Spectral-DP: Differentially Private Deep Learning through Spectral Perturbation and Filtering

Ce Feng*[1], Nuo Xu*[1], Wujie Wen[1], Parv Venkitasubramaniam[1], Caiwen Ding[2]
[1]*Lehigh University,* [2]*University of Connecticut*
{*cef419, nux219, wuw219, pav309*}*@lehigh.edu, caiwen.ding@uconn.edu*

2023 IEEE Symposium on Security and Privacy (SP) | 978-1-6654-9336-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/SP46215.2023.10179457

***Abstract*—Differential privacy is a widely accepted measure of privacy in the context of deep learning algorithms, and achieving it relies on a noisy training approach known as differentially private stochastic gradient descent (DP-SGD). DP-SGD requires direct noise addition to every gradient in a dense neural network, the privacy is achieved at a significant utility cost. In this work, we present *Spectral-DP*, a new differentially private learning approach which combines gradient perturbation in the spectral domain with spectral filtering to achieve a desired privacy guarantee with a lower noise scale and thus better utility. We develop differentially private deep learning methods based on *Spectral-DP* for architectures that contain both convolution and fully connected layers. In particular, for fully connected layers, we combine a block-circulant based spatial restructuring with *Spectral-DP* to achieve better utility. Through comprehensive experiments, we study and provide guidelines to implement *Spectral-DP* deep learning on benchmark datasets. In comparison with state-of-the-art DP-SGD based approaches, *Spectral-DP* is shown to have uniformly better utility performance in both training from scratch and transfer learning settings.**

1. Introduction

Deep Learning algorithms have had tremendous success in a variety of domains in the last several years, due to their ability to extract inferences from data that aid in a variety of tasks. Deep learning, however, requires substantial training of several layers densely populated with weight vectors, which is enabled by large datasets often containing sensitive information. As a result, the learned models can be exploited by adversaries to extract the sensitive information in the training datasets. For instance, information about medical procedures can be determined using models built on hospital datasets [41]. It is therefore critical to provide strong and rigorous privacy guarantees for learning, and in particular, deep learning algorithms.

Over the last several years, different approaches [3], [21], [34], [36], [42], [51], [52] have been proposed to guarantee *Differential privacy* [14] – an accepted quantitative measure of privacy– for learning algorithms, which has been subsequently adapted specifically to deep learning algorithms. These methods invariably rely on a *noisy* training approach known as *Differentially Private Stochastic Gradient (DP-SGD)*, which while privacy preserving, often results in high utility loss. While there have been other advancements to enhance the privacy of deep learning algorithms, they supplement rather than provide an alternative to DP-SGD based noise addition. The main contribution of this work is a new approach to achieving differential privacy in deep learning, called Spectral-DP, which is an alternative to DP-SGD based approaches, and our results will show that this alternative can outperform DP-SGD based approaches.

In the context of deep learning, the fundamental DP-SGD approach, along with its many variants, requires direct noise addition to every weight in a dense neural network, which has a significant impact on utility. Consequently, the more significant improvements to DP-SGD in recent advances have either considered altering specifics of the deep learning architecture, or ”curing” datasets, rather than altering the methodology of noise addition. For instance, [34] explored the use of tempered sigmoid activations to improve the deep learning model's private-learning suitability and achievable privacy-utility tradeoffs (with noise addition through DP-SGD). Yet another approach that improves DP-SGD based methods is to derive handcrafted features (with a data independent preprocessing model) as in [44], where it is shown that for a fixed privacy level, deep learning model with handcrafted features outperforms end-to-end deep learning models.

Our approach is motivated by the knowledge that the utility loss in DP-SGD based perturbation methods is consequent to the direct gradient clipping and noise addition at the “signal” domain of the weights. As a result, the utility is highly sensitive to the noise scale and the clipping norm. Furthermore, there is a tension between using more weights to overcome the effects of noise, and the consequent overfitting that leads to lower utility. Our work here comes out of a hypothesis that although weight vectors have large dimension in the signal or time domain, given the density of the network, they can afford to be sparsely distributed, albeit in a transformed domain, a *spectral domain*. In other words, if the weights are restricted to a subspace in the spectral domain, it is possible to reduce the level of noise required for privacy without necessarily impacting utility. Our approach, referred to as Spectral-DP, is a method that performs a low-bandwidth noise addition in the *Fourier domain* of the weights, and combined with a filtering based dimensionality reduction, we demonstrate that it outperforms DP-SGD in trading utility for privacy.

*These authors contributed equally to this work.

DOI 10.1109/SP46215.2023.00171

Fourier transform is a classical transformation approach, and is a unitary and invertible transform, which allows for the learning gradients to be embedded into the spectral domain without impacting the privacy accounting. Furthermore, we note that the frequency components of the weights that have a significant impact on the model outcomes have lower dimensionality than those in the signal domain of the weights. Put another way, forcing weights to fall into a low frequency spectrum provides a way to *compress* the weight representation, and hence serves as a regularizer to prevent loss of utility through overfitting thus overcoming a weakness in existing methods. We derive our motivation from empirical studies, such as in [23], [37], [48], that demonstrate the so-called "Frequency Principle", wherein deep neural networks tend to fit functions in the low to medium frequencies during training. In accordance, we develop, test, and demonstrate in this work a Fourier transform based method to provide differentially private low bandwidth noise addition for deep learning architectures.

Specifically, we propose spectral domain based differential privacy for deep learning architectures that can include both convolutional and fully connected layers. Owing to the classical convolution theorem, convolutional layers are more amenable to computation friendly low bandwidth spectral perturbation. The direct adaptation of the spectral DP to fully connected layers, however, is not straightforward. In particular, the high density of weights in fully connected layers and lack of spatially localized features make direct adaptation of the spectral-DP approach challenging. In this regard, we propose an alternative to spectral filtering to reduce the dimensionality of the weights. Specifically, we adapt a spatial compression technique using block circulant matrices [11], [22] which we combine with our Fourier based noise addition approach to develop a compressed spectral domain differentially private training methodology, Block-Spectral DP. As our results will show, in networks with fully connected layers, Block Spectral DP outperforms DP-SGD based approaches.

The overarching contribution of our work is a viable alternative to DP-SGD for deep learning with differential privacy. In particular, we propose approaches that combine spectral noise addition with dimensionality reduction to achieve better utility for a given differential privacy guarantee. Our specific contributions are as follows:

- We address a critical challenge in achieving differential privacy in deep learning algorithms which is to reduce the noise scale to achieve better utility.
- Through theoretical analysis, we develop the spectral filtering based noise scale reduction technique, and provide the analytical reasoning for the improved utility performance of our methodologies.
- We develop differentially private deep learning algorithms based on our *Spectral-DP* approach for a general class of neural network architectures. Specifically, for convolutional layers our approach combines filtering and spectral gradient perturbation to achieve the desired noise scale reduction.
- For fully connected layers, we develop a variant of our fundamental approach, block *Spectral-DP*, where we adapt a spatial compression mechanism using block circulant matrices to the spectral gradient perturbation which further reduces the impact of differentially private noise addition on the utility.
- Through comprehensive experimental study, we provide guidelines to choose the right parameters including filtering ratio and clipping norms to achieve the best privacy utility tradeoff using Spectral-DP.
- Through several experiments on three benchmark image classification datasets, namely MNIST, CIFAR10 and CIFAR100, we demonstrate that *Spectral-DP* can outperform the state-of-the-art implementation of DP-SGD. Specifically, *Spectral-DP* incurs less than 1% accuracy drop for privacy budget as low as $(2, 10^{-5})$ for MNIST and 20% higher accuracy than DP-SGD for CIFAR10 with privacy budget $(3, 10^{-5})$ for training from scratch models. In the transfer learning setting, *Spectral-DP* achieves 94.85% for CIFAR10 and 77.52% for CIFAR100 with privacy budget $(1, 10^{-5})$. Moreover, when combined with Scatter-net based data curation, *Spectral-DP* incurs less than 1% accuracy loss with privacy budget $(3, 10^{-5})$.

The remainder of the paper is organized as follows. We provide some preliminaries in Section 2. We formulate the mathematical model and related formulations of *Spectral-DP* in Section 3. We conduct several experiments and analyze *Spectral-DP* in Section 4, and discuss the limitations of *Spectral-DP* in Section 5. In Section 6, we detail the related work to place our work in broader scientific context. Some concluding remarks are presented in Section 7.

2. Preliminary

In this section, we first provide a brief background on differential privacy. We then introduce the basics of differentially private stochastic gradient descent (DG-SGD) for privacy-preserving deep learning model training, followed by its Rényi differential privacy (RDP) based version for the tightest privacy account analysis on DP-SGD.

2.1. Differential Privacy

Differential privacy [14] is a quantitative definition of privacy, initially designed in the context of databases. Specifically, for two adjacent databases - i.e. databases that differ only on a single entry - differential privacy achieved by an algorithm $\mathcal{M}$ is formally defined as follows: Differential privacy [14] is a privacy definition that describes the privacy loss associated with application that utilizes from a database. It is defined on adjacent databases. We say two databases are adjacent if they differ only in a single entry. Hence, the different privacy is defined by

Definition 1. *A randomized algorithm $\mathcal{M}$ with domain $\mathcal{D}$ and $\mathcal{R} = Range(\mathcal{M})$ is $(\epsilon, \delta)-$differentially private if for all $S \subseteq \mathcal{R}$ and for any two adjacent sets $d, d' \in \mathcal{D}$:*

$$Pr[\mathcal{M}(d) \in S] \leqslant \exp^{\epsilon} Pr[\mathcal{M}(d') \in S] + \delta \qquad (1)$$

A prevalent technique for designing a differentially private mechanism is to add controlled noise from specific

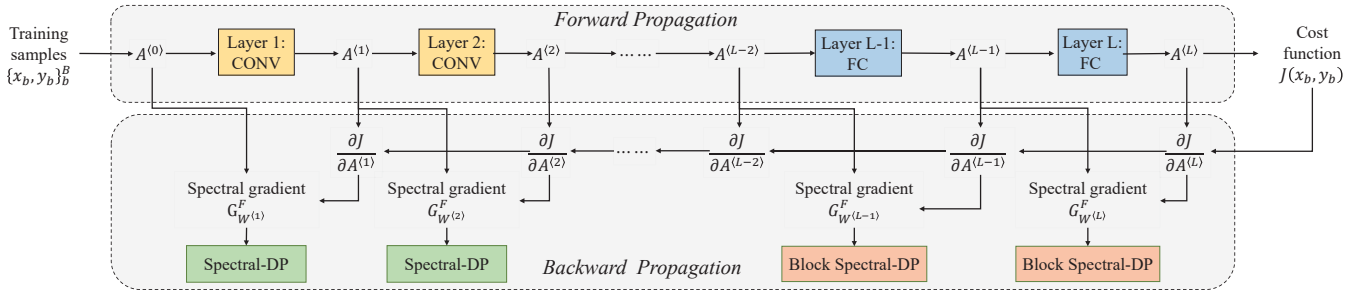


Figure 1: The backpropagation process of *Spectral-DP* based private training for deep learning models.

distribution. The noise level is controlled by the sensitivity of a function $f : \mathcal{D} \mapsto \mathbb{R}$. For instance, Gaussian mechanism takes as input the function f with sensitivity S and a parameter σ controlling the Gaussian noise. The Gaussian mechanism is formulated as noise perturbation in the output of the sequence:

$$Gauss(f, S, \sigma) \triangleq f + \mathcal{N}(0, S^2 \cdot \sigma^2) \tag{2}$$

where $\mathcal{N}(0, S^2 \cdot \sigma^2)$ is the Gaussian distribution with zero mean and variance $S^2 \cdot \sigma^2$. It is noticed that the noise scale in Gaussian mechanism is proportional to the sensitivity. And the sensitivity of f is the maximum of the absolute distance $|f(d) - f(d')|$ where d and d' are adjacent inputs. According to Theorem 3.22 in [14], for any $\epsilon, \delta \in (0, 1)$, the Gaussian mechanism achieve $(\epsilon, \delta)-$differential privacy when $\sigma = \sqrt{2\log(1.25/\delta)}/\epsilon$.

