P3SGD: Patient Privacy Preserving SGD for Regularizing Deep CNNs in Pathological Image Classification

Bingzhe Wu[1]* Shiwan Zhao[2] Guangyu Sun[1] Xiaolu Zhang[3]
Zhong Su[2] Caihong Zeng[4] Zhihong Liu[4]

[1]Peking University [2]IBM Research [3]Ant Financial Service Group
{wubingzhe,gsun}@pku.edu.cn, {zhaosw,suzhong}@cn.ibm.com, yueyin.zxl@antfin.com
[4]National Clinical Research Center of Kidney Diseases, Jinling Hospital
zengch_nj@hotmail.com, liuzhihong@nju.edu.cn

Abstract

Recently, deep convolutional neural networks (CNNs) have achieved great success in pathological image classification. However, due to the limited number of labeled pathological images, there are still two challenges to be addressed: (1) overfitting: the performance of a CNN model is undermined by the overfitting due to its huge amounts of parameters and the insufficiency of labeled training data. (2) privacy leakage: the model trained using a conventional method may involuntarily reveal the private information of the patients in the training dataset. The smaller the dataset, the worse the privacy leakage.

To tackle the above two challenges, we introduce a novel stochastic gradient descent (SGD) scheme, named patient privacy preserving SGD (P3SGD), which performs the model update of the SGD in the patient level via a large-step update built upon each patient's data. Specifically, to protect privacy and regularize the CNN model, we propose to inject the well-designed noise into the updates. Moreover, we equip our P3SGD with an elaborated strategy to adaptively control the scale of the injected noise. To validate the effectiveness of P3SGD, we perform extensive experiments on a real-world clinical dataset and quantitatively demonstrate the superior ability of P3SGD in reducing the risk of overfitting. We also provide a rigorous analysis of the privacy cost under differential privacy. Additionally, we find that the models trained with P3SGD are resistant to the model-inversion attack compared with those trained using non-private SGD.

1. Introduction

In recent years, deep CNNs have emerged as powerful tools for various pathological image analysis tasks, such as tissue classification [30, 10], lesion detection [17, 22], nuclei segmentation [38, 50, 26], etc. The superior performance of deep CNNs usually relies on large amounts of labeled training data [40]. Unfortunately, the lack of labeled pathological images for some tasks may lead to two notorious issues: (1) overfitting of the CNN models [39, 42, 48] and (2) privacy leakage [48, 33, 9, 46] of the patients. Firstly, the performance of a CNN-based model is always harmed by the overfitting due to its large amounts of parameters and the insufficiency of training data. Secondly, pathological datasets usually contain sensitive information, which can be associated with each individual patient. The CNN-based models trained using conventional SGD may involuntarily reveal the private information of patients according to recent studies [48, 9]. For example, Zhang *et al.* [48] show that the CNN model can easily memorize some samples in the training dataset. Fredrikson *et al.* [9] propose a model-inversion attack to reconstruct images in the training dataset. In Figure 1 (a) and (b), we demonstrate an attacking example in our task, reconstructing the outline of a patch in the training dataset by leveraging a well-trained CNN model and its intermediate feature representations.

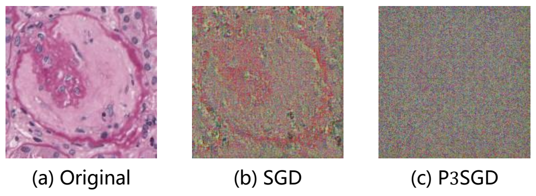

Figure 1. Illustration of the model-inversion attack on ResNet-18. (a) is the original image patch. (b) and (c) are the images reconstructed from the model trained with non-private SGD and P3SGD.

There have been numerous studies to solve either of two issues individually. On the one hand, to reduce the risk of overfitting in deep CNNs, previous research suggests

*Work performed while interning at IBM Research - China.

adding appropriate randomness into the training phase [39, 42, 45]. For example, Dropout [39] adds randomness in activation by randomly discarding the hidden layers' outputs. DropConnect [42] adds randomness in weight parameters by randomly setting weights to zero during training. On the other hand, differential privacy [5, 6] emerges as a strong standard, which offers rigorous privacy guarantees for algorithms applied on the sensitive database. Recent works [1, 31] are introduced to train deep CNN models within differential privacy. The main idea of these works is to perturb the gradient estimation at each step of an SGD algorithm. For example, Abadi et al. [1] use a differentially private additive-noise mechanism on the gradient estimation in an SGD. In addition, a few recent studies [33, 46] have shown that these two seemingly unrelated issues are implicitly relevant based on a natural intuition: *"reducing the overfitting"* and *"protecting the individual's privacy"* share the same goal of encouraging a CNN model to learn the population's features instead of memorizing the features of each individual.

In this paper, we propose a practical solution to alleviate both issues in a task of pathological image classification. In particular, we introduce a novel SGD algorithm, named P3SGD, which injects the well-designed noise into the gradient to obtain a degree of differential privacy and reduce overfitting at the same time. It is worth noting that a pathological database usually consists of a number of patients, each of whom is further associated with a number of image patches. We should protect the privacy in the **patient level** instead of image level as in most of the previous works. To achieve this goal, we propose to calculate the model update upon individual patient's data and add carefully-calibrated Gaussian noise to the update for both privacy protection and model regularization. The most similar work to ours is the differentially private federated learning [25, 11], which focuses on protecting the user-level privacy. In contrast to previous works, which use a globally fixed noise scale to build the noisy update [1, 25, 11], we propose an elaborated strategy to adaptively control the magnitude of the noisy update. In the experiment, we show that this strategy plays a key role in boosting performance of a deep CNN model. At last, we provide a rigorous privacy cost analysis using the moments accountant theorem [1].

