Not Just Privacy: Improving Performance of Private Deep Learning in Mobile Cloud

Ji Wang*
College of Systems Engineering
National University of Defense Technology
Changsha, P.R. China
wangji@nudt.edu.cn

Jianguo Zhang*
Department of Computer Science
University of Illinois at Chicago
Chicago, USA
jzhan51@uic.edu

Weidong Bao
College of Systems Engineering
National University of Defense Technology
Changsha, P.R. China
wdbao@nudt.edu.cn

Xiaomin Zhu
College of Systems Engineering
State Key Laboratory of High Performance Computing
National University of Defense Technology
Changsha, P.R. China
xmzhu@nudt.edu.cn

Bokai Cao
Facebook Inc.
Menlo Park, USA
caobokai@fb.com

Philip S. Yu
Department of Computer Science
University of Illinois at Chicago
Chicago, USA
Institute for Data Science
Tsinghua University
Beijing, P.R. China
psyu@uic.edu

ABSTRACT

The increasing demand for on-device deep learning services calls for a highly efficient manner to deploy deep neural networks (DNNs) on mobile devices with limited capacity. The cloud-based solution is a promising approach to enabling deep learning applications on mobile devices where the large portions of a DNN are offloaded to the cloud. However, revealing data to the cloud leads to potential privacy risk. To benefit from the cloud data center without the privacy risk, we design, evaluate, and implement a cloud-based framework Arden which partitions the DNN across mobile devices and cloud data centers. A simple data transformation is performed on the mobile device, while the resource-hungry training and the complex inference rely on the cloud data center. To protect the sensitive information, a lightweight privacy-preserving mechanism consisting of arbitrary data nullification and random noise addition is introduced, which provides strong privacy guarantee. A rigorous privacy budget analysis is given. Nonetheless, the private perturbation to the original data inevitably has a negative impact on the performance of further inference on the cloud side. To mitigate this influence, we propose a noisy training method to enhance the cloud-side network robustness to perturbed data. Through the sophisticated design, Arden can not only preserve privacy but also improve the inference performance. To validate the proposed Arden, a series of experiments based on three image datasets and a real mobile application are conducted. The experimental results demonstrate the effectiveness of Arden. Finally, we implement Arden on a demo system to verify its practicality.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**; • **Computing methodologies → Machine learning**; • **Computer systems organization → Distributed architectures**;

KEYWORDS

deep learning; differential privacy; mobile cloud

ACM Reference Format:
Ji Wang, Jianguo Zhang, Weidong Bao, Xiaomin Zhu, Bokai Cao, and Philip S. Yu. 2018. Not Just Privacy: Improving Performance of Private Deep Learning in Mobile Cloud. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220106

1 INTRODUCTION

Recent years have seen impressive successes of deep neural networks in various domains. DNNs demonstrate superb capabilities in discovering high-dimensional structures from large volume of data. Meanwhile, mobile devices such as smartphones, medical tools, and Internet of Things (IoT) devices have become nearly ubiquitous. There is a high demand for the on-device machine learning services, including object recognition, language translation, health monitoring, and many others [6, 15, 28, 41]. Encouraged by the outstanding performance of DNNs in these services, people naturally attempt to push deep learning to mobile devices [23].

The unprecedented advances of deep learning are enabled, in part, by the ability of very large deep neural networks. For example, in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) from 2010 to 2015, the continuously decreasing error rates achieved by the state-of-the-art DNNs are accompanied by their increasing complexity as shown in Fig. 1. There are millions to hundreds of

*Both authors contribute equally to the paper.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220106

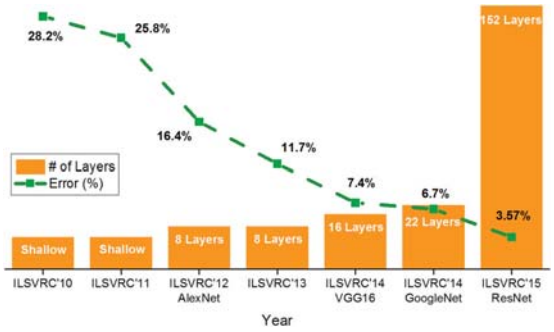


Figure 1: The performance and size of the DNNs in ILSVRC'10-15 [16, 21, 40, 42].

millions of parameters in these advanced DNNs which require significant processing and storage resource. However, the miniature nature of mobile devices imposes the intrinsic capacity bottleneck that makes resource-hungry applications remain off bounds. Particularly, the large DNN models exceed the limited on-chip memory of mobile devices, and thus they have to be accommodated by the off-chip memory, which consumes significantly more energy [7, 14]. What is worse, the enormous dot products aggravate the burden of processing units. Running a deep learning application can easily dominate the whole system energy consumption and drain a mobile device's battery. Up to now, it is still difficult to deploy powerful deep learning applications on mobile systems [27].