2.2. Differential Privacy in Deep Learning

In the context of deep learning, we require that the algorithm that produces the learned model is $(\epsilon, \delta)-$ differentially private with respect to the training dataset. In that regard, the most common approaches utilize the idea of differentially private stochastic gradient descent (DP-SGD) [1]. DP-SGD introduces differential privacy into deep learning by controlling the influence of the training data during the training process. Specifically in each training iteration, the per-example gradients g_i are first computed with respect to the cost function J. The gradients are clipped according to some predefined threshold C. The key to achieving privacy is to add Gaussian noise to the average of the clipped per-example gradients directly, wherein the noise level $\sigma^2 C^2$ is proportional to the L_2 sensitivity of the average gradient.

Since training a deep learning model occurs over several iterations, Gaussian noise is added at every iteration, and the overall differential privacy is computed through a composition or a privacy accountant mechanism. The best known privacy accounting is based on a modification of the differential privacy definition, known as Rényi Differential Privacy (RDP). RDP is defined in Definition 2.

Definition 2. *A randomized algorithm $\mathcal{M}$ with domain $\mathcal{D}$ and $\mathcal{R} = Range(\mathcal{M})$ is $(\epsilon, \delta)-$RDP if for any two adjacent sets $d, d' \in \mathcal{D}$:*

$$D_\alpha(\mathcal{M}(d) \| \mathcal{M}(d')) \leqslant \epsilon \tag{3}$$

where $D_\alpha(\cdot\|\cdot)$ is the Rényi divergence between two probability distributions. More detailed, it is defined by

Definition 3. *For two probability distributions P and Q defined on $\mathcal{X}$ over the same space, and let p and q denote the densities of P and Q, respectively, the Rényi divergence of order $\alpha > 1$ is given by*

$$D_\alpha(P\|Q) \triangleq \frac{1}{\alpha - 1} \log \mathop{\mathbb{E}}_{x\sim Q} (\frac{P(x)}{Q(x)})^\alpha \tag{4}$$

To analyze the composition of DP-SGD, the DP guarantee of a single iteration of DP-SGD is firstly converted into its equivalent RDP using Proposition 1 in [27], and subsequently for T training iterations, the total RDP guarantee can be obtained by Proposition 2 in [27]. Finally, the total DP guarantee can be converted back from the total RDP guarantee by Proposition 1.

Proposition 1. *(From RDP to (ϵ, δ)-DP). If f is an $(\alpha, \epsilon)-$ RDP mechanisms, it also satisfies $(\epsilon + \frac{\log 1/\delta}{\alpha - 1}, \delta)$-differential privacy for any $0 < \delta < 1$.*

Proposition 2. *For a $(\alpha, \epsilon_1) - RDP$ mechanism f and a $(\alpha, \epsilon_2) - RDP$ mechanism g, then the mechanism $f \circ g$ satisfies $(\alpha, \epsilon_1 + \epsilon_2) - RDP$.*

Although our approach is different from DP-SGD wherein noise is not added directly to the gradients, we will utilize the above propositions by drawing an equivalence to differential privacy in each iteration, and subsequently using the RDP based privacy accounting to compute the total differential privacy of our approach.

3. Approach

In this section, we formally present *Spectral-DP* in the context of deep learning. While *Spectral-DP* shares the same objective with DP-SGD which aims to perturb weight gradients during the training process, it can principally reduce the differential privacy (DP) noise, thus to significantly escalate model utility, by conducting dedicated spectral filtering in our designed DP noise addition process performed in the spectral domain, with theoretical guarantee on the privacy.

Figure 1 provides an overview of our approach. Consider a deep learning model consisting of L layers (either convolutional or fully connected). During the backward propagation process of the learning algorithm, the gradients at each layer are transformed into their spectral representation which is subsequently perturbed by Gaussian noise, and filtered, prior to transforming back to the signal (or spatial) domain. Since a convolutional operation in the spatial domain is

equivalent to multiplication in the spectral domain, **convolutional (CONV) layers** are more amenable to spectral gradient perturbation. For **fully connected (FC) layers**, we supplement this mechanism with a block-circulant matrix based weight compression and restructuring to address the high density of weights prior to spectral perturbation and filtering.

The remainder of the section is organized as follows. In Section 3.1, we present the conceptual basis of *Spectral-DP* and a theoretical analysis that provides the differential privacy guarantee of the method, and demonstrates the reduction in noise scale that enables the better utility performance of spectral perturbation and filtering. In Sections 3.2 and 3.3 we describe in detail the spectral perturbation and filtering methodology as applied to convolutional layers and fully connected layers respectively. In Section 3.4, we outline the overall training of a neural network with both kinds of layers over multiple iterations to achieve a desired guarantee of (ϵ, δ) differential privacy.

3.1. Conceptual Foundations of Spectral-DP

In this section, we present the concept and theoretical analysis of *Spectral-DP* which is the basis of the specific deep learning algorithms developed in subsequent sections.

3.1.1. *Spectral-DP* Overview. The key of *Spectral-DP* is to perturb weight gradients in the spectral or Fourier domain by taking advantage of existing primitives such as Fourier transform and the Gaussian mechanism for differential privacy. Specifically, Fourier transform (FT) is used to project data to the spectral domain (or frequency domain), and the algorithm perturbs the Fourier transform coefficients prior to filtering out a fraction of the coefficients. The approach is described in mathematical detail below.

Consider an N length sequence $\mathbf{Q} = \{Q_0, Q_1, \cdots, Q_{N-1}\}$ as an example. We denote $\{F^N\} := \{F_0, F_1, \cdots, F_{N-1}\}$ as a collection of all spectral coefficients of $\mathbf{Q}$, where each F_i is computed by:

$$F_i = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} Q_n \cdot e^{-\frac{j2\pi}{N} in}$$

The Gaussian noise addition mechanism is applied into $\{F^N\}$. Since the Gaussian noise scale is proportional to the L_2 norm of the $\{F^N\}$, we bound $\{F^N\}$ by a clipping parameter S:

$$\bar{F}^N = F^N / \max\{1, \frac{\|F^N\|_2}{S}\}$$

where $\|\cdot\|_2$ denotes the L_2 norm. The spectral coefficients are perturbed with additive Gaussian noise:

$$\tilde{F}^N = \bar{F}^N + V^N$$

where $V^N = \{V_0, V_1, \cdots, V_{N-1}\}$ is the noise vector, and each V_i is drawn from $\mathcal{N}(0, \sigma^2 S^2)$ independently. We denote this process as $Gauss(\bar{F}^N, S, \sigma)$. A key mechanism that allows *Spectral-DP* to limit the impact of noise is filtering, in other words, eliminating a fraction of the coefficients:

$$P_K(\tilde{F}_i) = \begin{cases} \tilde{F}_i & \text{if } i < K \\ 0 & \text{otherwise} \end{cases}$$

Algorithm 1 *Spectral-DP* perturbation

Require: Query $Q = \{Q_0, Q_1, \cdots, Q_{N-1}\}$, l_2 sensitivity S, noise scale σ
Output: $\tilde{Q}$
1: Compute the Fourier coefficients of Q
2: Clipping the Fourier coefficients by S
3: Noise addition: $\tilde{F}^N = Gaussian(\bar{F}^N, S, \sigma)$
4: Spectral filtering: $\hat{F}_i^K = P_K(\tilde{F}_i)$
5: Inverse Fourier transformation: $\tilde{Q_n} = \text{I-FT}(\hat{F_i}^K)$

the K/N determines the fraction of coefficients which are perturbed and allows us to reduce the overall noise scale. Our motivation is that it is sufficient to concentrate the weights in a *low bandwidth* space without compromising on utility while saving on the impact of noise. The perturbed and filtered coefficients are retransformed to the signal domain using the inverse Fourier Transform (I-FT). The overall procedure is outlined in Algorithm 1.

3.1.2. Theoretical analysis of *Spectral-DP*. In the following theorem, we determine the privacy budget of Algorithm 1, and prove that spectral perturbation achieves the desired differential privacy.

Theorem 1. *In Algorithm 1, the output $\tilde{Q_n}$ is (ϵ, δ) differentially private if we choose σ to be $\sqrt{2\log(1.25/\delta)}/\epsilon$.*

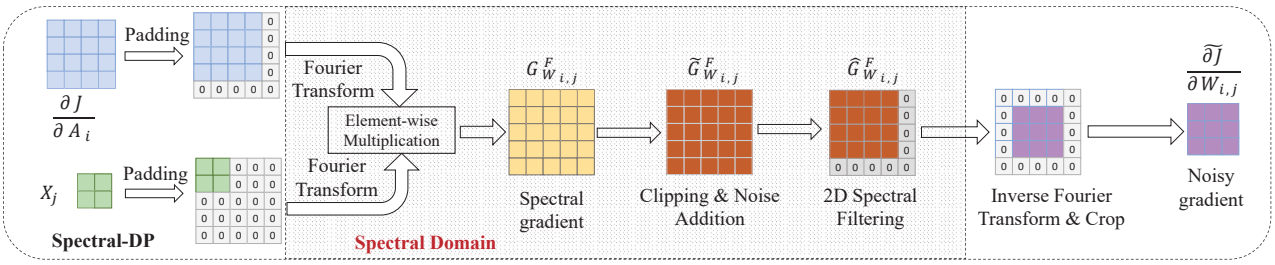
Proof. The proof relies on Theorem 3.22 in [14] and the post-processing property of DP algorithm. The detailed proof is given in Appendix A.1. □

Since both spectral filtering and inverse Fourier transformation can be treated as the post-processing steps that do not alter the DP budget, *Spectral-DP* better utilizes the privacy budget. As demonstrated in the following Proposition, the filtering operation in spectral domain leads to prominent noise scale reduction.

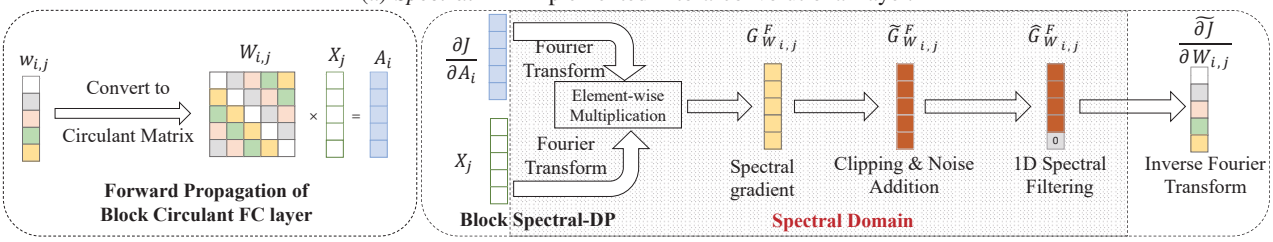
Proposition 3. *Let $V^N = \{V_0, V_1, \cdots, V_i, \cdots, V_{N-1}\}$ be a collection of noise vector in spectral domain, and each V_i is drawn from $\mathcal{N}(0, \sigma^2 S^2)$. Consider $v_n =$ I-FT$(P^K(V_i))$, then v_n follows a normal distribution $\mathcal{N}(0, \frac{K}{N}\sigma^2 S^2)$.*

Proof. The detailed proof is given in Appendix A.2. □

As Proposition 3 shows, the spectral filtering allows the reduction in the overall noise scale from $\sigma^2 S^2$ down to $\frac{K}{N} \cdot \sigma^2 S^2$, where $K < N$. Consequently, should the filtering mechanism not affect the utility, the noise reduction could significantly minimize the utility penalty incurred by differential private training. While filtering more frequency components (a smaller K) indicates more DP noise reduction thus less utility penalty by DP, the more weight distortion errors after Inverse Fourier Transform could inevitably impact the gained model utility. We define a key parameter– **filtering ratio** $\rho = (K - N)/N$, to balance the impact of these two factors and will discuss the impact and choice of this key parameter in Sections 4.2.1 and 4.2.2



(a) *Spectral-DP* implemented into a convolutional layer.



(b) *Block Spectral-DP* implemented into a block circulant fully connected layer.

Figure 2: Detailed Spectral-DP framework.

3.2. *Spectral-DP* in CONV Layer

3.2.1. Adapting *Spectral-DP* to 2D CONV. To adapt *Spectral-DP* for private deep learning, our first question would be how to perform gradient perturbations for different types of model layers using *Spectral-DP*. We first focus on the 2-dimensional (2D) convolution that dominates the operations of the convolutional layer.

According to the convolution theorem, the 2D convolution in spatial domain can be easily converted into element-wise multiplication of two matrices in the spectral domain. *Spectral-DP* is then applied into the element-wise multiplication and mainly consists of a Gaussian noise addition and a 2D spectral filtering. To demonstrate how effectively *Spectral-DP* reduces DP noises in 2D convolution, we further derive the relation between the noise scale and filtering parameter ρ in Corollary 1.