In summary, the main contributions of our work are as follows:

- We introduce a practical solution, named P3SGD, to simultaneously address the overfitting and privacy leaking issues of deep CNNs in the pathological image classification. To the best of our knowledge, this is the first work to provide rigorous privacy guarantees in medical image analysis tasks.
- Technically, we present a strategy to dynamically control the noisy update at each iterative step, which leads to a significant performance gain against the state-of-the-art methods [25, 11].
- We validate P3SGD on a real-world clinical dataset, which is less explored in previous studies. The results demonstrate that P3SGD is capable of reducing the risk of overfitting on various CNN architectures. Moreover, P3SGD provides a strong guarantee that the trained model protects the privacy of each patient's data, even when the attacker holds enough extra side-information of the raw training dataset.
- We qualitatively and quantitatively demonstrate that the CNN model trained using P3SGD is resistant to the model-inversion attack [9] (shown in Figure 1 (c)).

2. Related Work

Regularization in CNNs In the past years, numerous regularization techniques have been proposed to improve the generalization ability of deep CNNs [20, 39, 42, 45, 41, 12]. These works mainly fall into two categories: explicit regularization and implicit (*i.e.*, algorithmic) regularization.

For explicit regularization methods, various penalty terms are used to constrain weight parameters. For example, weight decay [20] uses $l2$-regularization to constrain the parameters of a CNN model. Another direction is to introduce regularizers to decorrelate convolutional filters in deep CNNs [44, 32], which improves the representation ability of the intermediate features extracted by those filters.

For implicit regularization methods, the core idea is to introduce moderate randomness in the model training phase. For example, Dropout [39] randomly discards the outputs of the hidden neurons in the training phase. However, Dropout is originally designed for fully-connected layers (FC). It is often less effective for convolutional layers, which limits its use in CNNs with few FC layers (*e.g.*, ResNet). This is possibly caused by the fact that Dropout discards features without taking its spatial correlation into account (features from convolutional layers are always spatially correlated) [12]. To address this problem, a few recent works [41, 12] propose to inject structured noise into the features from convolutional layers. One state-of-the-art technique, named DropBlock [12], is specially designed for convolutional layers, which randomly drops the features in a sub-region. Both of Dropout and DropBlock inject randomness into activation layers. In contrast, DisturbLabel [45] adds randomness into the loss function by randomly setting a part of labels to be incorrect in a training mini-batch. Data augmentation is another form of algorithmic regularization, which introduces noise into the input layer by randomly transforming training images [36]. Our method can be categorized as implicit regularization. In contrast to previous works, our approach (P3SGD) imposes regularization at the parameter updating phase.

Privacy-preserving Deep Learning Meanwhile, there is an increasing concern for privacy leakage in deep learning models, since the training datasets may contain sensitive information. This privacy issue has attracted many research interests on the privacy-preserving deep learning [1, 35, 25, 11, 13, 28]. One promising direction is to build machine learning models within differential privacy [1, 25, 28], which has been widely used in sensitive data analysis as a golden standard of privacy. The early solution is to perturb the model parameters [3, 49] or the objective function [3, 19, 31]. However, such kind of simple solutions cause considerable performance decreasing [4, 43], the situation may become worse in the context of deep learning. Therefore, some recent studies focus on the gradient perturbation based methods [1, 25, 11, 28, 29]. Abadi *et al*. [1] propose a differentially private version of SGD and present the moments accountant framework to provide tighter privacy bound than previous methods. The PATE framework [28, 29] protects the privacy via transferring knowledge to the student model, from an ensemble of teacher models, which are trained on partitions of the training data.

Different from these works, which focus on image-level privacy, we aim to provide **patient-level** privacy in specific scenarios of pathological image analysis. The most similar works to ours are [25, 11], which extend the private SGD into the federated learning paradigm [24]. However, applying these approaches to the real-world medical image data remains less explored. Moreover, these methods always lead to a performance drop compared with the models trained using non-private SGD. In this paper, we evaluate our method on a real-world pathological image dataset and show that the performance drop can be addressed by carefully controlling the noisy update using our strategy.

There are also some studies aiming to explore the relationship between the overfitting and the privacy leakage issues from the perspective of memorization [33, 46]. In this paper, we present a practical solution to alleviate these two related issues simultaneously.

3. Our Approach

In this section, we describe our approach in details and provide a rigorous privacy cost analysis using the moments accountant theorem [1].

3.1. Preliminaries

We firstly introduce some basic notations and definitions of differential privacy corresponding to our specific task.

In our setting, the pathological image dataset can be regarded as a database $\mathcal{D}$ with N_p patients. Generally speaking, each patient D_i consists of a number of image patches of various tissues, *i.e.*, $\mathcal{D}_i = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^{N_i}$, where N_i is the number of image patches of the i-th patient. With a slight abuse of notations, we also denote $\mathcal{D} = \bigcup_{i=1}^{N_p} \mathcal{D}_i$ as the whole set of images of all patients. Then, a basic concept of **image-level** adjacent databases can be defined as: two databases are adjacent if they differ in a single image-label pair [1]. This concept is widely used for image-level privacy protection.