In order to enable on-device deep learning applications, academia and industry put forward two solutions: (1) compressing large DNNs, and (2) partitioning large DNNs across mobile devices and cloud data centers. The former one tries to prune and accelerate DNNs to fit deep learning applications into mobile devices. But this complicated technique suffers from performance degradation [7] and uncontrolled energy consumption [45]. For the latter solution, the shallow portions of a DNN are deployed on mobile devices while the complex and large parts are offloaded to the cloud data center. A fast transformation is first carried out on the input data locally. Then, the transformed data are revealed to the cloud data center for the time and energy consuming inferences. The experiment in [24] shows that the total energy consumption can be reduced by approximately 10% when a larger proportion of the DNN inferences are conducted in the cloud. Apart from the resource and energy consideration, the cloud-based solution is appealing for deep learning service providers. Because of the intellectual property, the deep learning service providers are not always willing to share their valuable and highly tuned DNNs [35]. Deploying large parts of a DNN in the centralized data center provides a viable alternative to protect the service providers' intellectual property. The cloud-based solution paves a promising way to enable deep learning applications on mobile devices.

Despite the benefits brought by the cloud-based solution, it presents obvious privacy issues with transmitting data from mobile devices to the cloud data centers. Once an individual reveals its data to cloud data centers for further inferences, it is almost impossible for the individual to control the usage of the data. Hence, it is a critical requirement to lessen the privacy risk when implementing the cloud-based deep learning applications. Unfortunately, this is a challenging and tough problem for a number of reasons:

- **Guarantee the privacy**. Although the existing literature claimed that revealing the transformed data rather than the raw data could provide a satisfactory privacy protection [29, 43], there still exist the threats of information disclosure [26]. Therefore, a privacy-preserving mechanism that provides a rigorous and strong privacy guarantee should be designed.
- **Minimize the overhead**. It is to relieve the burden of mobile devices that we resort to the cloud-based solutions. It contradicts the original purpose to cause too much overhead when providing privacy protection. Hence, we should minimize the overhead of the privacy-preserving mechanism to provide lightweight property.
- **Improve the performance**. In order to preserve privacy, the data released to the cloud data center is distorted and perturbed, which definitely would lead to a degradation of the inference performance, *e.g.*, classification accuracy. An effective method is needed to alleviate degradation and yield a good inference result while still preserving privacy.

To address the above problems, we take both privacy and performance into consideration, and propose a priv**A**te infe**R**ence framework based on **D**eep n**E**ural **N**etworks in mobile cloud, named ARDEN. ARDEN partitions a deep neural network across the mobile device and the cloud data center. On the mobile device side[1], the raw data is transformed by the shallow portions of the DNN to extract their lower-level features. The local neural networks are derived from the pretrained neural networks to avoid local training, and can be regarded as a feature extractor for different inference tasks based on the idea of transfer learning [44]. In order to preserve privacy, we introduce the differential privacy mechanism [10] on the local side, which adds deliberate noise into the data before uploading. The large portions of the DNN are deployed in the cloud data center to run the complex and resource-hungry inference tasks. We propose a noisy training method to make the DNN robust to the additional noise in the data revealed by mobile devices, and so to improve the inference performance. Our main contributions are listed as follows:

- **A framework enabling deep learning on mobile devices**. We take the privacy, performance, and overhead into consideration, and design a framework to partition the very large deep neural networks in mobile cloud environment. All the resource-hungry tasks, *e.g.*, network training and complex inference are offloaded to the cloud data centers. The works in the cloud data center are transparent to end mobile devices, which enables the online model upgrade.
- **A differentially private local transformation mechanism**. To preserve privacy, we propose a new mechanism to perturb the local data transformation based on the differential privacy mechanism. Compared with the existing ones, our proposed mechanism can be customized to protect specific data items, and fits well with the stacking structure of neural networks. The corresponding privacy budget is analyzed theoretically to provide rigorous privacy guarantee.

[1]Notice that the terms mobile device side, local side, and end side are used interchangeably in this paper.

- **A noisy training approach for performance improvement**. We propose a noisy training approach that contains a generative model to inject delicate noisy samples into the training set to increase the robustness of cloud-side DNNs to the perturbed data. By this method, the negative effect incurred by the end-side noise on the inference tasks is largely alleviated.
- **Thorough empirical evaluation**. We evaluate the proposed framework on standard image classification tasks and a real mobile application. Further, ARDEN is deployed on a real demo system to test its overhead. Our code is open-sourced at GitHub[2].

2 PRELIMINARY

In this section, we briefly introduce the basic knowledge of deep learning, including deep neural networks, stochastic gradient descent algorithm, and transfer learning. Then, we revisit the definition of differential privacy.