Corollary 1. *Let V^N be the collection of a noise vector $\{V_{i,j}\}$ where $i \in \{0, 1, \cdots, N-1\}$ and $j \in \{0, 1, \cdots, N-1\}$ in spectral domain, and each $V_{i,j}$ be drawn from $\mathcal{N}(0, \sigma^2 S^2)$, consider a 2D spectral filtering:*

$$P_{2D}^K = \begin{cases} V_{ij} & \text{if } i < K \text{ and } j < K \\ 0 & \text{otherwise} \end{cases}$$

and $v_{mn} = \mathcal{F}^{-1}(P_{2D}^K(V_{i,j}))$, then v_{mn} follows a normal distribution $\mathcal{N}(0, \frac{K^2}{N^2}\sigma^2 S^2)$.

Proof. The detailed proof is given in Appendix A.3. □

3.2.2. Adapting *Spectral-DP* into CONV layer. Based on the 2D spectral filtering, we then provide more implementation details of *Spectral-DP* for training convolutional layers. We consider a typical 2-dimensional convolutional layer of a deep learning model with the input vector $X \in \mathbb{R}^{H_{in} \times W_{in} \times C_{in}}$ and convolution filters $W \in \mathbb{R}^{C_{out} \times C_{in} \times d \times d}$, where C_{in} and C_{out} are the number of input and output channels, $d \times d$ is the size of a 2D convolution filter. The forward propagation process of the inference in the layer is expressed as follow (bias and activation are omitted):

$$A_i = \sum_j^{C_{in}} X_j \circledast W_{i,j}$$

where A_i denotes the $i-$th channel of the convolution output with size $H_{out} \times W_{out}$ where $H_{out} = H_{in} + d - 1$ and $W_{out} = W_{in} + d - 1$, X_j denotes the $j-$th channel of the

Algorithm 2 *Spectral-DP* in a 2D convolutional layer

Require: A CONV layer with input X, filters W, $\frac{\partial J}{\partial A}$, clipping bound S, σ, filtering ratio $\rho \in (0, 1)$

Output: $\frac{\tilde{\partial J}}{\partial W}$

• ***Stage I:*** Noise addition

Compute: $G_W^F = \{G_{W_{i,j}}^F\}$, where $i \in \{1, \cdots, C_{out}\}$, $j \in \{1, \cdots, C_{in}\}$.

Clipping Norm: $\bar{G}_W^F = G_W^F / \max(1, \frac{\|G_w^F\|_2}{S})$

Gaussian mechanism: $\tilde{G}_W^F = Gauss(\bar{G}_W^F, S, \sigma)$

• ***Stage II:*** Filter-wise pruning and inverse Fourier Transform

for $i \in 1, 2, \cdots, C_{out}$ **do**

  for $j \in 1, 2, \cdots, C_{in}$ **do**

    2D Spectral filtering: $\hat{G}_{W_{i,j}}^F \leftarrow$Zero last ρ of rows and columns in $\tilde{G}_{W_{i,j}}^F$

    Inverse Fourier transform: $\frac{\partial J}{\partial W_{i,j}} = \mathcal{F}^{-1}[[\hat{G}_{W_{i,j}}^F]_0]$

  end for

end for

$\frac{\partial J}{\partial W} = \{\frac{\tilde{\partial J}}{\partial W_{i,j}}\}$, $i \in \{1, \cdots, C_{out}\}$, $j \in \{1, \cdots, C_{in}\}$.

input, $W_{i,j}$ denotes the convolution filter that connects the $i-$th channel of the output and the $j-th$ channel of the input, and $\circledast$ denotes the 2D convolution. For each convolution filter, the backward propagation and the convolution theorem indicate that the gradient can be approximately expressed as

$$\frac{\partial J}{W_{i,j}} = \frac{\partial J}{\partial A_i} \circledast \frac{\partial A_i}{\partial W_{i,j}} = \mathcal{F}^{-1}[\mathcal{F}[\frac{\partial J}{\partial A_i}] \odot \mathcal{F}[X_j]] \quad (5)$$

where $\frac{\partial J}{W_{i,j}}$ denotes the gradient of $W_{i,j}$ w.r.t. the cost function J, $\frac{\partial J}{\partial A_i}$ denotes the gradient of A_i w.r.t. J, $\frac{\partial A_i}{\partial W_{i,j}}$ denotes the gradient of $W_{i,j}$ w.r.t. A_i, $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier Transform and inverse Fourier Transform operator, respectively, and $\odot$ denotes the element-wise multiplication. Let $G^F_{W_{i,j}} = \mathcal{F}[\frac{\partial J}{A_i}] \odot \mathcal{F}[X_j]$ be the spectral gradient of $\frac{\partial L}{W_{i,j}}$, then *Spectral-DP* can be directly applied into the spectral gradient. Computing the spectral gradient requires a complexity of $O((H_{out} * W_{out}) \log(H_{out} * W_{out}))$ where conventional convolution of computing $\frac{\partial J}{W_{i,j}}$ requires a complexity of $O(H_{out} * W_{out} * H_{in} * H_{in})$. Theoretically, the spectral gradient computation is faster than the conventional convolution if $\log(H_{out} * W_{out}) < (H_{in} * H_{in})$. We provide detailed complexity analysis in the Appendix A.7.

Figure 2(a) depicts the implementation steps when applying *Spectral-DP* to the gradient perturbation of a single 2D convolution filter. $\frac{\partial J}{A_i}$ and X_j are first padded to the same size and transformed to the spectral gradient $G^F_{W_{i,j}}$. Consequently, the noisy spectral gradient $\tilde{G}^F_{W_{i,j}}$ is obtained by applying clipping and Gaussian noise addition into $G^F_{W_{i,j}}$. The filtered spectral gradient $\hat{G}^F_{W_{i,j}}$ is then computed with a filtering ratio ρ using the 2D spectral filtering approach mentioned in Corollary 1.

The main procedure of *Spectral-DP* in 2D convolutional layer is outlined in Algorithm 2. As the 2D convolutional layer usually contains multiple filters, we denote $G^F_W = \{G^F_{W_{i,j}}\}$ as the spectral gradients of W with respect to the cost function J. As shown at the stage I of Algorithm 2, by applying Gaussian mechanism into G^F_W, the differential privacy of all parameters in the layer is guaranteed. At stage II, the spectral domain filtering is applied within each convolution filter. According to Corollary 1, *the 2D spectral filtering provides larger noise reduction than the 1D spectral filtering but leads to larger weight distortion errors.* In Section 4, we conduct comprehensive experiments to evaluate how the filtering ratio affects the utility.

3.3. Block Spectral-DP in FC Layer

In addition to fitting *Spectral-DP* into CONV layers, our next question would be how to extend it to the fully connected (FC) layers. The weight matrix in an FC layer often has a much higher dimension than CONV layers which have a weight-sharing mechanism. Furthermore, unlike CONV layers, operations in FC layers cannot directly map to multiplication in the spectral domain. To address these, our key idea is to compress and restructure the weight matrix to facilitate the adoption of *Spectral-DP* to FC layers. In this regard, the structure of a block circulant weight matrix [11], [22] is a suitable choice. Each row vector in such a matrix is the circular shift form of the previous row, and the matrix vector multiplication in time domain can be simplified as vector-vector multiplication in the spectral domain. We further name this approach as *Block Spectral-DP*. As we shall show later, *Block Spectral-DP* not only mathematically supports the spectral transformation for adding gradient perturbation, but also compresses redundant weights in FC layers without impacting the utility.

3.3.1. Block circulant based FC layer. We first introduce the definition of a block circulant matrix. Given a matrix W of size $m \times n$, it is said that W is block circulant if W can be partitioned into $p \times q$ square blocks of circulant matrix $W_{ij} \in \mathbb{R}^{d \times d}$, where d is defined as the block size (size of each sub-matrix block), $p = m \div d$, $q = n \div d$, $i \in \{1, 2, \cdots, p\}$, and $j \in \{1, 2, \cdots, q\}$. And each square block matrix $W_{i,j}$ is circulant as specified below:

$$\begin{bmatrix} w_0 & w_1 & \cdots & w_{k-1} \\ w_{k-1} & w_0 & \cdots & w_{k-2} \\ \ddots & \ddots & \ddots & \ddots \\ w_1 & w_2 & \cdots & w_0 \end{bmatrix}$$

We note the circulant matrix can be represented by a vector $w = \{w_0, w_1, \cdots, w_{k-1}\}$Consider a fully connected layer consisting of m outputs and n inputs and a block circulant weight matrix W. Assume W is partitioned into $p \times q$ blocks of circulant matrix, the forward propagation in the block circulant weight matrix based fully connected layer is given by:

$$A = WX = \begin{bmatrix} \sum_{j=1}^{q} W_{1,j} X_j \\ \sum_{j=1}^{q} W_{2,j} X_j \\ \vdots \\ \sum_{j=1}^{q} W_{p,j} X_j \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_p \end{bmatrix} \quad (6)$$

where the input X is partitioned as $X = [X_1^T, X_2^T, \cdots, X_q^T]^T$, $A_i \in \mathbb{R}^k$ is a column vector that is the respective output of $\sum_{j=1}^{q} W_{i,j} x_j$. We further assume each square block sub-matrix $W_{i,j}$ is represented by a vector $w_{i,j}$ where $w_{i,j}$ is the first row of $W_{i,j}$, then according to the *circulant convolution theorem* [15], [31], the computation of $W_{i,j} X_j$ can be expressed as $A_i = w_{i,j} * X_j = \mathcal{F}^{-1}(\mathcal{F}(w_{i,j}) \odot \mathcal{F}(X_j))$, where $*$ is the operator of circulant convolution. This process is shown in the left block of Figure 2(b).

We then consider the backward propagation training of the fully connected layer with the block circulant weight matrix. Let J denote the cost function, and A_{il} be the $l-th$ element in A_i, by the chain rule, the backward propagation process is derived as

$$\frac{\partial J}{\partial w_{i,j}} = \sum_{l=1}^{k} \frac{\partial J}{A_{i,l}} \frac{\partial A_{i,l}}{w_{i,j}} = \frac{\partial J}{A_i} \frac{\partial A_i}{\partial w_{i,j}} \quad (7)$$

We note A_i is the circular convolution of w_{ij} and x_j which indicates that $\frac{\partial A_i}{\partial w_{ij}}$ is block circulant matrix. Hence, the computation of Eq. (7) can be expressed as the "Fourier Transform $\longrightarrow$ Element-wise Multiplication $\longrightarrow$ Inverse Fourier Transform". The complexity analysis of computing

Algorithm 3 *Block Spectral-DP* in a single FC layer

Require: A block circulant weight matrix based FC layer with input X and block circulant matrix W. $\frac{\partial J}{\partial A}$, $\{w_{i,j}\}$, p, q, block size d, m, n, clipping bound S,σ filtering parameter $\rho \in (0,1)$

Output: noisy gradient $\frac{\tilde{\partial J}}{\partial W}$

• ***Stage I:*** Noise addition

Compute: $G_W^F = \{G_{W_{i,j}}^F\}$, where $i \in \{1,2,\cdots,p\}$, $j \in \{1,2,\cdots,q\}$.

Clipping Norm: $\bar{G}_W^F = G_W^F / \max(1, \frac{\|G_w^F\|_2}{S})$

Gaussian mechanism: $\tilde{G}_W^F = Gauss(\bar{G}_W^F, S, \sigma)$

• ***Stage II:*** Block-wise pruning and inverse Fourier Transform

for $i \in 1,2,\cdots,p$ **do**

  for $j \in 1,2,\cdots,q$ **do**

    1D Spectral filtering: $\hat{G}_{W_{i,j}}^F \leftarrow$Zero last ρ of coefficients in $\tilde{G}_{W_{i,j}}^F$

    Inverse Fourier transform: $\frac{\partial J}{\partial W_{i,j}} = \mathcal{F}^{-1}[\hat{G}_{W_{i,j}}^F]$

  end for

end for

$\frac{\partial J}{\partial W} = \{\frac{\tilde{\partial J}}{\partial W_{i,j}}\}$, $i \in \{1,\cdots,p\}$, $j \in \{1,\cdots,q\}$.

the gradient is provided in Appendix A.7.