However, such image-level privacy protection is insufficient for our tasks. Instead, we introduce a concept of **patient-level** adjacent databases defined as follows:

Define 1 ***(Patient-level adjacent databases)*** *$\mathcal{D}'$ and $\mathcal{D}''$ are adjacent: if $\mathcal{D}'$ can be obtained by adding all images of a single patient to $\mathcal{D}''$ or removing all images of a single patient from $\mathcal{D}''$.*

This definition is inspired by the prior works [25, 11], in which the authors focus on user-level privacy. With the definition of adjacent databases, we can formally define the patient-level differential privacy as:

Define 2 ***(Differential privacy)*** *A randomized algorithm $\mathcal{A} : D \rightarrow R$ satisfies (ϵ, δ)-differential privacy if for any two adjacent databases $\mathcal{D}', \mathcal{D}'' \subseteq D$ and for any subset of outputs $S \subseteq R$ it holds:*

$$\mathbf{Pr}[\mathcal{A}(\mathcal{D}') \in S] \leq e^{\epsilon}\mathbf{Pr}[\mathcal{A}(\mathcal{D}'') \in S] + \delta \tag{1}$$

The randomized algorithm $\mathcal{A}$ is also known as the mechanism in the literature [5]. In our setting, $\mathcal{A}$ is the algorithm used to train deep CNNs, *e.g.*, the SGD algorithm. D denotes the training dataset (*i.e.*, $\mathcal{D}$ in our case) and R is the parameter space of a deep CNN. Intuitively, the Equation 1 indicates that participation of one individual patient in a training phase has a negligible effect on the final weight parameters. Another concept is the sensitivity of a randomized algorithm:

Define 3 ***(Sensitivity)*** *The sensitivity of a randomized algorithm $\mathcal{A}$ is the upper-bound of $||\mathcal{A}(\mathcal{D}') - \mathcal{A}(\mathcal{D}'')||_2$, where $\mathcal{D}'$ and $\mathcal{D}''$ are any adjacent databases (see in Define 1).*

To establish a randomized algorithm that satisfies differential privacy, we need to bound its sensitivity. The most used strategy is to clip the norm of the parameter update. In next two subsections, we will introduce the traditional SGD and P3SGD separately, as two instances of the randomized algorithm $\mathcal{A}$.

3.2. Standard SGD Algorithm

We start with the standard SGD (*i.e.*, non-private SGD) algorithm for training a deep CNN-based classification model. The goal of the classification is to train a CNN model $\mathbb{M} : \hat{y} = \mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$, where $\hat{y}$ is the predicted label, and $\boldsymbol{\theta}$ are the model parameters. Training of the model is to minimize the empirical loss $\mathcal{L}(\mathcal{D};\boldsymbol{\theta})$. In practice, we estimate the gradient of the empirical loss on a mini-batch. We denote the classification loss over a mini-batch as:

$$\mathcal{L}(\mathcal{B}_t;\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}_t|}\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{B}_t} l(\mathbf{f}(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}) \tag{2}$$

Here, $l(\mathbf{x},\mathbf{y})$ is the loss function, *e.g.*, cross-entropy loss. $\mathcal{B}_t$ refers to a mini-batch of images which are randomly and independently drawn from the whole image set $\mathcal{D}$. Note that we can add an additional regularization term into Equation 2, such as $l2$ term. At the t-th step of the SGD algorithm, we can update the current parameter $\boldsymbol{\theta}_t$ as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma_t \cdot \nabla_{\boldsymbol{\theta}_t}\mathcal{L}(\mathcal{B}_t;\boldsymbol{\theta}_t)$.

3.3. P3SGD Algorithm

Overall, our framework comprises of three components, which are *update computation*, *update sanitization*, and *privacy accumulation*. Our method inherits the computing paradigm of federated learning [24]. Moreover, to protect the privacy, we need to inject well-designed Gaussian noise into each step's update, which is marked as *update sanitization*. At last, we can use the moments accountant for *privacy accumulation*. The pseudo-code is depicted in Algorithm 1. Next, we will describe each of these components in details.

For *update computation*, at the beginning of the t-th step of P3SGD, we randomly sample a patient batch $\mathcal{B}_t$ from the database $\mathcal{D}$ with a sampling ratio p. Here, the notation $\mathcal{B}_t$ is different from the one in Equation 2, where the $\mathcal{B}_t$ is sampled from individual images instead of patients.

Then, for each patient i in the sampled batch, we perform a back propagation to calculate gradients of the parameters via images of the patient i. After that, we locally update the model using the computed gradients. After we traverse all images of this patient, we can obtain the model update with respect to patient i. This procedure can be interpreted as performing SGD on the local data from patient i.

In the next step, we average updates of all patients in $\mathcal{B}_t$ to obtain the final update at the t-th step. Note that we need to control the sensitivity of the total update for further *update sanitization*. In practice, this is implemented by clipping the $l2$ norm of the update, with respect to each individual patient (as shown in line 26 in Algorithm 1). $\mathrm{C_u}$ in Algorithm 1 denotes a predefined upper-bound. Thus, the sensitivity of the total update can be bounded by $2\mathrm{C_u}$ (a proof can be found in supplementary materials). The main idea of *update computation* is implemented by a function **PatientUpdate**, as shown in Algorithm 1.

To protect privacy, *update sanitization* needs to be performed. Specifically, we use Gaussian mechanism [7] to inject well-calibrated Gaussian noise into the original update, which leads to a noisy update. The variance of injected Gaussian noise is jointly determined by the upper-