2.1 Deep Learning

Deep learning, which revolutionizes many machine learning tasks, transforms inputs to desired outputs by feed-forward neural networks comprising many layers of basic units like affine transformations and nonlinear activation functions. The essential idea of stacking multiple layers is to extract complex representations from high-dimensional inputs progressively. A typical neural network with two hidden layers is shown in Fig. 2(a). Each node in the network is a neuron that takes a weighted sum of the outputs of the prior layer, and then expose the sum to the next layer through a nonlinear activation function.

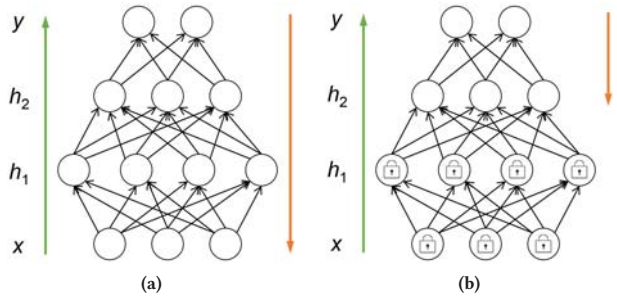


Figure 2: Illustrations of deep neural networks and transfer learning. The green and the orange arrows represent the feed-forward and the back-propagation procedures, respectively.

The main work of training a DNN is to automatically learn the parameters that minimize the loss function of the DNN from large volume of training data. It is usually done by the mini-batch stochastic gradient descent (SGD) algorithm and its variants [17, 47]. At each step, the SGD first calculates the loss function through the feed-forward procedure. Given the batch of random training samples $\boldsymbol{x}_N = \{x_1, ..., x_{|N|}\}$ and the parameters $\boldsymbol{\omega}$, the feed-forward procedure computes the output of the DNN sequentially, and then calculates the loss for each training sample $J(\boldsymbol{\omega}; x_i)$ that depicts the difference between the feedforward output and the ground truth. After that, the backpropagation procedure calculates the partial derivative of $J(\boldsymbol{\omega}; x_i)$ with respect to each parameter ω_j in $\boldsymbol{\omega}$, and then takes an average value over the batch, i.e., $g_j = \frac{1}{|N|}\sum_i \nabla_{\omega_j} J(\boldsymbol{\omega}; x_i)$. The update rule for ω_j is:

$$\omega_j = \omega_j - \alpha g_j, \tag{1}$$

where α is the learning rate. One full iteration over all training samples is referred as an epoch.

As the stacking layers represent input data at a progressively higher level of abstraction, the shallow-layer representations appear to be general to diverse datasets or tasks, rather than specific to a particular one [44]. It is plausible to regard the shallow layers of a pretrained DNN on one dataset as a generic feature extractor that can be applied to other target tasks or datasets. This is the key idea of transfer learning in deep neural networks. More precisely, the usual transfer learning approach [34, 44] is to copy the first n layers of a pretrained DNN to the first n layers of a target DNN. Then, the remaining layers of the target DNN are retrained towards the task or dataset of interest while the transferred shallow layers are left frozen. One illustration of the transfer learning approach is shown in Fig. 2(b).

2.2 Differential Privacy

Differential privacy is a concept of privacy tailored to the privacy-preserving data analysis. It aims at providing provable privacy guarantee for sensitive data and is increasingly regarded as a standard notion for rigorous privacy [5]. Formally, the definition of ε-differential privacy is given as below:

Definition 2.1. [9] A randomized mechanism $\mathcal{A}$ is ε-differentially private, iff for any adjacent input d and d', and any output S of $\mathcal{A}$,

$$\Pr[\mathcal{A}(d) = S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(d') = S]. \tag{2}$$

Typically, the inputs d and d' are adjacent inputs when they differ by only one data item. The adjacent input is an application-specific notion. For example, a sentence is divided into several items for every 5 words. Two sentences are considered to be adjacent sentences if they differ by at most 5 consecutive words. The parameter ε is the privacy budget [9], which controls the privacy guarantee of the randomized mechanism $\mathcal{A}$. A smaller value of ε indicates a stronger privacy guarantee. According to this definition, a differentially private algorithm can provide aggregate representations about a set of data items without leaking information of any data item.

A general method for approximating a deterministic function f with ε-differential privacy is to add noise calibrated to the global sensitivity of f, denoted as Δf, which is the maximal value of $\|f(d) - f(d')\|$ among any pair of d and d'. For instance, the Laplacian mechanism is defined by,

$$\mathcal{A}_f(d) = f(d) + Lap(\frac{\Delta f}{\varepsilon}), \tag{3}$$

where $Lap(\frac{\Delta f}{\varepsilon})$ is a random variable sampled from the Laplace distribution with scale $\frac{\Delta f}{\varepsilon}$.