3.3.2. Implementing *Block Spectral-DP* into FC layer. We now demonstrate how to implement *Block Spectral-DP* into the fully connected layer. Eq. (7), allows us to apply *Block Spectral-DP* into the spectral gradients of the parameters. Then for each square block, the spectral gradient is computed by

$$G_{w_{i,j}}^F = \mathcal{F}(\frac{\partial J}{\partial A_i}) \circ \mathcal{F}(\frac{\partial A_i}{\partial w_{i,j}}) \tag{8}$$

Algorithm 3 outlines *Block Spectral-DP* as applied to a fully connected layer. We denote G_W^F as the spectral gradient of W with respect to the cost function J. At stage I, the Gaussian mechanism is applied into the G_W^F to ensure the differential privacy guarantee of the spectral gradients. Operations at stage II such as spectral filtering operation and inverse Fourier Transform are introduced as the post-processing of the Gaussian mechanism. We note operations are applied at the sub-matrix level. Unlike the CONV layer, we note that both (spectral) filtering ratio, and (block-circulant) compression ratio can impact the privacy level and utility, which we study in Section 4.

3.4. Integrate *Spectral-DP* into a DL model

In this section, we show how to apply *Spectral-DP* to train a general deep learning model consisting of many such layers. Specifically, at each training iteration, *Spectral-DP* computes the per-sample spectral gradient G^F, bounds G^F using L_2 norm clipping, and adds noise to the spectral gradient using Gaussian mechanism. Then *Spectral-DP* applies spectral filtering to each layer. Without loss of generality,

Algorithm 4 Training algorithm of *Spectral-DP* in a deep learning model

Require: A model with L layers, model parameters W, clipping bound $\{C_l\}_l^L$, σ, filtering parameter $\rho \in (0,1)$, training samples $\{x_i, y_i\}_{i=1}^N$, batch size B, total training epochs T_e, cost function J learning rate α

Output: Model parameters after $T_e * N/B$ training iterations $\hat{W}_{T_e * N/B}$

for $t \in [T_e * N/B]$ **do**

  Sample a mini-batch of training samples $\{x_b, y_b\}_{b=1}^B$ by selecting each $\{x_i, y_i\}$ independently with probability $\frac{B}{N}$ using SGM.

  Stage I: Noise addition

  for $b \in 1,2,\cdots,B$ **do**

    Compute per-sample spectral gradient: $G^F(x_b, y_b) = \{G_{W_t^{<l>}}^F(x_b, y_b)\}_l^L$.

    for $l \in 1,2,\cdots,L$ **do**

      L_2 **norm of clipping:** $\bar{G}_{W_t^{<l>}}^F(x_b, y_b) = G_{W_t^{<l>}}^F(x_b, y_b) / \max\{1, \frac{\|G_{W_t^{<l>}}^F(x_b, y_b)\|_2}{C_l}\}$

    end for

  end for

  $G_{sum}^F = \sum_{b=1}^B \bar{G}^F(x_b, y_b)$

  Gaussian mechanism: $\hat{G}_{sum}^F = Gauss(G_{sum}^F, C, \sigma)$, where $C = \sqrt{\sum_{l=1}^L C_l^2}$

  Stage II: Pruning and Inverse Fourier Transform

  for $l \in 1,2,\cdots,L$ **do**

    Spectral filtering and Inverse Fourier Transform: $\tilde{G}_{sum}^F \leftarrow \mathcal{F}^{-1}(filtering(\hat{G}_{sum}^F))$

  end for

  Gradient descent $\hat{W}_{t+1} \leftarrow W_t - \alpha \frac{1}{B} \tilde{G}_{sum}^F$

end for

we present the detailed procedure of *Spectral-DP* learning in Algorithm 4.

At each training iteration t, the mini-batch $\{x_b, y_b\}_{b=1}^B$ is sampled using the Sampled Gaussian mechanism (SGM) [28]. In practical implementation, *Spectral-DP* clips each g_l by a different clipping norm C_l.The L_2 norm of G^F can be computed by $C = \sqrt{\sum_{l=1}^L C_l^2}$. Based on this clipping strategy, the noise scale in Gaussian mechanism is proportional to C instead of C_l–the L_2 norm of each layer's gradients. This ensures that the perturbed gradients of all parameters have the same privacy level. In our experiments, we set equal C_l and study the impact of the clipping norm in Section 4.

In Corollary 2, we leverage the RDP based privacy accountant as described in Section 2.2 to compute the overall differential privacy across T epochs.

Corollary 2. *Algorithm 4 achieves* $((T_e * N/B)\epsilon + \frac{\log(1/\delta)}{\alpha-1}, \delta)$*−DP if* $\sigma = \frac{\sqrt{2\log(1.25/\delta)}}{\epsilon'}$ *where* $\epsilon' = \epsilon + \frac{\log(1/\delta)}{\alpha-1}$.

Proof. The detailed proof is provided in Appendix A.4. □

4. Experiment

4.1. Experiment setup

Experimental Environment. We use Pytorch [35] to implement *Spectral-DP* and DP-SGD [1]. For a fair comparison, we follow the provided codes of Opacus [49] to build and train the models for DP-SGD. All experiments are conducted on a Linux PC with AMD Ryzen Threadripper Pro 3975WX 32-Core Processor, 256 GB memory and NVIDIA GeForce RTX 3090 GPU with 24 GB graphic memory.

Datasets. We evaluate the proposed *Spectral-DP* training on public image classification datasets MNIST, CIFAR10 and CIFAR100. MNIST [20] consists of 70,000 images of 28 × 28 handwritten grayscale digit, 60,000 images for training and 10,000 for testing. CIFAR10 [18] (32 × 32 RGB) has 50,000 training images and 10,000 testing images from 10 classes. CIFAR100 [18] (32 × 32 RGB) contains 100 classes, each with 500 training images and 100 test images.

Models. We evaluate the model using two training schemes. In the first scheme, training from scratch, we apply block circulant matrix and perform DP training on all layers. We choose four neural networks with different structures. The first model (`Model1`) follows the structure evaluated in [22], which contains 4 FC layers with 2048, 1024, 160 and 10 neurons. The second model (`Model2`) uses a similar architecture as [34], consisting of 5 CONV layers with Tanh $(\cdot)$ activation function. The detailed structure is shown in Table 12 in Appendix A.5. We further use the LeNet-5 [20] for MNIST and a model (`Model3`) that is a similar variant of `Model2` with 6 convolutional layers and 2 FC layers for CIFAR10 as details shown in Table 13 in Appendix A.5. In the second scheme, transfer learning, we load a ResNet-18 [17] model that is trained over a public dataset (ImageNet [19]) and perform transfer training on CIFAR10. Furthermore, we use a ResNeXt-29 [47] for transfer learning from CIFAR100 to CIFAR10, following the settings in [44]. We also follow the state-of-the-art work [10] by using a Wide-ResNet (WRN-28-10) model [53] pretrained on down-sampled 32×32 ImageNet images [9] to perform the transfer learning on CIFAR10 and CIFAR100 datasets. DP training is applied to the predefined trainable layers.

Evaluation metrics. *Testing accuracy:* the accuracy of a DP trained method on the testing set. A good defense should obtain the testing accuracy close to that of a model trained without differential privacy. *Privacy Budget (ϵ, δ):* to measure the privacy constraint of DP training. We set $\delta = 10^{-5}$ for all training and conduct training in two ways. The first way gives a target privacy budget (ϵ, δ) and sets the training epochs. We report the model accuracy after training. The second way sets the noise scale (σ) and trains the model for at most 200 epochs, we report the best accuracy epoch and the corresponding accumulated privacy budget (ϵ, δ).

Parameters setting and general guidelines. The key parameter for *Spectral-DP* is the filtering ratio ρ that controls the balance between the DP noise and reconstruction noise. We will discuss the impact of choosing different filtering ratios in Section 4.2.1 and 4.2.2. In general, we find that while filtering more coefficients leads to a reduction in the added DP noise, a high filtering ratio results in a greater loss of utility. As a general rule, we keep at least 50% of the coefficients in the frequency domain. For more complex tasks, a smaller filtering rate (leaving more coefficients to add noise) may lead to a better utility-privacy tradeoff.

TABLE 1: Results for different models with DP-SGD and proposed *Spectral-DP* training.

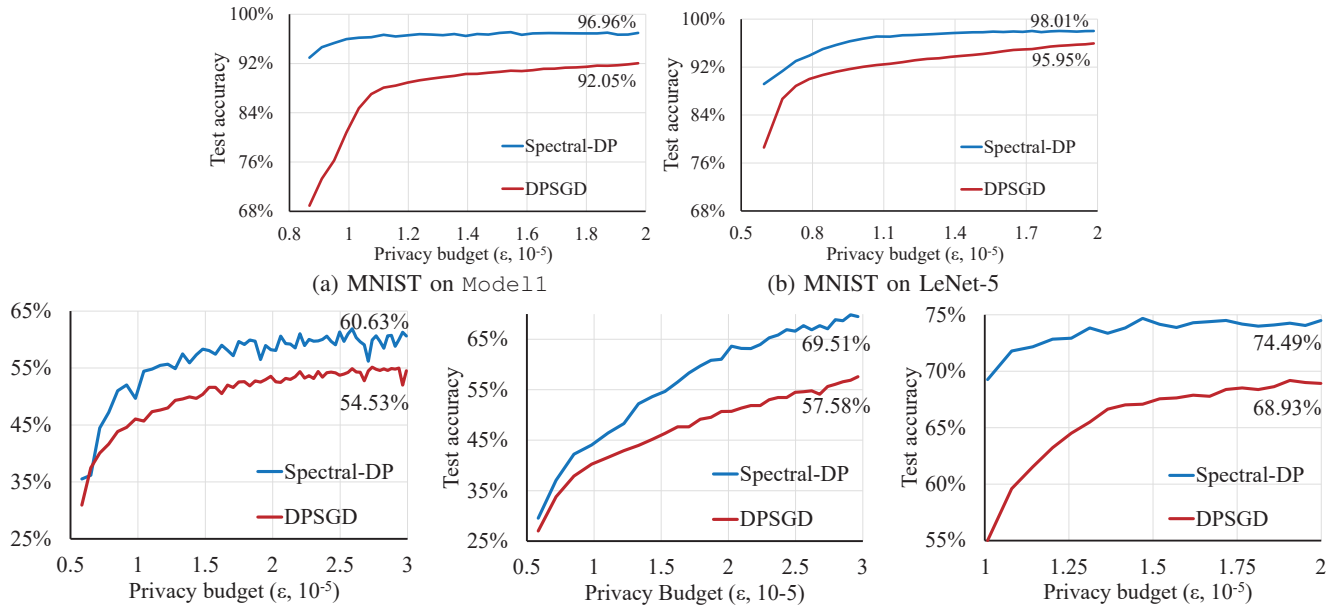
Dataset	MNIST		CIFAR10	
Model	`Model1`	LeNet-5	`Model2`	`Model3`
Privacy budget	$(2, 10^{-5})$	$(2, 10^{-5})$	$(3, 10^{-5})$	$(3, 10^{-5})$
Non-Private	97.87%	99.15%	77.23%	81.22%
DP-SGD	92.05%	95.95%	55.15%	57.58%
Spectral-DP	97.1%	98.03%	61.88%	69.51%
Max gain	34.85%	13.48%	19.18%	25.45%
Average gain	10.70%	4.49%	11.91%	19.92%

Block size is another hyperparameter used in *Block Spectral-DP* to determine the size of the block circulant matrix used in FC layer. Similar to existing work [11]–[13], specifically, we set the block size of the final FC layer as 10, and other layers as 8 for all models. We set a uniform clipping norm to 0.5 for `Model1`, while other models as 0.1. For training from scratch models, we set the learning rate as 0.01 for `Model1` and `Model2` models, and 0.001 for LeNet-5 and `Model3`. The batch size is 500. For the ResNet-18 model used in transfer learning, we choose a learning rate of 0.001, a clipping bound $C = 0.1$, and a filtering ratio $\rho = 0.2$ for CONV layers. The batch size is 256. For the last two FC layers with 160 and 10 neurons, block sizes for BCM are 16 and 10 with the number of preserved coefficients $k = 8$.