Algorithm 1: P3SGD

```
1  Inputs:
2      Patient database: $\mathcal{D}$, Empirical Loss: $\mathcal{L}$.
3      Patient sampling ratio: $p$.
4      Noise scale set $\Omega_z$: $\{z_i\}_{i=1}^{N_z}$.
5      Noise budget $\epsilon'$ for selecting update per iteration.
6      Bound of update's norm: $\mathrm{C_u}$.
7      Bound of objective function's norm: $\mathrm{C_o}$.
8      Initialize $\boldsymbol{\theta}_0$ randomly
9  for $t \in [T]$ do
10     Take a subset $\mathcal{B}_t$ of patients with sampling ration $p$
11     for each patient $i \in \mathcal{B}_t$ do
12         $\Delta_t^i \leftarrow$ PatientUpdate$(i, \boldsymbol{\theta}_t)$
13     end
14     $\Delta_t \leftarrow \frac{1}{|\mathcal{B}_t|}(\sum_i \Delta_t^i)$
15     $\Omega_\sigma = \{\sigma = zC_u/|\mathcal{B}_t| :$ for $z$ in $\Omega_z\}$
16     $\Omega_\Delta = \{\widetilde{\Delta} = \Delta_t + \mathcal{N}(0,(\sigma^2\mathbf{I})) :$ for $\sigma$ in $\Omega_\sigma\}$
17     $\widetilde{\Delta}_t \leftarrow$ NoisyUpdateSelect$(\Omega_\Delta, \epsilon', \mathcal{B}_t, \boldsymbol{\theta}_t, \mathcal{L})$
18     $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \widetilde{\Delta}_t$
19 end
20 Function PatientUpdate($i$, $\boldsymbol{\theta}_t$):
21     $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_t$
22     for batch image samples $b$ from Patient $i$ do
23         $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma\nabla\mathcal{L}(b;\boldsymbol{\theta})$
24     end
25     $\Delta^i = \boldsymbol{\theta} - \boldsymbol{\theta}_t$
26     $\Delta^i = ClipNorm(\Delta^i, \mathrm{C_u})$
27     return $\Delta^i$
```

Algorithm 2: NoisyUpdateSelect

```
1  Function NoisyUpdateSelect($\Omega$, $\epsilon$, $\mathcal{B}$, $\boldsymbol{\theta}$, $\mathcal{L}$):
2      $\Omega_u = \{ u = -Clip(\mathcal{L}(\mathcal{B};\boldsymbol{\theta}+\Delta), \mathrm{C_o}) :$ for $\Delta$ in $\Omega \}$
3      Select $\Delta$ with probability $\frac{exp(\frac{\epsilon u}{2\mathrm{C_o}})}{\sum_{u\in\Omega_u} exp(\frac{\epsilon u}{2\mathrm{C_o}})}$
4      return $\Delta$
```

bound $\mathrm{C_u}$ of the update's $l2$ norm and the noise scale z. In this paper, we use a common strategy to set $\mathrm{C_u}$ as a globally fixed value similar to prior works [1, 25]. Therefore, the choice of a noise scale factor z is critical to train CNN model with high performance. Previous works [1, 25] usually use a fixed noise scale throughout the training phase. However, the fixed noise scale factor may lead to the departure of the noisy update from the descent direction or an ignorable regularization effect, because the magnitudes of the updates may vary at different iterative steps. Thus, we argue that the strategy that uses a fixed noise scale may hinder the classification performance.

In this paper, we present an elaborated strategy to adaptively select the noise scale. This strategy is originated from the exponential mechanism [7], which is a commonly used mechanism to build a differentially private version of the **Argmax** function. In this paper, the **Argmax** function refers to select the argument which maximizes a specific objective function. In our task, we use the negative loss function as the objective function, and the argument is the noisy update built upon different noise scales from the predefined set Ω_z. We implement this strategy as a function **NoisyUpdateSelect** depicted in Algorithm 2. The predefined set Ω_z contains N_z noise scale factors. Increasing N_z leads to more subtle control of the noisy update, which further boosts the performance. However, the increase of N_z also results in an increase of computational cost. Precisely, one more noise scale will bring about one more forward computation on all images in $\mathcal{B}_t$. In practice, we find that setting $N_z = 2$ suffices for our task. Note that setting $N_z = 1$ degenerates to the method used in [25, 11]. In the experiments, we show this strategy is crucial to boost the performance.

For *privacy accumulation*, the composition theorem can be leveraged to compose the privacy cost at each iterative step. In this paper, we make use of the moments accountant [1], which can obtain tighter bound than previous strong composition theorem [8]. Specifically, the moments accountant is to track a bound of the privacy loss random variable instead of a bound on the original privacy budget. Given a randomized algorithm $\mathcal{A}$, the privacy loss at output o is defined as:

$$c(o; \mathcal{A}, \mathbf{aux}, \mathcal{D}^{'}, \mathcal{D}^{''}) \triangleq \log \frac{\mathbf{Pr}[\mathcal{A}(\mathbf{aux}, \mathcal{D}^{'}) = o]}{\mathbf{Pr}[\mathcal{A}(\mathbf{aux}, \mathcal{D}^{''}) = o]} \quad (3)$$

Then, the privacy loss random variable $C(\mathcal{A}, \mathbf{aux}, \mathcal{D}^{'}, \mathcal{D}^{''})$ is defined by evaluating the privacy loss at the outcome sampled from $\mathcal{A}(\mathcal{D}')$ [28]. Here, $\mathcal{D}^{'}$ and $\mathcal{D}^{''}$ are adjacent. **aux** denotes the auxiliary information. In our P3SGD algorithm, auxiliary information at step t is the weight parameters $\boldsymbol{\theta}_{t-1}$ at the step $t-1$. The algorithm $\mathcal{A}$ is also known as the adaptive mechanism in literature [1]. We can then define the moments accountant as follows:

$$M_c(\lambda) \triangleq \max_{\mathbf{aux}, \mathcal{D}', \mathcal{D}''} M_c(\lambda; \mathbf{aux}, \mathcal{D}^{'}, \mathcal{D}^{''}) \quad (4)$$

where $M_c(\lambda; \mathbf{aux}, \mathcal{D}^{'}, \mathcal{D}^{''})$ is the moment generating function of the privacy loss random variable, which is calculated as:

$$M_c(\lambda; \mathbf{aux}, \mathcal{D}^{'}, \mathcal{D}^{''}) \triangleq \log \mathbb{E}[\exp(\lambda C(\mathcal{A}, \mathbf{aux}, \mathcal{D}^{'}, \mathcal{D}^{''}))] \quad (5)$$

Then, we introduce the composability and the tail bound of moments accountant as:

Theorem 1 ***(Composability)*** *Suppose that a randomized algorithm $\mathcal{A}$ consists of a sequence of adaptive mechanisms $\mathcal{A}_1, \ldots, \mathcal{A}_k$ where $\mathcal{A}_i : \prod_{j=1}^{i-1} \mathcal{R}_j \times \mathcal{D}^{'} \rightarrow \mathcal{R}_i$. The moments accountant of $\mathcal{A}_i$ is denoted as $M_c^i(\lambda)$. For any λ:*

$$M_c(\lambda) \leq \sum_{i=1}^{k} M_c^i(\lambda) \quad (6)$$

Theorem 2 ***(Tail bound)*** *For any $\epsilon \geq 0$, the algorithm $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy for*

$$\delta = \min_{\lambda} \exp(M_c(\lambda) - \lambda\epsilon) \quad (7)$$

Theorem 2 indicates that if the moments accountant of a randomized algorithm $\mathcal{A}$ is bounded, then $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy. The bound of the moments accountant for our strategy implemented in Algorithm 2 is guaranteed by the following theorem:

Theorem 3 *Given λ, the moments accountant of Algorithm 2 is bounded by $q \cdot \dfrac{\lambda(\lambda+1)\epsilon^2}{2}$.*

The proof can be done using the privacy amplification [18] and the theorem in the prior literature [2]. More details can be found in appendix of this work.

Privacy guarantee: In this paper, *privacy accumulation* is to accumulate the moments accountant's bound at each step. Note that privacy accumulation needs to be performed at the noisy update selection (line 17 in Algorithm 1) and the model update via noisy update (line 18 in Algorithm 1). For the **NoisyUpdateSelect** in line 17, we can calculate a bound via Theorem 3. For the model update in line 18, the bound is obtained based on the property of Gaussian Mechanism (Lemma 3 in appendix of [1]). Once we bound the moments accountant at each iterative step, we can compose these bounds using Theorem 1. At last, the total privacy cost is obtained based on Theorem 2. It suffices to compute the $M_c(\lambda)$ when $\lambda \leq 32$. In practice, we use a finite set $\{1, \cdots, 32\}$ following prior work [1].

4. Experimental Results

4.1. Experimental Settings

In this section, we verify the effectiveness of P3SGD on a real-world clinical dataset. This dataset is collected by the doctors in our team. The dataset consists of 1216 patients and each patient contains around 50 image patches. The task we consider in this paper is glomerulus classification, which aims to classify whether an image patch contains a glomerulus or not. This task has also been studied in a recent work [10]. We ask the doctors to manually label the image patches. For a fair comparison, we set the

Model	Type	# Params	SGD			SGD+Dropout			P3SGD		
			Training	Testing	Gap	Training	Testing	Gap	Training	Testing	Gap
AlexNet	**T**	60.9 M	99.87	91.58	8.29	98.97	**93.13**	5.84	96.85	92.74	4.11
VGG-16	**T**	14.7 M	99.81	92.19	7.81	99.28	**94.32**	4.96	96.23	93.87	2.36
ResNet-18	**M**	11.2 M	99.85	92.25	7.60	99.63	92.12	7.51	95.70	**95.23**	0.47
ResNet-34	**M**	21.3 M	99.23	93.19	6.04	99.16	93.22	5.94	95.80	**95.34**	0.46
MobileNet	**M**	3.2 M	98.73	92.01	6.72	98.65	91.61	7.04	94.79	**94.13**	0.66
MobileNet v2	**M**	2.3 M	98.52	93.24	5.28	98.37	93.28	5.09	95.32	**94.86**	0.46

Table 1. Training and testing accuracies (%) of various network architectures trained with different strategies. The gap between training and testing accuracies is used for measuring the overfitting of the CNN models.The type **T/M** denotes traditional/modern CNNs.

weight decay to 1e-4 and use data augmentation in all experiments. Specifically, we perform data augmentation by (1) randomly flipping input images vertically and/or horizontally, and (2) performing random color jittering, including changing the brightness and saturation of input images. All input images are resized into 224×224, and pixel intensity values are normalized into $[0, 1]$. All 1216 patients in the dataset are randomly split into a training dataset (1000 patients) and a testing dataset (216 patients).

4.2. Classification Evaluation

To validate the superiority of P3SGD in reducing overfitting, we compare it with the standard SGD (without Dropout). We also provide comparisons with the strategy that combines the standard SGD with Dropout. As a result, there are three training strategies: SGD, SGD+Dropout, and P3SGD.

We first evaluate our method on the ResNet-18 architecture [14]. For the standard SGD with Dropout, we insert Dropout between convolutional layers and set the drop ratio to 0.3 following [47]. To provide a reasonable weight initialization, we firstly pre-train the CNN model on a publicly available pathological image dataset[1]. The pre-training does not take an extra privacy cost, since we do not interact with the original training dataset in this stage. The pre-training can also help us to determine the hyper-parameters in Algorithm 1. For P3SGD, we set the total updating rounds T to 100 and set the noise scale ϵ^2 to 0.1 for selecting noisy update. The sampling ratio p is set to 0.1 and Ω_z is set to be $\{3.0, 1.0\}$. The C_u and C_o are set to 5.0 and 3.0, respectively. To facilitate the discussion, we denote `SGD` and `P3SGD` as the models trained using SGD (without Dropout) and P3SGD (we use the abbreviations in the following discussions).

From the results of ResNet-18 (Table 1), we observe that `SGD` obviously overfits (it even reaches nearly 100% training accuracy). In contrast, `P3SGD` drastically decreases the gap between training and testing accuracies and improves the testing accuracy. In particular, `P3SGD` outperforms `SGD` by 2.98% in the testing accuracy (a 38.5% relative drop in classification error), while the gap is decreased from 7.60% to 0.47%, which shows a 93.8% relative improvement. These results indicate that P3SGD significantly reduces the overfitting in the ResNet-18 model compared with the standard SGD. We also plot the loss curves of ResNet-18 in Figure 2, which further demonstrates the regularization effect of P3SGD. Besides, **there is no significant performance improvement when we apply Dropout on ResNet-18**. Dropout even leads to a slight decrease (from 92.25% to 92.12%) in testing accuracy. We will discuss this phenomenon in details in Section 4.4.