The differential privacy enjoys a superb property, the immunity to post-processing [11]. Any algorithm cannot compromise the

[2]https://github.com/jianguoz/Not-Just-Privacy

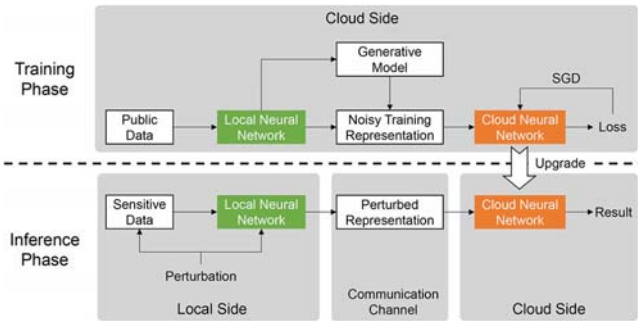


Figure 3: The overview of ARDEN.

differentially private output and make it less differentially private. That is, if a differential privacy mechanism protects the privacy of sensitive data item, then the privacy loss cannot be increased even by the most sophisticated algorithm.

3 THE PROPOSED FRAMEWORK

This section describes the proposed framework toward DNN-based private inference in mobile cloud. We first give the overview of ARDEN. Then, the two key techniques in ARDEN are detailed.

3.1 Overview

The overview of ARDEN is presented in Fig. 3. ARDEN relies on the mobile cloud environment and divides the DNN into the local-side part and the cloud-side part. The local neural network is derived from the pretrained DNN whose structure and weights are frozen. The cloud neural network is fine-tuned in the training phase. The whole training phase and the complex inference phase are performed in the cloud data centers. Mobile devices merely undertake the simple and lightweight feature extraction and perturbation.

In the inference phase, the sensitive data is transformed by the local neural network to extract the general features embedded in it. For preserving privacy, the transformation is perturbed by both nullification and random noise which are consistent with differential privacy. Then, the perturbed representations are transmitted to the cloud for further complex inference. Because the data to be transmitted are the abstract representation of the raw data, the size of the data to be transmitted is smaller than that of the raw data. The local initial transformation can reduce communication cost compared with transmitting raw data directly.

In the training phase, we use the public data of the same type as the sensitive data to train the cloud neural network. In order to improve the robustness of the cloud neural network to the noise, we propose a noisy training method where both raw training data and generative training data are fed into the network to tune the weights. The key component in noisy training is the generative model that generates sophisticated noisy training samples based on the public data. In addition, it is worth noting that the training phase and the inference phase can run in parallel once we get an initial cloud neural network. Benefitting from the transfer learning, we only need to train the cloud neural network for different datasets and tasks while keeping the local neural network frozen. It indicates that all the works on the cloud side are transparent to end mobile devices. The cloud neural network can be upgraded online without the service interruption to end users. This transparency property facilitates the deep learning service on one hand and on the other hand protects the intellectual property of service providers.

3.2 Differentially Private Transformation

To preserve privacy, the data transformation on the local side is perturbed. One of the key techniques in ARDEN is how to inject the perturbation that satisfies the differential privacy and measure the privacy budget of the perturbation.

Regarding the local neural network as a deterministic function $\boldsymbol{x}_r = \mathcal{M}(\boldsymbol{x}_s)$, where $\boldsymbol{x}_s$ represents sensitive input data, one intuitive attempt of providing ε-differential privacy is to add noise that conforms to the Laplace distribution with scale $\Delta\mathcal{M}/\varepsilon$ into the output $\boldsymbol{x}_r$. However, it is difficult to estimate the global sensitivity. An overly conservative estimation would add too much noise into the output representation, which destroys the utility of the representation for future inferences. Besides, this straightforward approach cannot be customized to provide personalized private requirements such as masking particular data items that are highly sensitive. Therefore, we propose a more sophisticated mechanism including nullification and layer-wise perturbation. The corresponding privacy budget analysis is given in detail.

Algorithm 1 outlines the differentially private data transformation on the local side. For each sensitive data $\boldsymbol{x}_s$, some data items are masked by the nullification operation. Then the data is fed into the local neural network for feature extraction. At a specific layer l, for the output of $\mathcal{M}_l$ (the neural network from the first layer to the l-th layer), we bound its infinity norm by B, and inject noise to protect privacy. At last, the perturbed final representation is generated by $\overline{\mathcal{M}}_l$ (the neural network after the l-th layer), and is transmitted to the cloud side. Next we discuss each operation in Algorithm 1 in detail and analyze the privacy budget.