4.2. *Spectral-DP* for Training from Scratch Models

We set target privacy budget to $(2, 10^{-5})$ for MNIST and $(3, 10^{-5})$ for CIFAR10 and train the model for 30 epochs from scratch. The test accuracy and privacy budget plots for all models are shown in Figure 3. The best accuracy is reported in Table 1. *Spectral-DP* performs well for MNIST tasks on small models and gains more benefits from more sophisticated model training for CIFAR10 tasks. Table 1 shows that *Spectral-DP* can maintain accuracy with $\sim 1\%$ accuracy drop compared to non-private models on MNIST dataset under privacy budget $\epsilon = 2$. For CIFAR10 task, we can achieve the accuracy as high as 69.51% for privacy budget $(3, 10^{-5})$. As a comparison, the state-of-the-art DP training accuracy in [34] only delivers 66.2% accuracy even at a much higher privacy budget–$(7.53, 10^{-5})$. We match this accuracy with $\epsilon = 2.43$, which is an improvement in the DP-guarantee of $e^{5.1} \approx 164$.

According to Figure 3, we observe that both *Spectral-DP* and DP-SGD exhibit a similar trend, i.e., relaxing privacy constraint increases the accuracy. But *Spectral-DP* always outperforms DP-SGD in all cases. In most cases, *Spectral-DP* achieves much more accuracy improvement compared to DP-SGD under strict privacy budget constraints. In particular, in Figure 3a, when $\epsilon = 1$, our approach has resulted in 18.75% accuracy improvements. Overall, *Spectral-DP* leads to $13.48\% \sim 34.85\%$ max accuracy gain and on average



(a) MNIST on `Model1` (b) MNIST on LeNet-5

(c) CIFAR10 on `Model2` (d) CIFAR10 on `Model3` (e) CIFAR10 on ResNet-18 transfer learning

Figure 3: Achieved test accuracy at each privacy budget $(\epsilon, 10^{-5})$ on models and datasets with Spectral-DP and DP-SGD.

TABLE 2: Test accuracy on CIFAR10 dataset with different filtering ratios for *Spectral-DP* training on `Model2` models under different privacy budgets.

Privacy Budget	Filtering ratio (ρ)				
	0	0.25	0.5	0.75	0.875
$(3, 10^{-5})$	58.34%	58.38%	60.26%	**60.63%**	55.56%
$(5, 10^{-5})$	63.74%	63.18%	**65.45%**	62.20%	55.79%
$(7, 10^{-5})$	65.65%	65.58%	**66.56%**	62.28%	58.93%

$49\% \sim 19.92\%$ accuracy gain among all the privacy budget cases compared to DP-SGD.

4.2.1. Filtering ratio choice of *Spectral-DP* for convolutional layer. Without loss of generality, we use `Model2` as an example to evaluate the effectiveness of *Spectral-DP* with different filtering ratio ρ. We extend the tight privacy budget $(3, 10^{-5})$ evaluated in the previous section with larger ϵ values (5 and 7) and different filtering ratios on convolutional layers to explore utility improvements in more settings as shown in Table 2. It is intuitive that increasing the filtering ratio decreases the dimension of the additive differentially private noise, but causes more information loss of the weights. We note that there is a tradeoff between the noisy error and the reconstruction error (information loss) which is controlled by the filtering ratio. As Table 2 show, there is a significant accuracy loss for large filtering ratios ($\rho = 0.875$) in all cases. The models achieve the highest accuracy at $\rho = 0.5$ with $\epsilon = 5$ and 7. With a tight privacy budget of $\epsilon = 3$, adding noise causes more prominent utility loss, and filtering more coefficients with $\rho = 0.75$ provides better accuracy. **In general, $\rho = 0.5$ can be a good starting point to achieve the best utility for convolutional layers.**

4.2.2. Filtering ratio and block size choice of *Block Spectral-DP* for FC layer. We apply DP-SGD and *Spectral-DP* on `Model1` under a fixed privacy budget $(2, 10^{-5})$ with various hyper-parameters such as batch size, learning rate, and clipping bound. We report the best test accuracy over all running cases in Table 3. We apply block circulant matrices to DP-SGD using the same block sizes as *Spectral-DP*. We observe an accuracy improvement for DP-SGD trained circulant `Model1`. By using *Block Spectral-DP* for training, we further take advantage of spectral domain based noise reduction and spectral filtering and achieve much better utility than DP-SGD.

TABLE 3: Test accuracy of DP-SGD and *Block Spectral-DP* on MNIST dataset with privacy budget $(2, 10^{-5})$.

Methods	`Model1`	Circulant `Model1`	
		BS =8	BS =16
Non-Private	98.48%	97.87%	97.38%
DP-SGD	93.55%	94.77%	95.54%
Spectral-DP	N/A	96.96%	96.85%

We further explore the impact of block size (BS) and filtering ratio (ρ) using `Model1`. Two different block sizes (8 and 16) under four target differential privacy budgets are evaluated. The results under different filtering ratios are shown in Table 4. Overall, models with $BS = 8$ achieve better utility. The average model accuracy at four target ϵ across all five filtering ratios, is 94.02%, 95.78%, 96.33%, and 96.57%, respectively, which is higher than that with $BS = 16$ (92.90%, 95.29%, 96.03%, 96.17%). This trend is consistent with that of non-private `Model1`, which is 97.89% with $BS = 8$ and 97.38% with $BS = 16$ (in Table 3). It indicates that the FC layer often has redundancy and the block circulant matrix can help us further reduce the model size, and compressing model weights benefit the spectral calculation of *Block Spectral-DP*. **Generally, we can adopt a large BS (a power 2 number such as 16) for complex models often containing more redundancy in FC layers, and a small BS (i.e. 8) for small models.**

TABLE 4: Test accuracy on MNIST dataset with different filtering ratios and block sizes for *Block Spectral-DP* training on `Model1` under different privacy budgets.

Block size	Privacy Budget	Filtering ratio (ρ)				
		0	0.25	0.5	0.75	0.875
8	$(0.5, 10^{-5})$	93.08%	93.69%	93.64%	**94.98%**	94.71%
	$(1.0, 10^{-5})$	95.07%	95.77%	95.57%	96.19%	**96.29%**
	$(1.5, 10^{-5})$	95.93%	96.24%	96.35%	**96.60%**	96.52%
	$(2.0, 10^{-5})$	95.83%	96.58%	96.59%	**96.96%**	96.89%
16	$(0.5, 10^{-5})$	91.57%	92.92%	93.32%	93.30%	**93.41%**
	$(1.0, 10^{-5})$	95.04%	94.08%	95.73%	95.78%	**95.80%**
	$(1.5, 10^{-5})$	95.55%	95.76%	**96.46%**	96.21%	96.15%
	$(2.0, 10^{-5})$	96.02%	95.81%	96.41%	**96.85%**	95.74%

For filtering ratio (ρ), as shown in Table 4, a larger ρ on FC layers leads to better utility in most cases. By filtering more frequency coefficients, adding DP noise becomes more smoothly, and the loss due to the large filtering ratio can be compensated. As a result, FC layer with a larger ρ often benefits the utility in *Block Spectral-DP* training. Therefore, **FC layers in general can have a larger filter ratio than that of convolutional layers ($\rho = 0.75$ vs. $\rho = 0.5$).**

4.3. *Spectral-DP* in Transfer Learning

4.3.1. ResNet-18. In this section, we evaluate the performance of the proposed *Spectral-DP* training on the transfer learning setting. Specifically, we select three transfer training models with different numbers of trainable layers from the bottom of the pretrained ResNet-18. We set the noise scale $\sigma = 0.9$ and train the models for 100 epochs, the best accuracy epoch and corresponding privacy budget are reported in Table 5. The 1FC layer model (`Transfer1`) follows previous work [1], [52]–retraining only a hidden layer with 1000 units and a softmax layer with differential privacy. The `Transfer2` consists of the last 2 CONV layers and 2 FC layers of the model to be trained. `Transfer3` further increases the trainable layers to the last 4 CONV layers and 2 FC layers. The baseline accuracy of non-private model increases from 62.31% to 75.94% as we increase the number of trainable layers on transfer learning models.

Spectral-DP can benefit more from the increasing number of trainable layers. When increasing to 6 trainable layers in `Transfer3`, *Spectral-DP* achieves model accuracy close to the non-private model (75.32% vs 75.94%) at a small privacy budget (ϵ=2.15). In contrast, the gain of DP-SGD is limited. It suffers more accuracy degradation (6.61%) even with lower privacy guarantee (ϵ=2.89). This clearly indicates that our *Spectral-DP* works much better than DP-SGD when protecting more layers' weights for better privacy is needed. **This further highlights the key advantage of our *Spectral-DP*–better preserving model utility than DP-SGD especially for training models from the scratch with a high-level privacy requirement**, as validated in Section 4.2. Since we obtain the best accuracy in `Transfer3`, we conduct the following experiments on this model.

Spectral-DP can always provide a better tradeoff between utility and privacy than DP-SGD in differential private transfer learning. Figure 4 shows the results of the `Transfer3` transfer training with different target privacy budgets. *Spectral-DP* always provides higher accuracy than DP-SGD, from the case with strict privacy constraint (small ϵ) to cases with relaxed privacy requirements. Under an extreme privacy constrain (e.g. ϵ=0.5), our method only causes a 4.41% accuracy drop compared to the non-privacy model (71.52% vs 75.94%), while DP-SGD only achieves 63.60%, yielding a 12.33% accuracy loss.

TABLE 5: Transfer learning results for non-private model, DP-SGD training and *Spectral-DP* training with different number of trainable layers.

Model	`Transfer1`		`Transfer2`		`Transfer3`	
	ϵ	Test acc	ϵ	Test acc	ϵ	Test acc
Non-private	∞	62.31%	∞	70.08%	∞	75.94%
DP-SGD	3.88	60.10%	3.24	66.47%	2.89	69.33%
Spectral-DP	**2.11**	**59.11%**	**2.11**	**70.75%**	**2.15**	**75.32%**

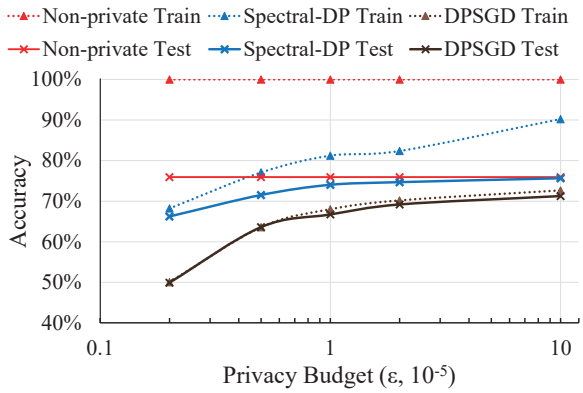


Figure 4: Model accuracy under different privacy budget $(\epsilon, 10^{-5})$ for CIFAR10 transfer models.

4.3.2. ResNeXt-29. In addition to the layer-wise exploration on ResNet-18, we also adapt our *Spectral-DP* for comparison according to the state-of-the-art transfer learning setup. We consider the setting proposed in [44] for transfer learning from CIFAR100 to CIFAR10. An FC layer-based model is trained on features that are extracted from a ResNeXt [47] model trained on CIFAR100. DP-SGD and *Spectral-DP* are implemented on the FC layer-based model. Our results are reported in Table 6. We also compare our results with a DP-SGD utility improvement method [24] that uses transfer learning. Overall, *Spectral-DP* outperforms the DP-SGD trained models in both [44] and [24] across different privacy budgets ϵ.

4.3.3. WRN-28-10. We further evaluate *Spectral-DP* using a pretrained WRN-28-10 model from the down-sampled ImageNet32 dataset to perform transfer training on CIFAR10 and the more complex CIFAR100 datasets. As presented in Table 7 and Table 8, we retrain the classifier of the WRN model for 20 epochs with different privacy budgets and compare the results with several state-of-the-art works [10], [44], [50]. The 1 FC layer setting retrains the last layer with 1000 units, while the 2 FC layers setup retrains the whole classifier layer. Our results show a similar trend–the utility of DP training improves as the number of training layers increases. *Spectral-DP* achieve 94.85% and 77.52% at $\epsilon = 1$

TABLE 6: ResNeXt-29 for transfer learning on CIFAR10

ϵ	0.5	1.0	1.5	2.0	∞
DP-SGD in [44]	-	-	-	80.00%	84.00%
Spectral-DP	**80.29%**	**80.81%**	**81.71%**	-	84.00%
[24]	73.28%	76.64%	81.57%	-	94.10%

TABLE 7: WRN-28-10 transfer learning on CIFAR10

ϵ	1	2	4
Spectral-DP (2 FC layers)	**94.85%**	**95.11%**	**95.33%**
Spectral-DP (1 FC layer)	93.19%	93.24%	93.36%
DeepMind (2022) [10]	93.10%	93.60%	94.00%
GEP (2021) [50]	94.30%	94.80%	-
Feature extraction (2021) [44]	-	92.70%	-

in 2 FC layers setting for CIFAR10 and CIFAR100, which is higher than the retraining results (93.36% and 75.99%) under a relaxed budget $\epsilon = 4$ in 1 FC layer training setting.