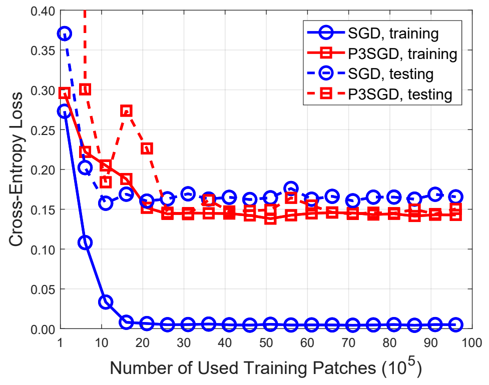

Figure 2. The training and testing loss curves of ResNet-18 with different training strategies. P3SGD significantly reduces the overfitting compared to SGD.

Besides the ResNet-18, we also conduct extensive experiments on other popular CNN architectures. In general, we mainly test on two types of CNN models, namely, traditional CNNs and modern CNNs [2] (denoted by **T** and **M** in Table 1). Specifically, six architectures are included: AlexNet [21], VGG-16 [37], ResNet-18 [14], ResNet-34 [14], MobileNet [15], and MobileNet v2 [34]. For traditional CNNs (*e.g.*, AlexNet), we insert Dropout between fully connected (FC) layers and set the drop ratio to 0.5 following [39]. The results are summarized in Table 1. On the one hand, our method consistently boosts the testing accuracy over the standard SGD (without Dropout) on various

[1]http://www.andrewjanowczyk.com/use-case-4-lymphocyte-detection/

[2]The modern CNN consists of convolutional layers except the final prediction layer, which comprises of a global average pooling and a fully-connected layer.

CNN architectures. The ResNet-34 trained with P3SGD achieves the highest testing accuracy at 95.34% among all network architectures and training strategies. In particular, **P3SGD outperforms Dropout technique on all modern CNNs**, *e.g.*, the testing accuracy gain is 2.12% in the case of ResNet-34. On the other hand, the training accuracy is suppressed when we use P3SGD to train the CNN model, which further leads to a decrease of the gap between training and testing accuracy.

Despite the superiority of our method, we observe that **Dropout is usually more effective than P3SGD on the traditional CNNs**, *e.g.*, it obtains a slight accuracy gain of 0.45% in the case of VGG-16 compared to P3SGD. We provide some interpretations in the discussion part. We also notice that, under the standard SGD (without Dropout) training strategy, the modern CNNs have less overfitting (measured as the gap between training and testing accuracies) than traditional CNNs. This may be caused by the regularization effect brought by the Batch Normalization [16] which exists in the modern CNNs.

4.3. Privacy Cost Analysis

Another advantage of P3SGD is to provide patient-level privacy within differential privacy. The differentially private degree is measured by (ϵ, δ) (*i.e*, privacy cost) in Equation 1. In this part, we calculate the total spend of privacy cost via the moments accountant theorem. The target δ is fixed to $\frac{1}{|N_p|^{1.1}}$ (N_p is the number of patients in the training set), which is suggested by the previous literature [5]. In our task, the δ is around $5e-4$ ($N_p = 1000$). To verify the effectiveness of our proposed strategy for dynamically controlling the noisy update, we compare it with the strategy of fixed noise scale (marked by ✗ in Table 2) which is adopted by the state-of-the-art works [11, 25]. For simplicity, we use `adaptive` and `fixed` to denote these two strategies. All the experiments are performed on ResNet-18.

We test on various noise scale sets Ω_z to show how the noise scale affects the performance. We find that the noise scale greater than 3.0 leads to unstable training. In practice, we build Ω_z using the noise scale from $\{1.0, 2.0, 3.0\}$. Overall, P3SGD with the `adaptive` strategy ($\Omega_z = \{3.0, 1.0\}$) achieves the best testing accuracy of 95.23% at a privacy cost of 6.97. For the `fixed` strategy, a larger noise scale leads to a lower privacy cost, however, it may cause the noisy update deviating from the decent direction and further hinders the testing accuracy. For example, setting Ω_z to $\{3.0\}$ leads to the lowest privacy cost of 4.70 and the worst accuracy of 92.15%, while setting $\Omega_z = \{1.0\}$ achieves a better accuracy of 94.38% but a much higher privacy cost of 8.48. The `adaptive` strategy provides a reasonable solution for this dilemma of the `fixed` strategy.

Adaptive	Ω_z	Testing	ϵ
✓	$\{3.0, 1.0\}$	**95.23**	6.97
✓	$\{2.0, 1.0\}$	94.31	7.10
✓	$\{3.0, 2.0\}$	93.57	4.97
✗	$\{1.0\}$	94.38	8.48
✗	$\{2.0\}$	93.24	5.13
✗	$\{3.0\}$	92.15	4.70

Table 2. Testing accuracy (%) and privacy cost of different settings on ResNet-18. The *Adaptive* column indicates if an adaptive noise scale set (marked by a ✓) or a fixed scale (marked by a ✗) is used. Setting a fixed noise scale leads to the method in [11, 25]

Strategy	Training	Testing	Gap
SGD+Dropout	99.63	92.12	7.51
SGD+DropBlock	98.85	94.87	3.68
P3SGD	95.70	**95.23**	0.47

Table 3. Training and testing accuracies (%) on ResNet-18 with different regularization strategies.