Algorithm 1: Differentially Private Transformation

Input: Each sensitive data $\boldsymbol{x}_s$; Local network $\mathcal{M}(\cdot) = \overline{\mathcal{M}}_l(\mathcal{M}_l(\cdot))$.
Nullification matrix I_n; Noise scale σ; Bound threshold B; Injection layer l. $\boldsymbol{x}'_s \leftarrow \boldsymbol{x}_s \odot I_n$;
$\boldsymbol{x}_l \leftarrow \mathcal{M}_l(\boldsymbol{x}'_s)$;
$\boldsymbol{x}'_l \leftarrow \boldsymbol{x}_l / \max(1, \frac{\|\boldsymbol{x}_l\|_\infty}{B})$;
$\tilde{\boldsymbol{x}}'_l \leftarrow \boldsymbol{x}'_l + Lap(B/\sigma\mathbf{I})$;
$\tilde{\boldsymbol{x}}_r \leftarrow \overline{\mathcal{M}}_l(\tilde{\boldsymbol{x}}'_l)$;
Output: Perturbed representation $\tilde{\boldsymbol{x}}_r$.

Nullification: Given the input sensitive data $\boldsymbol{x}_s$ that consists of N data items, nullification performs item-wise multiplication of $\boldsymbol{x}_s$ with I_n, where I_n is a binary matrix constituted of 0 and 1 with the same dimensions as $\boldsymbol{x}_s$. I_n can be either specified by end users to nullify the highly sensitive data items or generated randomly. The number of zeros in I_n is determined by $\lceil N \cdot \mu \rceil$, where $\lceil \cdot \rceil$ is the ceiling function, and μ is the nullification rate. The zeros are located in I_n conforming to the uniform distribution. Apparently, a high value of μ has a negative impact on the inference performance, which will be examined in Section 4.

Norm bounding: It is difficult to estimate the global sensitivity of neural networks. Hence, for each sensitive data $\boldsymbol{x}_s$, we clip the max

value of the output of the injection layer within fixed bounds to estimate sensitivity, *i.e.*, the output $\boldsymbol{x}_l$ is bounded as $\boldsymbol{x}_l/\max(1, \frac{\|\boldsymbol{x}_l\|_\infty}{B})$. It indicates that $\boldsymbol{x}_l$ is preserved when $\|\boldsymbol{x}_l\|_\infty \leq B$, whereas it is scaled down to B when $\|\boldsymbol{x}_l\|_\infty > B$. Then, the global sensitivity can be estimated as $2B$. The bound threshold B is input-independent and so would leak no sensitive information. In practice, the value of B can be set as the median of the infinity norm of the original outputs over the training [1].

We add random noise sampled from the Laplace distribution into the bounded output $\boldsymbol{x}'_l$ to protect the privacy. Different from the existing works where the noise is added into the final output of the deterministic function, the noise is added during the transformation in Algorithm 1. This injection method is more flexible, and fits much better with the stacking structure of neural networks. One important issue of differentially private transformation is determining the privacy budget that represents a rigorous privacy guarantee provided by the designed mechanism. It is given as follows.

Theorem 3.1. *Given the sensitive data $\boldsymbol{x}_s$ and the local neural network $\mathcal{M}$, Algorithm 1 is ε-differentially private,*

$$\varepsilon = \ln[(1-\mu)e^{2\sigma/\Lambda} + \mu], \tag{4}$$

where $\Lambda = \|\nabla_{\boldsymbol{x}'_l}\overline{\mathcal{M}}_l\|_\infty$.

See Appendix A for the proof.

We can find that the Laplacian mechanism is a special case of Theorem 3.1. If we do not perform the nullification, and directly add noise into the final output of the local neural network, then $\mu = 0$, $\overline{\mathcal{M}}_l(\boldsymbol{x}) = \boldsymbol{x}$, and so $\nabla_{\boldsymbol{x}'_l}\overline{\mathcal{M}}_l = \mathbf{1}$. As a result, $\varepsilon = 2\sigma$, which agrees with the Laplacian mechanism.

3.3 Noisy Training

The differentially private transformation on the mobile side provides a guaranteed privacy for sensitive data, but perturbing the local transformation incurs the performance sacrifice in the cloud-side inferences nonetheless. The cloud-side DNN trained by the conventional training method where the loss of the DNN is minimized on the purely clean training data lacks the robustness to the noisy representations revealed by mobile devices. In order to mitigate the negative effectiveness caused by the perturbation, we propose the noisy training method, another key technique in Arden, to enhance the robustness of cloud-side DNNs.

One explanation of the DNN's lack of robustness is that noisy representations are the blind points of the DNN trained by the conventional method. The robustness of cloud-side DNN could be enhanced by training on a mixture of noisy and clean representations. The new training loss is defined as:

$$\begin{aligned} J(\boldsymbol{\omega}; \boldsymbol{x}_r, \widetilde{\boldsymbol{x}}_r) &= \lambda J(\boldsymbol{\omega}; \boldsymbol{x}_r) + (1-\lambda) J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r), \\ \widetilde{\boldsymbol{x}}_r &= \boldsymbol{x}_r + Lap(B/\sigma\mathbf{I}). \end{aligned} \tag{5}$$

Instead of calculating training loss merely on the clean representations, (5) combines the losses on both the clean and the noisy representations, and uses λ to control the tradeoff between the two losses. Then, the robustness of the cloud-side DNN to noisy representations could be improved.