Our *Spectral-DP* achieves higher accuracy with a strict privacy budget (94.85% with $\epsilon = 1$) compared to other works with relaxed budgets (94% with $\epsilon = 4$ [10], 94.80% with $\epsilon = 2$ [50] and 92.70% with $\epsilon = 2$ [44]). Even when compared to the strongest setting in [10] (fine-tuning all layers), *Spectral-DP* can achieve similar accuracy with $\epsilon = 1$ on CIFAR10 (94.8%) while exhibiting much higher accuracy (77.52% with $\epsilon = 1$) on the more complex 100-class dataset–CIFAR100 than that of [10] (e.g. 74.7% at a relaxed privacy budget $\epsilon = 2$). **These results demonstrate that *Spectral-DP* can achieve better utility on more complex datasets and large models as well.**

4.4. Replace DP-SGD with *Spectral-DP* in DP-SGD based Existing Works

Since *Spectral-DP* is proposed as an alternative algorithm to DP-SGD, techniques orthogonal to DP-SGD can be integrated into it as well. Therefore, we can easily replace DP-SGD with *Spectral-DP* in the DP-SGD based existing frameworks, to achieve further utility improvement. Specifically, we combine a state-of-the-art work [44] with our *Spectral-DP* in the training to show the scalability of our approach. We adopt the same setting from [44] that uses the default Scattering Network (ScatterNet) of depth two with wavelets rotated along eight angles from [30] as a feature extractor to preprocess each data sample. We also apply the same data normalization from [44] on top of the ScatterNet features to obtain the best utility. With a target differential privacy budget of $(3, 10^{-5})$, we conduct a grid-search on hyperparameters and report the best results in Table 9. Here ScatterLinear and ScatterCNN adopt similar architectures used in [44]. We can observe that on MNIST, *Block Spectral-DP* outperforms DP-SGD on ScatterLinear with accuracy close to that of the non-private model. For CIFAR10, we show the training results on ScatterCNN, and CNN represents the DP training results on `Model3` as a baseline. We find that with *Spectral-DP* training, we obtain higher accuracy than that of DP-SGD on ScatterCNN. In addition, training with *Spectral-DP* on ScatterNet improves the accuracy by 1.42% compared to our CNN result, and the accuracy gap is less than 1% compared with the non-private ScatterCNN result. We also show the test accuracy and privacy budget plot for DP-SGD and *Spectral-DP* in Figure 5. The results indicate that *Spectral-DP* can always outperform DP-SGD and has more gains under tighter privacy budgets.

TABLE 8: WRN-28-10 transfer learning on CIFAR100

	ϵ	1	2	4
Spectral-DP	2 FC layers	**77.52%**	**77.78%**	78.03%
	1 FC layer	74.42%	75.65%	75.99%
DeepMind [10]	Classifier layer	70.30%	73.90%	76.10%
	All layers	67.40%	74.70%	**79.20%**

TABLE 9: Testing accuracy on ScatterNet based model with DP-SGD and proposed *Spectral-DP* training with privacy budget $(3, 10^{-5})$.

Methods	MNIST	CIFAR10	
	ScatterLinear	ScatterCNN	CNN
Non-private	99.10%	71.68%	81.22%
DP-SGD	97.66%	67.77%	57.58%
Spectral-DP	98.63%	70.93%	69.51%

4.5. Ablation Study

4.5.1. Training from scratch setting. We further analyze how different choices of clipping norm, batch size, and learning rate, impact model performance based on `Model3` and CIFAR10 dataset. The DP budget is set to be $(\epsilon = 3.0, \delta = 10^{-5})$, and the model is trained for 30 epochs.

Impact of clipping norm. Figure 6 shows how clipping norm (C) impacts the differentially private learning using *Spectral-DP*. We observe that the large clipping norm degrades the model utility. Since the scale of DP noise is proportional to the clipping norm, increasing the norm constraint causes increased gradient noise. However, too small clipping norms can lead to a large utility loss. This is because the gradient may turn out to be in the opposite direction of the true gradient if the clipping norm is set too low. This phenomenon is consistent with the description of the original DP-SGD [1]. To ensure an efficient *Spectral-DP* private learning, **we recommend choosing an appropriate clipping norm (beginning with $C = 0.1$).**

Impact of batch size. We select 4 batch sizes (B) and show the accuracy with each B in Figure 6. We observe that, unlike the non-private model training, B has a relatively large impact on the test accuracy. Changing B from 512 to 2048 leads to 8.17% accuracy improvement. This is because a larger B leads to fewer noise addition iterations. However, the noise scale at a single iteration is positively associated with B. When B is too large, the noise has a relatively larger effect than the training iterations. Therefore, an appropriate B is essential to balance the utility and the noise addition.

We also find that B and the clipping norm jointly affect the accuracy. When $B = 512$, the best accuracy is achieved with $C > 0.1$. Meanwhile, when $B = 4096$, the best accuracy is achieved by choosing $C < 0.1$. For *Spectral-DP* training with a fixed privacy budget, a larger batch size implies a larger noise size, while a smaller clipping norm can reduce the noise in the gradient. Therefore, a larger B is compacted with a relatively smaller clipping norm. As the batch size decreases, the noise scale decreases accordingly. In this case, it requires a relatively large clipping norm

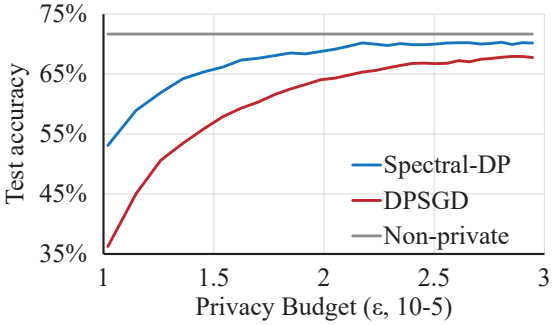


Figure 5: Test accuracy and target privacy budget $(\epsilon, 10^{-5})$ for ScatterCNN model on CIFAR10.

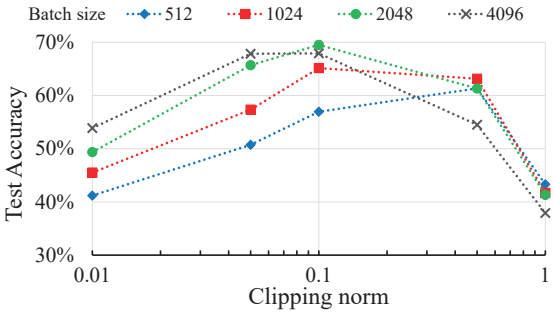


Figure 6: Impact of batch size (B) and clipping norm on test accuracy on CIFAR10 dataset.

to contain as much gradient information as possible. **We recommend starting *Spectral-DP* private learning with a relatively large batch size** $(B = 2048)$ **and a relatively small clipping norm** $(C = 0.1)$**.**

Impact of learning rate. Based on the study of batch size and clipping norm results, we pick the setting of $(B = 2048, C = 0.1)$ and perform *Spectral-DP* training with different learning rates (LR). Figure 7 shows the plots of the accuracy trends over the training epochs. Too small LR (LR=0.005) can lead to slow convergence of the model and reduce the accuracy over the target training epoch. A large LR=0.1 may also undermine the accuracy significantly. Since a larger LR boosts the weight noise, leading to a random gradient direction that hurts the training convergence. **The test accuracy is stable when the learning rate is set within a range of** $[0.01, 0.025]$**. Generally, we can set LR=0.01 for the *Spectral-DP* training.**

4.5.2. Transfer learning setting. In addition to the training from scratch setting, we also discuss the impact of hyperparameters in the transfer learning setting on CIFAR100. In general, the settings of the hyperparameters follow similar trends in the CIFAR10 model for transfer learning, details of which can be found in the Appendix A.6. We retrain the 2 FC layers of the pretrained WRN-28-10 model from ImageNet32 to CIFAR100 by 20 epochs. We adopt different batch sizes, filtering ratios, block sizes, and privacy budgets and summarize the results in Table 10.

The hyperparameter selection for transfer learning does not follow the same trend as that of training from scratch. First, we compare the impact of different batch sizes from 256 to 2048 with filtering ratio $\rho = 0$ and $\epsilon = 1$. As

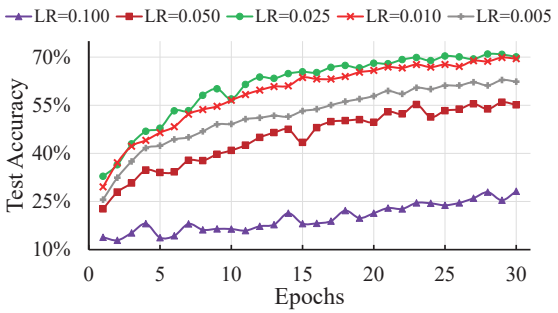


Figure 7: Test accuracy with different learning rate (LR) on CIFAR10 dataset.

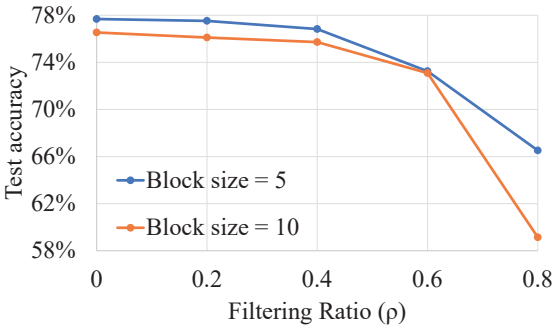


Figure 8: Model accuracy with different filtering ratios under privacy budget $(1, 10^{-5})$ with block size 5 and 10 for CIFAR100 transfer models.

Table 10 shows, the best accuracy is reached at a batch size of 256, and model accuracy decreases as the batch size increases. Since transfer learning just retrains a small number of parameters, a smaller batch size facilitates more fine-grained learning and ultimately results in improved utility.

Next we explore the impact of filtering ratios. We choose five filtering ratios (0, 0.2, 0.4, 0.6, and 0.8) for the two FC layers with the accuracy shown in Figure 8. Our findings show that a smaller filtering ratio results in better utility, indicating that *Block Spectral-DP* can benefit from the model training with noise added in the spectral domain and reconstruction error has a more significant impact on model accuracy in transfer learning settings.

Block size is not a dominating factor for model utility in this case. We can observe that from Figure 8, a smaller block size (5) leads to slightly better model utility, but its influence is not as significant as that of filtering ratios. This also indicates a tradeoff between efficiency and utility as a larger block size means fewer trainable parameters due to higher model compression, but lower model utility.

5. Limitations

Although *Spectral-DP* effectively reduces the DP noise and achieves better privacy utility tradeoff, there is scope for further improvement. First, while *Spectral-DP* achieves the same level of utility as that of the non-private learning for the transfer learning setup, there still exists the utility or accuracy gap for model training from scratch setup. Second, offering rigorous guidelines to select optimal key parameters

TABLE 10: Different settings of block size, filtering ratio, privacy budgets, and batch size on CIFAR100 dataset for WRN-28-10 transfer learning with 2 FC layers.

Block size	5	5	5	5	5	5	5	5	5	5	5	5	5
Filtering Ratio (ρ)	0	0	0	0	0.2	0.4	0.6	0.8	0	0	0	0	0
Privacy Budget ($\epsilon, 10^{-5}$)	1	1	1	1	1	1	1	1	2	4	2	4	4
Batch size	2048	1024	512	256	256	256	256	256	256	256	512	512	2048
Test Accuracy	73.65%	75.42%	76.59%	77.68%	77.52%	76.83%	73.25%	66.52%	77.78%	78.03%	76.98%	78.00%	77.60%

such as filtering ratio ρ, with theoretical guarantees is still challenging, due to the interplays among multiple factors, including dataset characteristics, model complexity, privacy budget, clipping specification, and batch size. A comprehensive analysis of these factors would be a promising avenue for future research. Third, there are other domain transform methods such as wavelet, etc. that can be explored for further improvement, as Spectral-DP opens up a new era for private deep learning using DP.