In general, the `adaptive` strategy leads to a better trade off between the privacy cost and the testing accuracy. Specifically, extending the fixed scale $\{z_1\}$ or $\{z_2\}$ to $\{z_1, z_2\}$ achieves the testing accuracy higher than or approaching to the best testing accuracy among the corresponding `fixed` strategies, while with a reasonable privacy cost. For instance, the `adaptive` strategy with $\{3.0, 1.0\}$ achieves the accuracy of 95.23%, which is higher than the `fixed` strategy with either $\{3.0\}$ or $\{1.0\}$. Our strategy also outperforms a naive solution by setting the noise scale to the average of 1.0 and 3.0 (*i.e.*, $\{2.0\}$). There is even an accuracy gain of 0.85% by extending $\{1.0\}$ to $\{3.0, 1.0\}$. We infer this accuracy gain comes from the stronger regularization effect brought by the larger noise scale. Meanwhile, the `adaptive` strategy ($\{3.0, 1.0\}$) achieves a moderate privacy cost between the costs obtained by the corresponding `fixed` strategies (setting Ω_z to $\{1.0\}$ or $\{3.0\}$).

To conclude, our proposed strategy can be seen as a simplified version of line search in numerical optimization [27], and provides a more careful way to control the magnitude of the added noise. The effectiveness of our strategy comes from the fine-grained way to control the noisy update.

4.4. Discussions

In this subsection, we first analyze the performance of different types of CNNs. We then compare P3SGD with the state-of-the-art regularization mechanism. Finally, we show that the model trained with P3SGD is resistant to a model-inversion attack.

Network Architecture. As shown in Table 1, Dropout and our method P3SGD demonstrate totally different effects on the two types of CNN architectures (traditional CNNs and modern CNNs). Specifically, our method outperforms Dropout on modern CNNs, instead, Dropout is more ef-

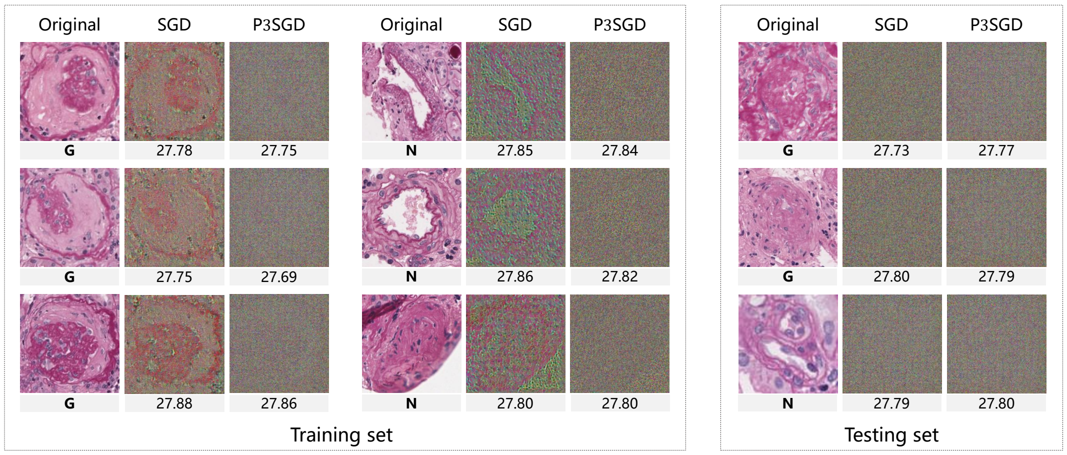

Figure 3. Visualization of the model-inversion attack. **G/N** below each original patch denotes if the patch contains a glomerulus or not. The number below each reconstructed image is the PSNR value. The reconstructed examples of the training dataset are demonstrated in the left part. For comparison, we also show some examples from the testing dataset in the right part.

fective on the traditional CNN architectures. This may be caused by following reasons: (1) Dropout is originally designed for the FC layers due to its huge numbers of parameters (*e.g.*, around 90% parameters of VGG-16 are from the FC layers). However, there is only one FC layer with a few parameters in modern CNN architectures. (2) The cooperation of Dropout and Batch Normalization can be problematic [16]. As we know, batch normalization layer widely exists in modern CNNs (*e.g*, ResNet [14]). (3) Dropout discards features randomly, however, the features extracted by convolutional layers are always spatially correlated, which impedes the use of Dropout on convolutional layers. Some recent works propose to modify Dropout for convolutional layers. We compare our method with a variant of Dropout in the next part.

Other Regularization Techniques. From the previous discussion, some advanced forms of Dropout should be adopted in the modern CNN. In this part, we compare our method with a recent technique, named DropBlock [12], on ResNet-18. For a fair comparison with Dropout, we insert DropBlock between every two convolutional layers and set the drop ratio to 0.3 following [12]. The results are shown in Table 3. DropBlock achieves a testing accuracy gain of 2.75% against Dropout, while P3SGD outperforms both Dropout and DropBlock. In contrast to P3SGD, DropBlock has no suppression effect on the training accuracy. We guess that the performance gain of DropBlock comes from the effect of the implicit model ensemble. We further combine P3SGD with DropBlock but do not obtain obvious accuracy boost.

Model-inversion Attack. To demonstrate that `P3SGD` is resistant to the model-inversion attack [24, 23], we perform an inversion attack on CNN models trained with different strategies. As a case study, we conduct experiments on the ResNet-18 and use the output features from the 3-th residual block to reconstruct the input image (see details in the appendix). Some visualizations are shown in Figure 3. We can reconstruct the outline of the tissue in the input image using the features from the `SGD`. In contrast, we can not obtain any valuable information from `P3SGD` (*i.e.*, the model is oblivious to training samples). It indicates that `SGD` is more vulnerable than `P3SGD`. Quantitatively, we perform attack on all the training images and report the average PSNR values as: 27.82 for `P3SGD` and 27.84 for `SGD`. We also conduct the same study on patches from the testing dataset and show some examples in the left part in Figure 3. The results show that it is hard to reconstruct the input image for both `SGD` and `P3SGD`, since the testing examples are not touched by the model in the training phase. This provides some cues for the memorization ability of CNNs [48].