However, the way in which the random noise is added during the inference is unknown to anyone except for the mobile device itself. In addition, due to the nature of randomness, it is obviously impractical to train the cloud-side DNN on all possible noisy representations. A deviation from the trained noisy representations could still be the blind point of the cloud-side DNN. To overcome this problem and further enhance the robustness, we train the cloud-side DNN under the worst situation, which can be viewed as a min-max problem. We inject the worst perturbation $\boldsymbol{r}$ into the generated noisy representations to maximize the DNN's deviation from the original output, *i.e.*, the maximal loss $J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r + \boldsymbol{r})$, while the DNN tries to minimize the deviation through training:

$$\min_{\boldsymbol{r}; ||\boldsymbol{r}||_2 \leq \eta} \max J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r + \boldsymbol{r}), \tag{6}$$

where $\boldsymbol{r}$ is the perturbation added to the trained noisy representations, and η controls the scale of the noise.

During the noisy training, we determine the worst perturbation that maximizes the deviation, and train the cloud-side DNN to be robust to such perturbation through (6). Generally, it is difficult to obtain a closed form for the exact $\boldsymbol{r}$ that maximizes $J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r + \boldsymbol{r})$, especially for complex models such as the DNN. To tackle this problem, we use the first-order Taylor series to approximate $J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r + \boldsymbol{r})$:

$$J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r + \boldsymbol{r}) \approx J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r) + \nabla_{\widetilde{\boldsymbol{x}}_r} J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r)\boldsymbol{r}. \tag{7}$$

The perturbation to the noisy representation causes the deviation to grow by $\nabla_{\widetilde{\boldsymbol{x}}_r} J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r)\boldsymbol{r}$. We can maximize this increase subject to the L2-norm constraint on $\boldsymbol{r}$ by assigning the perturbation in the gradient direction of $J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r)$ with respect to $\widetilde{\boldsymbol{x}}_r$:

$$\begin{aligned} \boldsymbol{r} &= \eta \frac{\boldsymbol{g}}{||\boldsymbol{g}||_2}, \\ \boldsymbol{g} &= \nabla_{\widetilde{\boldsymbol{x}}_r} J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r). \end{aligned} \tag{8}$$

The gradient $\boldsymbol{g}$ can be efficiently computed by the backpropagation [38]. Then, the training loss is defined as:

$$J(\boldsymbol{\omega}; \boldsymbol{x}_r, \widetilde{\boldsymbol{x}}_r) = \lambda J(\boldsymbol{\omega}; \boldsymbol{x}_r) + (1-\lambda)[J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r) + J(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r + \boldsymbol{r})]. \tag{9}$$

Algorithm 2 outlines the noisy training framework for the cloud-side DNN. It is designed based on the stochastic gradient descent algorithm. Different from the conventional SGD, the noisy training jointly minimizes the loss on clean representations $\mathcal{L}_1$, the loss on noisy representations $\mathcal{L}_2$, and the loss on perturbed noisy representations $\mathcal{L}_3$. In each batch, Algorithm 2 firstly computes the

Algorithm 2: Noisy Training in Each Batch

Input: Clean representation $\boldsymbol{x}_r$; Cloud-side DNN $C(\boldsymbol{\omega})$.

Batch size N; Noise scale σ; Bound threshold B; Controllers λ and η; Learning rate α. $\boldsymbol{d} \leftarrow \mathbf{0}$;

foreach $\boldsymbol{x}_r^{(i)}$ *in* $\{\boldsymbol{x}_r^{(1)}, \ldots, \boldsymbol{x}_r^{(N)}\}$ **do**

- $\mathcal{L}_1 \leftarrow \text{Loss}(\boldsymbol{y}_{true}^{(i)}; C(\boldsymbol{\omega}; \boldsymbol{x}_r^{(i)}))$;
- $\widetilde{\boldsymbol{x}}_r^{(i)} \leftarrow \boldsymbol{x}_r + Lap(B/\sigma\mathbf{I})$;
- $\mathcal{L}_2 \leftarrow \text{Loss}(\boldsymbol{y}_{true}^{(i)}; C(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r^{(i)}))$;
- $\boldsymbol{g} \leftarrow \nabla_{\widetilde{\boldsymbol{x}}_r^{(i)}} \mathcal{L}_2$;
- $\boldsymbol{r} \leftarrow \eta \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}$;
- $\mathcal{L}_3 \leftarrow \text{Loss}(\boldsymbol{y}_{true}^{(i)}; C(\boldsymbol{\omega}; \widetilde{\boldsymbol{x}}_r^{(i)} + \boldsymbol{r}))$;
- $\mathcal{L} \leftarrow \lambda\mathcal{L}_1 + (1-\lambda)(\mathcal{L}_2 + \mathcal{L}_3)$;
- $\boldsymbol{d} \leftarrow \boldsymbol{d} + \nabla_{\boldsymbol{\omega}} \mathcal{L}$;