6. Related Work

The subject of differential privacy in the context of machine learning attracted significant scientific interest and has been used in support vector machines [40], linear regression [6], [54], and risk minimization [5], [7]. In recent years, more works have focused on privacy-preserving training for deep learning. Private Aggregation of Teacher Ensembles (PATE) [32], [33] is one approach that transfers the knowledge from an ensemble of teachers trained on the disjoint subsets of training data to train a student model through the noisy aggregation of teachers' answers.

Differentially private (stochastic) gradient descent (DP-SGD) [1], [52] as described earlier perturbs the gradient at each update with random noise drawn from Gaussian distribution during the training. Some recent works aim to do noise reduction on DP training by adding noise into the reduced gradient. In [50], a gradient embedding perturbation (GEP) is proposed to achieve higher utility by adding noise into a low-dimensional projected gradient. [51] designs reparametrized gradient perturbation (RGP), which perturbs the gradients of the low-rank gradient-carrier matrix and reconstructs the update of the original weights from the noisy gradients. The framework in [29] encodes gradients, mapping them to a smaller vector space, and hence is able to provide DP guarantees for different noise distributions.

These gradient dimension reduction techniques rely on either a projection or decomposition that maps gradients into a smaller subspace. We note that the key principle of these methods is gradient approximation, and therefore they would inevitably cause undesired utility loss. Our work involves performing lossless transformation of gradients using the Fourier transform and applying filtering in the spectral domain to improve privacy utility trade-off. Fourier Perturbation Algorithm (FPA) is proposed in [38] and further optimized in [2] to address the poor performance of conventional differential privacy aggregation algorithm for time-series data. FPA focuses on the differential privacy of time-series data and conducts noise aggregation in the frequency domain, which is similar to *Spectral-DP*.

Several recent works focus on a wealth of areas to improve the utility of the DP-SGD trained models. In Section 1, we discussed tempered sigmoid activations [34] introduced to help control the gradient norm of the loss function, thus mitigating the negative effects of clipping and noising. [24] leverages additional public data transfer learning to minimize the number of trainable parameters in the model to optimize the privacy-utility tradeoff. [8] proposes a framework to perform a neural architecture search with DP-aware candidate model training to find the suitable model for DP training. Work in [44] as mentioned earlier demonstrates that better features in data can lead to higher utility of DP-SGD trained model.

We note that these works focus on how to improve DP-SGD by modifying the model structure or preprocessing the data. Furthermore, other directions including i.e. clipping [3], [36], [42] and privacy budget allocation [4], [21], [52] do not change the DP-SGD algorithm. In contrast, *Spectral-DP* focuses on the gradient updating algorithm of DP training that adapts spectral domain DP perturbation into deep learning as an alternative to DP-SGD.

Another emerging topic is the use of differential privacy to protect privacy and robustness in federated learning (FL) [25]. Client-based Differential Privacy has been introduced in [16], [26] in order to hide any information that is specific to a single client's training data. Noising before model aggregation FL (NbAFL) [46] and LDP-Fed [45] perturb the trained parameters locally in each client before aggregation to ensure local differential privacy. [43] proposes a new protocol for differentially private secure aggregation based on techniques from Learning With Errors [39]. As future work, it would be a good opportunity to adapt and combine our *Spectral-DP* to the FL process to provide better privacy-utility tradeoffs in these complex scenarios.

7. Conclusion

In this work, we propose *Spectral-DP*, an alternative to DP-SGD in the context of differentially private deep learning. *Spectral-DP* combines differentially private noise addition in the spectral domain with spectral filtering which enables reduction of noise scale to improve utility. Our extensive experimental results show that our approach has uniformly better privacy utility tradeoff compared to state-of-the-art methods. Our contribution is a new paradigm to gradient perturbation in the context of deep learning, which can be further built upon. For instance, Fourier is only one example of a unitary transformation, and although it is widely used, other transformations could be considered for spectral perturbation. Likewise, developing alternative weight restructuring, and more general filtering approaches might yield interesting and broader insights into the general principle of spectral domain based methods to achieve differential privacy in deep learning.

Acknowledgment

This research was partially supported by the National Science Foundation through the grants CCF-1617889, CCF-2011236, CCF-2006748, and partially through a Lehigh internal CORE grant.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[2] Gergely Acs, Claude Castelluccia, and Rui Chen. Differentially private histogram publishing through lossy compression. In *2012 IEEE 12th International Conference on Data Mining*, pages 1–10. IEEE, 2012.

[3] Galen Andrew, Om Thakkar, H. Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping, 2019.

[4] Hilal Asi, John Duchi, Alireza Fallah, Omid Javidbakht, and Kunal Talwar. Private adaptive gradient methods for convex optimization. In *International Conference on Machine Learning*, pages 383–392. PMLR, 2021.

[5] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*, pages 464–473. IEEE, 2014.

[6] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. *Advances in neural information processing systems*, 21, 2008.

[7] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[8] Anda Cheng, Jiaxing Wang, Xi Sheryl Zhang, Qiang Chen, Peisong Wang, and Jian 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.

[9] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[10] Soham De, Leonard Berrada, Jamie Hayes, Samuel L Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*, 2022.

[11] Caiwen Ding, Siyu Liao, Yanzhi Wang, Zhe Li, Ning Liu, Youwei Zhuo, Chao Wang, Xuehai Qian, Yu Bai, Geng Yuan, Xiaolong Ma, Yipeng Zhang, Jian Tang, Qinru Qiu, Xue Lin, and Bo Yuan. Circnn: Accelerating and compressing deep neural networks using block-circulant weight matrices. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, page 395–408, New York, NY, USA, 2017. Association for Computing Machinery.

[12] Caiwen Ding, Shuo Wang, Ning Liu, Kaidi Xu, Yanzhi Wang, and Yun Liang. Req-yolo: A resource-aware, efficient quantization framework for object detection on fpgas. In *proceedings of the 2019 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 33–42, 2019.

[13] Shi Dong, Pu Zhao, Xue Lin, and David Kaeli. Exploring gpu acceleration of deep neural networks using block circulant matrices. *Parallel Computing*, 100:102701, 2020.

[14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, aug 2014.

[15] Wayne Eberly. Polynomial and matrix computations volume 1: Fundamental algorithms (dario bini and victor pan). *SIAM Review*, 38(1):161–165, 1996.

[16] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Jaewoo Lee and Daniel Kifer. Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1656–1665, 2018.

[22] Sheng Lin, Ning Liu, Mahdi Nazemi, Hongjia Li, Caiwen Ding, Yanzhi Wang, and Massoud Pedram. Fft-based deep learning deployment in embedded systems. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1045–1050, 2018.

[23] Tao Luo, Zheng Ma, Zhi-Qin John Xu, and Yaoyu Zhang. Theory of the frequency principle for general deep neural networks. *CSIAM Transactions on Applied Mathematics*, 2(3):484–507, 2021.

[24] Zelun Luo, Daniel J Wu, Ehsan Adeli, and Li Fei-Fei. Scalable differential privacy with sparse network finetuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5059–5068, 2021.

[25] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[26] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[27] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275, 2017.

[28] Ilya Mironov, Kunal Talwar, and Li Zhang. Rényi Differential Privacy of the Sampled Gaussian Mechanism. *arXiv e-prints*, page arXiv:1908.10530, August 2019.

[29] Milad Nasr, Reza Shokri, et al. Improving deep learning with differential privacy using gradient encoding and denoising. *arXiv preprint arXiv:2007.11524*, 2020.

[30] Edouard Oyallon and Stéphane Mallat. Deep roto-translation scattering for object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2865–2873, 2015.

[31] Victor Y. Pan. *Structured Matrices and Polynomials: Unified Superfast Algorithms*. Springer-Verlag, Berlin, Heidelberg, 2001.

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

[33] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

[34] Nicolas Papernot, Abhradeep Thakurta, Shuang Song, Steve Chien, and Úlfar Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9312–9321, May 2021.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[36] Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X. Yu, Sashank J. Reddi, and Sanjiv Kumar. Adaclip: Adaptive clipping for private sgd, 2019.

[37] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[38] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, page 735–746, New York, NY, USA, 2010. Association for Computing Machinery.

[39] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[40] Benjamin IP Rubinstein, Peter L Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *arXiv preprint arXiv:0911.5708*, 2009.

[41] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2017.

[42] Timothy Stevens, Ivoline C. Ngong, David Darais, Calvin Hirsch, David Slater, and Joseph P. Near. Backpropagation clipping for deep learning with differential privacy, 2022.

[43] Timothy Stevens, Christian Skalka, Christelle Vincent, John Ring, Samuel Clark, and Joseph Near. Efficient differentially private secure aggregation for federated learning via hardness of learning with errors. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1379–1395, 2022.

[44] Florian Tramer and Dan Boneh. Differentially private learning needs better features (or much more data). In *International Conference on Learning Representations*, 2021.

[45] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, pages 61–66, 2020.

[46] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[47] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[48] Zhi-Qin John Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. In *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part I 26*, pages 264–274. Springer, 2019.

[49] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*, 2021.

[50] Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. In *International Conference on Learning Representations*, 2021.

[51] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. Large scale private learning via low-rank reparametrization. In *International Conference on Machine Learning*, pages 12208–12218. PMLR, 2021.

[52] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. Differentially private model publishing for deep learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 332–349, 2019.

[53] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

[54] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. Functional mechanism: regression analysis under differential privacy. *arXiv preprint arXiv:1208.0219*, 2012.

Appendix A.

A.1. Proof of Theorem 1

Theorem 1. *In Algorithm 1, the output $\tilde{Q_n}$ is (ϵ, δ) differentially private if we choose σ to be $\sqrt{2\log(1.25/\delta)}/\epsilon$.*

Proof. The proof relies on the Theorem 3.22 in [14] and the post-processing property of DP algorithm. First, we show that $\tilde{F^N}$ is (ϵ, δ) if $\sigma = \sqrt{2\log(1.25/\delta)}/\epsilon$. Since the mechanism of obtaining $\tilde{F^N}$ is a Gaussian mechanism, and the variance of the $2S^2\log(1.25/\delta)/\epsilon^2$ where S is the sensitivity of F^N, Following the Theorem 3.22 in [14], $\tilde{F^N}$ is (ϵ, δ) differentially private under the condition that the L_2-sensitive of F^N is S. The spectral filtering and the inverse Fourier transformation are the post-processing of $\tilde{F}^N$. Since the post-processing does not change the differential privacy budget, the Fourier DP follows the same privacy budget as the Gaussian mechanism. □

Theorem 2. *(Theorem 3.22 in [14]) Let $\epsilon \in (0, 1)$ be arbitrary. For $c^2 > 2ln(1.25/\delta)$, the Gaussian Mechanism with parameter $\sigma \geq c\triangle_2(f)/\epsilon$ is (ϵ, δ) differentially private.*

A.2. Proof of Proposition 3

Proposition 3. *Let $V^N = \{V_0, V_1, \cdots, V_i, \cdots, V_{N-1}\}$ be a collection of noise vector in spectral domain, and each V_i is drawn from $\mathcal{N}(0, \sigma^2 S^2)$. Consider $v_n =$ I-FT$(P^K(V_i))$, then v_n follows a normal distribution $\mathcal{N}(0, \frac{K}{N}\sigma^2 S^2)$.*

Proof. Let $V_i^K = P_K(V_i)$, then the resulting $v_n =$ I-FT(V_i^K). In detail, it can be formulated as

$$\begin{aligned} v_n &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} V_i^K \cdot e^{\frac{j2\pi}{N} in} \\ &= \frac{1}{\sqrt{N}} \sum_{i=0}^{K} V_i \cdot e^{\frac{j2\pi}{N} in} \\ &= \frac{1}{\sqrt{N}} \sum_{i=0}^{K-1} \{V_i\} \cdot \cos(\tfrac{2\pi}{N} in) \\ &\quad + j\frac{1}{\sqrt{N}} \sum_{i=0}^{K-1} \{V_i\} \cdot \sin(\tfrac{2\pi}{N} in) \end{aligned}$$

Define $c_{n,i} = \frac{1}{\sqrt{N}}\{V_i\} \cdot \cos(\frac{2\pi}{N} in)$, then we have

$$\frac{1}{\sqrt{N}} \sum_{i=0}^{K-1} \{V_i\} \cdot \cos(\frac{2\pi}{N} in) = \sum_{i=0}^{K-1} c_{n,i}$$