5. Conclusion

In this paper, we introduce a novel SGD schema, named P3SGD, to regularize the training of deep CNNs while provide rigorous privacy protection within differential privacy. P3SGD consistently outperforms SGD on various CNN architectures. **The key technical innovation lies in the strategy that adaptively controls the noisy update**. We conduct an analysis and show the effectiveness of this strategy. We also perform a model-inversion attack and show that the model trained with P3SGD is resistant to such an attack.

This research paves a new way to regularize deep CNNs on pathological image analysis with an extra advantage of appealing patient-level privacy protection. Applying this method to other types of medical image analysis tasks is promising and implies a wide range of clinical applications.

Acknowledgment. Bingzhe Wu and Guangyu Sun are supported by National Natural Science Foundation of China (No.61572045).

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS*, 2016. 2, 3, 4, 5

[2] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, 2016. 5

[3] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *NIPS*, 2009. 3

[4] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *JMLR*, 12:1069–1109, July 2011. 3

[5] Cynthia Dwork. Differential privacy. In *ICALP*, 2006. 2, 3, 7

[6] Cynthia Dwork. Differential privacy: A survey of results. In *TAMC*, 2008. 2

[7] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9:211–407, Aug. 2014. 4, 5

[8] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *FCOS*, 2010. 5

[9] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *CCS*, 2015. 1, 2

[10] J. Gallego, A. Pedraza, S. Lopez, G. Steiner, L. Gonzalez, A. Laurinavicius, and G. Bueno. Glomerulus classification and detection based on convolutional neural networks. *Journal of Imaging*, 4, 2018. 1, 5

[11] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017. 2, 3, 5, 7

[12] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *arXiv preprint arXiv:1810.12890*, 2018. 2, 8

[13] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML*, 2016. 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6, 8

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 6

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *arXiv preprint arXiv:1502.03167*, 2015. 7, 8

[17] A. Janowczyk and A. Madabhushi. Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases. *JPI*, 7(1):29, 2016. 1

[18] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011. 5

[19] Daniel Kifer, Adam Smith, Abhradeep Thakurta, Shie Mannor, Nathan Srebro, and Robert C Williamson. Private convex empirical risk minimization and high-dimensional regression. In *COLT*, 2012. 3

[20] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, 2009. 2

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 6

[22] Yun Liu, Krishna Kumar Gadepalli, Mohammad Norouzi, George Dahl, Timo Kohlberger, Subhashini Venugopalan, Aleksey S Boyko, Aleksei Timofeev, Philip Q Nelson, Greg Corrado, Jason Hipp, Lily Peng, and Martin Stumpe. Detecting cancer metastases on gigapixel pathology images. Technical report, arXiv, 2017. 1

[23] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015. 8

[24] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017. 3, 4, 8

[25] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *ICLR*, 2018. 2, 3, 4, 5, 7

[26] P. Naylor, M. La, F. Reyal, and T. Walter. Nuclei segmentation in histopathology images using deep neural networks. In *ISBI*, 2017. 1

[27] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, 2006. 7

[28] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*, 2017. 3, 5

[29] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. In *ICLR*, 2018. 3

[30] A. Pedraza, J. Gallego, S. Lopez, L. Gonzalez, A. Laurinavicius, and G. Bueno. Glomerulus classification with convolutional neural networks. In *MIUA*, 2017. 1

[31] NhatHai Phan, Yue Wang, Xintao Wu, and Dejing Dou. Differential privacy preservation for deep auto-encoders: An application of human behavior prediction. In *AAAI*, 2016. 2, 3

[32] Pau Rodríguez, Jordi Gonzàlez, Guillem Cucurull, Josep M. Gonfaus, and F. Xavier Roca. Regularizing cnns with locally constrained decorrelations. In *ICLR*, 2017. 2

[33] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Déja vu: an empirical evaluation of the memorization properties of convnets. *arXiv preprint arXiv:1809.06396*, 2018. 1, 2, 3

[34] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *arXiv preprint arXiv:1801.04381*, 2018. 6

[35] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *CCS*, 2015. 3

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 6

[38] K. Sirinukunwattana, S. E. A. Raza, Y. Tsang, D. R. J. Snead, I. A. Cree, and N. M. Rajpoot. Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images. *IEEE Transactions on Medical Imaging*, 35(5):1196–1206, May 2016. 1

[39] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15:1929–1958, 2014. 1, 2, 6

[40] V. Sze, Y. Chen, T. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, Dec 2017. 1

[41] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *CVPR*, 2015. 2

[42] Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013. 1, 2

[43] Di Wang, Minwei Ye, and Jinhui Xu. Differentially private empirical risk minimization revisited: Faster and more general. In *NIPS*. 2017. 3

[44] Bingzhe Wu, Zhichao Liu, Zhihang Yuan, Guangyu Sun, and Charles Wu. Reducing overfitting in deep convolutional neural networks using redundancy regularizer. In *ICANN*, 2017. 2

[45] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. DisturbLabel: Regularizing CNN on the Loss Layer. In *CVPR*, 2016. 2

[46] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *CSF*, 2018. 1, 2, 3

[47] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 6

[48] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2016. 1, 8

[49] Jiaqi Zhang, Kai Zheng, Wenlong Mou, and Liwei Wang. Efficient private ERM for smooth objectives. In *IJCAI*, 2017. 3

[50] Y. Zhou, H. Chang, K. E. Barner, and B. Parvin. Nuclei segmentation via sparsity constrained convolutional regression. In *ISBI*, 2015. 1