$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \alpha \frac{\boldsymbol{d}}{N}$;

loss on clean representations. Then, the noisy representations are generated, and are fed into the cloud-side DNN to get the loss on noisy representations. In order to augment the robustness to the random noisy representations, Algorithm 2 computes the gradient of $\mathcal{L}_2$ with respect to the noisy representations and generates the worst perturbation to the noisy representations. Finally, the back-propagation calculates the partial derivative of the joint loss with respect to each parameter in $\boldsymbol{\omega}$. The parameters $\boldsymbol{\omega}$ are updated by an average value over the batch.

4 EXPERIMENTAL EVALUATION

In this section, image classification tasks and a real mobile application are used as experimental examples to evaluate the effectiveness of ARDEN. We first examine the effect of different parameters based on two image benchmark datasets, MNIST [25] and SVHN [32], and then verify the performance improvement based on CIFAR-10 [20] and the preceding two datasets. In addition, in order to verify the effectiveness in a real scenario, we use a mobile application[3] DeepMood [6] to test ARDEN's performance and analyze the performance under different privacy budgets. DeepMood harnesses the sequential information collected from the basic keystroke patterns and the accelerometer on the phone (*e.g.*, alphanumeric character typing pattern, special character typing pattern, and accelerometer values) to predict the user's mood.

For image classification tasks, three widely used convolutional deep neural networks (Conv-Small, Conv-Middle, and Conv-Large) are implemented in ARDEN [22, 36]. We derive the local neural network from Conv-Small, which is pretrained on CIFAR-100 dataset [20]. The first 3 layers of the pretrained Conv-Small are deployed as the local neural network. For MNIST and SVHN, we use Conv-Middle as the cloud-side DNN. In the performance comparison, Conv-Large is used for CIFAR-10. For different datasets and cloud-side DNNs, the local neural network remains unchanged to show the transfer learning ability and the transparency property.

For DeepMood, the single-view DNN proposed in [6] is used as the cloud-side DNN. We add an extra dense layer as the local network which is pretrained on special character typing pattern. Alphanumeric character typing pattern (ALPH.) and accelerometer values (ACCEL.) are used to test ARDEN's performance.

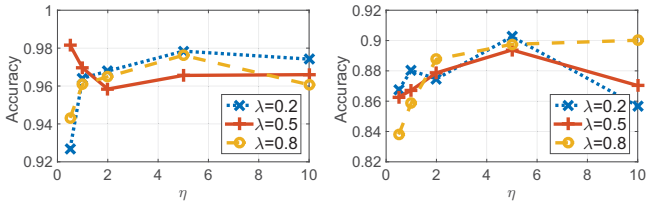
The proposed models are implemented using TensorFlow [2]. For image classification tasks, the learning rate and batch size are set as 0.0015 and 128, respectively. The numbers of epochs for MNIST, SVHN, and CIFAR-10 are 35, 45, and 70, respectively. For DeepMood, the learning rate, the batch size, and the epochs are 0.001, 256, and 100, respectively. Considering the randomness during perturbation, we run the test experiments ten times independently to obtain an averaged value. The code is provided for reproducibility.

We suppose that the deep learning service provider and the mobile device users reach a consensus over the perturbation strength in advance, which is a common practice in reality. The perturbation strength is represented by (b, μ), where b is the diversity of the Laplace distribution, and μ is the nullification rate. In this section, the perturbation strengths for MNIST, SVHN, CIFAR-10, and DeepMood (including ALPH. and ACCEL.) are set as (5, 10%), (2, 5%), (2, 5%), and (0.5, 10%), respectively. When the perturbation is injected to the last layer of local network, based on Theorem 3.1, the privacy budgets for MNIST, SVHN, CIFAR-10, ALPH., and ACCEL. are 0.7, 3.7, 3.5, 7.8, and 9.8, respectively.

[3] http://www.biaffect.com

4.1 Parameter Selection

The parameters λ and η are two important parameters in the proposed noisy training. Fig. 4 shows the change of the performance with varying λ and η. Due to space limitation, here we focus on MNIST and SVHN. It can be found that when η is too small, it has limited ability to resist the randomness of perturbation. However, when η is too large, the loss $\mathcal{L}_3$ would overwhelm the other losses, which makes the DNN focus more on resisting randomness rather than classifying samples correctly. Because a small value of λ means a large weight of $\mathcal{L}_3$ in the joint loss, small λ usually aggravates this phenomenon. Based on the results in Fig. 4, η is set as 5 for the following experiments. The effect of λ will be further discussed.