Following the property of the normal distribution, we have

$$\sum_{i=0}^{K-1} c_{n,i} \sim \mathcal{N}(0, \sum_{i=0}^{K-1} \frac{1}{N} \cdot \cos^2(\frac{2\pi}{N} in)\sigma^2 S^2)$$

Simplifying the variance of the distribution, we have

$$\sum_{i=0}^{K} c_{n,i} \sim \mathcal{N}(\frac{K}{N} \cdot \frac{\sigma^2 S^2}{2})$$

Similarly, we have $\frac{1}{\sqrt{N}} \sum_{i=0}^{K} \{V_i\} \cdot \sin(\frac{2\pi}{N} in) \sim \mathcal{N}(\frac{K}{N} \cdot \frac{\sigma^2 S^2}{2})$. This indicates that $v_n \sim \mathbb{C}\mathcal{N}(\frac{K}{N} \cdot \sigma^2 S^2)$ has the same scale of $\mathcal{N}(\frac{K}{N} \cdot \sigma^2 S^2)$ □

A.3. Proof of Proposition 3

Corollary 1. *Let V^N be the collection of a noise vector $\{V_{i,j}\}$ where $i \in \{0, 1, \cdots, N-1\}$ and $j \in \{0, 1, \cdots, N-1\}$ in spectral domain, and each $V_{i,j}$ be drawn from $\mathcal{N}(0, \sigma^2 S^2)$, consider a 2D spectral filtering:*

$$P_{2D}^K = \begin{cases} V_{ij} & \text{if } i < K \text{ and } j < K \\ 0 & \text{otherwise} \end{cases}$$

and $v_{mn} = \mathcal{F}^{-1}(P_{2D}^K(V_{i,j}))$, then v_{mn} follows a normal distribution $\mathcal{N}(0, \frac{K^2}{N^2}\sigma^2 S^2)$.

Proof. Let $V_{i,j}^K = P_{2D}^K(V_{i,j})$, then the resulting $v_{mn} = \mathcal{F}^{-1}(V_{ij}^K)$. In detail, it can be formulated as

$$\begin{aligned} v_{mn} &= \frac{1}{N} \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} V_{ij} \cdot e^{\frac{j2\pi}{N} im+jn} \\ &= \frac{1}{N} \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} \{V_{ij}\} \cdot \cos(\frac{2\pi}{N}(im+jn)) + \\ &\quad \sqrt{-1} \frac{1}{N} \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} \{V_{ij}\} \cdot \sin(\frac{2\pi}{N}(im+jn)) \end{aligned}$$

Define $c_{mn,ij} = \frac{1}{N}\{V_{i,j}\} \cdot \cos(\frac{2\pi}{N}(im+jn))$, then we have

$$\frac{1}{N} \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} \{V_{ij}\} \cdot \cos(\frac{2\pi}{N}(im+jn)) = \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} c_{mn,ij}$$

Following the property of the normal distribution, we have

$$\sum_{i=0}^{K-1} \sum_{j=0}^{K-1} c_{mn,ij} \sim \mathcal{N}(0, \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} \frac{1}{N^2} \cdot \cos^2(\frac{2\pi}{N}(im+jn))\sigma^2 S^2)$$

Simplifying the variance of the distribution, we have

$$\sum_{i=0}^{K-1} \sum_{j=0}^{K-1} c_{mn,ij} \sim \mathcal{N}(\frac{K^2}{N^2} \cdot \frac{\sigma^2 S^2}{2})$$

Similarly, we have $\sum_{i=0}^{K-1} \sum_{j=0}^{K-1} \{V_{ij}\} \cdot \sin(\frac{2\pi}{N} im + jn) \sim \mathcal{N}(\frac{K^2}{N^2} \cdot \frac{\sigma^2 S^2}{2})$. This indicates that $v_{mn} \sim \mathbb{C}\mathcal{N}(\frac{K^2}{N^2} \cdot \sigma^2 S^2)$ which has the same scale of $\mathcal{N}(\frac{K^2}{N^2} \cdot \sigma^2 S^2)$ □

A.4. Proof of Corollary 2

Corollary 2. *Algorithm 4 achieves $((T_e * N/B)\epsilon + \frac{\log(1/\delta)}{\alpha-1}, \delta)-DP$ if $\sigma = \frac{\sqrt{2\log(1.25/\delta)}}{\epsilon'}$ where $\epsilon' = \epsilon + \frac{\log(1/\delta)}{\alpha-1}$.*

Proof. In Algorithm 4, let $\sigma = \frac{\sqrt{2\log(1.25/\delta)}}{\epsilon'}$ and $\epsilon' = \epsilon + \frac{\log(1/\delta)}{\alpha-1}$, the Gaussian mechanism guarantees $(\epsilon', \delta)-$DP following Theorem 3.22 in [14]. This is equivalent with $(\alpha, \epsilon)-$RDP according to Proposition 1.

TABLE 11: Architecture of model that only consists of Fully Connected (FC) layers. A layer name ending with BC means that the weights matrix of such layer is block circulant.

Model-1		Circulant `Model1`		
Layer	Weight	Layer	Weight	Block
FC1	784×2048	FC1-BC	784×2048	8/16
FC2	2048×1024	FC2-BC	2048×1024	8/16
FC3	1024×160	FC3-BC	1024×160	8/16
FC4	160×10	FC4-BC	160×10	10

TABLE 12: Architecture of `Model2` model that consists of convolutional layers.

`Model2`	
Layer	Parameters
Convolution	32 filters of 3x3, stride 1, padding 1
Max-Pooling	2 × 2, stride 2
Convolution	64 filters of 3x3, stride 1, padding 1
Max-Pooling	2 × 2, stride 2
Convolution	64 filters of 3x3, stride 1, padding 1
Max-Pooling	2 × 2, stride 2
Convolution	64 filters of 3x3, stride 1, padding 1
Max-Pooling	2 × 2, stride 2
Convolution	10 filters of 3x3, stride 1, padding 1
Average	Over spatial dimensions

TABLE 13: The architecture of `Model3` model for CIFAR10.

`Model3`	
Layer	Parameters
Convolution x2	32 filters of 3x3, stride 1, padding 1
Max-Pooling	2 × 2, stride 2
Convolution x2	64 filters of 3x3, stride 1, padding 1
Max-Pooling	2 × 2, stride 2
Convolution x2	128 filters of 3x3, stride 1, padding 1
Max-Pooling	2 × 2, stride 2
Fully connected	120 units
Fully connected	10 units

TABLE 14: The architecture of ScatterCNN model for CIFAR10, with Tanh activations.

ScatterCNN	
Layer	Parameters
Convolution	64 filters of 3x3, stride 1, padding 1
Max-Pooling	2 × 2, stride 2
Convolution	60 filters of 3x3, stride 1, padding 1
Fully connected	10 units

Proposition 2 further shows Algorithm 4 satisfies $(\alpha, (T_e * N/B)\epsilon)-$RDP. The $(\alpha, (T_e * N/B)\epsilon)-$RDP can be converted back to $((T_e * N/B)\epsilon + \frac{\log(1/\delta)}{\alpha-1}, \delta)-$DP based on the Proposition 1. □

A.5. Model architectures

Here we show the detailed model architectures of the model used in the paper.

A.6. Ablation study on CIFAR10 transfer models

In this section, we also discuss the impact of hyper-parameters in the transfer learning setting based on the `Model3` and CIFAR-10 dataset. We summarize the results

TABLE 15: Different settings of training epochs, filtering ratios and privacy budgets on CIFAR10 dataset for ResNet-18 transfer learning with 4CONV+2FC trainable layers (`Transfer3`).

Training Epochs	10	10	10	20	40	10	10	10	20
Filtering Ratio (ρ)	0.75	0.5	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Privacy Budget ($\epsilon, 10^{-5}$)	0.5	0.5	0.5	0.5	0.5	0.2	1	2	2
Train Accuracy	76.36%	77.50%	77.68%	76.83%	74.39%	71.44%	81.11%	83.82%	86.55%
Test Accuracy	71.50%	72.61%	72.89%	71.98%	70.31%	68.52%	73.72%	74.53%	74.49%

with different pairs of training epochs, filtering ratio (ρ) and target privacy budget in Table 15. First, we discuss the impact of training epochs for a target privacy budget training. We set a strict target privacy budget $\epsilon = 0.5$ and conduct 10, 20, and 40 epochs of training. More training epochs results in less noise addition to each training step but leads to an increased number of training steps. Different choices of training epochs may affect the convergence and final accuracy of the model. As Table 15 shows, fewer training epochs leads to higher accuracy when $\epsilon = 0.5$. However, when the privacy budget is relaxed to $\epsilon = 2$, we observe the similar test accuracy for 10 and 20 epochs.

We thus conduct experiments with 10 training epochs and set ρ to 0.2, 0.5 and 0.75 under target privacy budget $(0.5, 10^{-5})$. Under this tight budget, setting ρ to 0.2 and 0.5 results in the similar model performance. However, unlike the cases in Section 4.2.1 for the training from scratch models, in transfer learning, setting a smaller rate like 0.2 gives better performance. The reason is because during the transfer learning process, we usually only tune the final layers of a pre-trained model, a large filtering ratio indicates losing too much information, thus worse accuracy.

Finally, we draw training curves using 10 training epochs with $\rho = 0.2$ under different privacy budgets in Figure 9. It demonstrates that our *Spectral-DP* allows the model to converge quickly in the first several epochs. When $\epsilon >= 0.5$, the training curves demonstrate the similar tradeoff between privacy budget and accuracy. The utility of the model is limited only when the privacy budget becomes very low ($\epsilon = 0.2$). In this case, training after the 5th epoch cannot further improve the accuracy. Nevertheless, this result (68.52%) is still higher than the DP-SGD trained result (66.72%) under a more relaxed privacy budget $\epsilon = 1$.

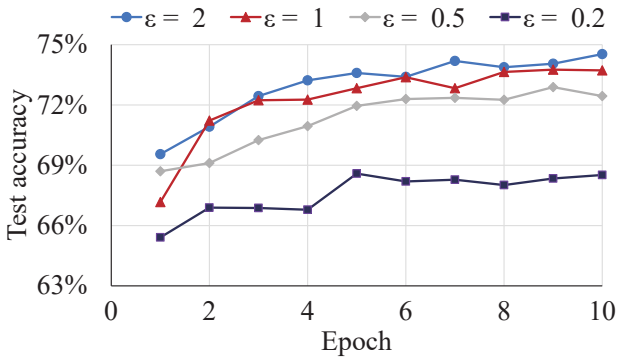


Figure 9: Test accuracy under different target privacy budgets $(\epsilon, 10^{-5})$ for CIFAR10 transfer models.

A.7. Discussion on complexity

A.7.1. Complexity of *Spectral-DP* in CONV layer. We implement the DFT using the Fast Fourier Transform (FFT) algorithm. Given a 2D convolution with signal size $n \times n$ and kernel size $d \times d$, the computational complexity of conventional convolution is $O(n^2 d^2)$. For FFT-based convolution, the complexity composes of two parts: First, the complexity of FFT operation is $O(n^2 \log n)$ and there are three times for doing the FFT (including FFT of the signal, FFT of the kernel, and inverse FFT of the spectral multiplication). Second, the multiplication operation has a complexity of $4n^2$. Therefore, the complexity of FFT-based convolution is $O(n^2 \log n)$. This implies that FFT-based convolutions are more computationally efficient if $log(n) < d^2$.

In Eq. (5), the gradient of $w_{i,j}$ is the convolution of the gradient of A_I and X_j. In neural network, most commonly used sizes of $w_{i,j}$ are 3×3 and 5×5. When computing the gradient of such $w_{i,j}$, the size of X_j is close to the size of A_i, leading a more efficient convolution using FFT.

A.7.2. Complexity of *Block Spectral-DP* in FC layer. An advantage of Block Spectral-DP is that it employs a block circulant matrix to reduce storage complexity. Specifically, for a $d \times d$ block circulant matrix, the storage complexity can be reduced from $O(d^2)$ to $O(d)$ by using a block size of d. In addition to this, the use of a block circulant matrix can also result in a reduction in computational complexity. For example, in a fully connected layer with a block circulant matrix-based weight of size $m \times n$, the computational complexity can be reduced from $O(n^2)$ to $O(n \log n)$ [11].