan, K. Feng, A. K. Qin, A survey on differentially private machine learning, IEEE computational intelligence magazine (2020) 49–64.
- [10] H. Xiao, J. Wan, S. Devadas, Differentially private deep learning with modelmix, ArXiv abs/2210.03843 (2022).
- [11] X. Zhang, J. Ding, M. Wu, S. T. Wong, H. Van Nguyen, M. Pan, Adaptive privacy preserving deep learning algorithms for medical data, in: IEEE/CVF Winter Conf on Applications of Computer Vision, 2021, pp. 1169–1178.

- [12] A. Koskela, A. Honkela, Learning rate adaptation for differentially private learning, in: International Conference on Artificial Intelligence and Statistics, 2020, pp. 2465–2475.
- [13] C. Chen, J. Lee, Stochastic adaptive line search for differentially private optimization, in: 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 1011–1020.
- [14] X. Zhang, Z. Bu, S. Wu, M. Hong, Differentially private SGD without clipping bias: An error-feedback approach, in: The Twelfth International Conference on Learning Representations, 2024.
- [15] I. A. Monir, M. I. Fauzan, G. Ghinita, A review of adaptive techniques and data management issues in DP-SGD, *IEEE Data Eng. Bull.* 47 (2) (2024) 93–124.
- [16] K. L. van der Veen, R. Seggers, P. Bloem, G. Patrini, Three tools for practical differential privacy (2018). [arXiv:1812.02890](https://arxiv.org/abs/1812.02890).
- [17] J. He, X. Li, D. Yu, H. Zhang, J. Kulkarni, Y. T. Lee, A. Backurs, N. Yu, J. Bian, Exploring the limits of differentially private deep learning with group-wise clipping, in: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023, OpenReview.net, 2023.
- [18] A. Davody, D. I. Adelani, T. Kleinbauer, D. Klakow, On the effect of normalization layers on differentially private training of deep neural networks, [arXiv preprint arXiv:2006.10919](https://arxiv.org/abs/2006.10919) (2020).
- [19] T. N. Nguyen, P. H. Nguyen, L. M. Nguyen, M. Van Dijk, Batch clipping and adaptive layerwise clipping for differential private stochastic gradient descent, [arXiv preprint arXiv:2307.11939](https://arxiv.org/abs/2307.11939) (2023).
- [20] A. Arous, A. Guesmi, M. Hanif, I. Alouani, M. Shafique, Exploring machine learning privacy/utility trade-off from a hyperparameters lens, in: Intl. Joint Conf. on Neural Networks (IJCNN), 2023, pp. 1–10.
- [21] A. Priyanshu, R. Naidu, F. Mireshghallah, M. Malekzadeh, Poster: Efficient hyperparameter optimization for differentially private deep learning, *IEEE Technical Community on Security and Privacy* (2021).
- [22] A. Cheng, J. Wang, X. S. Zhang, Q. Chen, P. Wang, J. Cheng, Dpnas: Neural architecture search for deep learning with differential privacy, *Proc. of the AAAI Conference on Artificial Intelligence* 36 (6) (2022) 6358–6366.
- [23] N. W. Remerscheid, A. Ziller, D. Rueckert, G. Kaissis, Smoothnets: Optimizing cnn architecture design for differentially private deep learning, [arXiv preprint arXiv:2205.04095](https://arxiv.org/abs/2205.04095) (2022).
- [24] F. Boenisch, C. Mühl, A. Dziedzic, R. Rinberg, N. Papernot, Have it your way: Individualized privacy assignment for DP-SGD, *Advances in Neural Information Processing Systems* 36 (2024).
- [25] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, Ú. Erlingsson, Scalable private learning with PATE, *ICLR* 1050 (2018) 24.
- [26] X. Tang, M. Nasr, S. Mhroujif, V. Shejwalkar, L. Song, A. Houmansadr, P. Mittal, Machine learning with differentially private labels: Mechanisms and frameworks, in: *Proc. on Privacy Enhancing Technologies*, Vol. 4, 2022, pp. 332–350.
- [27] W. Mendenhall, R. J. Beaver, B. M. Beaver, *Introduction to probability and statistics*, Cengage, 2020.
- [28] Statistics Canada, *Statistics: Power from data!*
URL <https://www150.statcan.gc.ca/n1/edu/power-pouvoir/toc-tdm/5214718-eng.htm>
- [29] B. Ding, H. Qian, J. Zhou, Activation functions and their characteristics in deep neural networks, in: 2018 Chinese Control And Decision Conference (CCDC), 2018, pp. 1836–1841.
- [30] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: *IEEE Conf on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.