(a) MNIST (5,10%) (b) SVHN (2,5%)

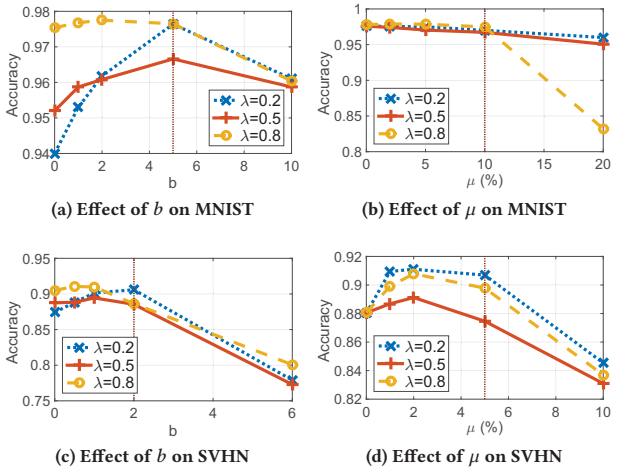
Figure 4: Effect of η and λ.

4.2 Impact of Private Perturbation

Although it is assumed that there is a prior consensus between the mobile device users and the deep learning service provider, mobile device users may change their perturbation strength. Hence, we investigate the impact of different perturbation strength on the performance when the model has been trained with a pre-assigned perturbation strength. It can be seen from Fig. 5 that the accuracy usually reaches the peak around the pre-assigned perturbation strength. An interesting observation is that when the perturbation strength is 0, *i.e.*, $b = 0, \mu = 0$, the accuracy is unexpectedly low, especially for $\lambda = 0.2, 0.5$. This is because a small value of λ makes the DNN biased to the noisy samples, which weakens the DNN's ability to classify the purely clean samples. When the perturbation strength is too large, it exceeds the resistance ability given by the noisy training, so the accuracy decreases. Owing to the noisy training, the performance changes are within 10% in most cases.

As the proposed perturbation method in Section 3 enables the layer-wise perturbation, we further study the impact of injecting perturbation into the different layers of the local neural network. The results in Table 1 show that the DNN trained by the proposed noisy training method is robust to the layer-wise perturbation. For different perturbation, the DNN can keep a relatively high accuracy.

To demonstrate the effectiveness of private perturbation intuitively, we visualize noise and reconstruction in Fig. 6. The convolutional denoising autoencoder [31], which has been successfully



(a) Effect of b on MNIST (b) Effect of μ on MNIST

(c) Effect of b on SVHN (d) Effect of μ on SVHN

Figure 5: Effect of b and μ when the model has been trained with the pre-assigned perturbation which is represented by the brown dash line.

Table 1: Accuracy in Layer-Wise Perturbation (%)

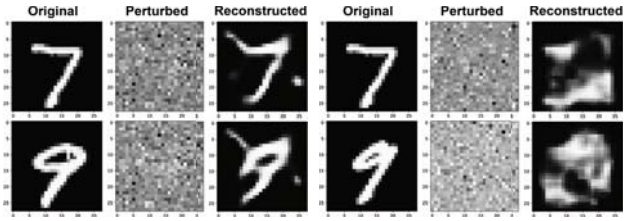
	Input	Layer 1	Layer 2	Layer 3
MNIST	97.67	97.24	97.57	98.02
SVHN	88.56	88.66	87.82	88.12

applied to image denoising and super-resolution reconstruction [8, 37], is used to reconstruct the original data from the perturbed ones. We train the model based on two perturbation strengths. When the perturbation strength is relatively weak, the perturbed pictures can to some extent be reconstructed, although the reconstructed pictures are quite different from the original ones. When the perturbation strength reaches (5,10%), the strength used in our experimental settings, the perturbed pictures can hardly be reconstructed, which indicates that the adversarial attacker cannot restore the original data based on the revealed perturbed ones even the perturbation strength is public.

4.3 Performance Comparison

To verify the performance improvement brought by ARDEN, we compare it with three variants on three image datasets and two real-scenario datasets. BASE denotes the cloud-side DNN, *i.e.*, Conv-Middle for MNIST and SVHN, Conv-Large for CIFAR-10, a sigle-view DNN for DeepMood. The training data and the testing data are fed into the network directly without local transformation, which can be regarded as revealing raw data to the cloud. ARDEN-L1 means that only the loss $\mathcal{L}_1$ is considered when training the cloud-side DNN. It is tested in two situations: test without private perturbation and test with private perturbation. ARDEN is the complete framework proposed in this paper. λ is set as 0.2 for MNIST, SVHN, and DeepMood, 0.5 for CIFAR-10.

Table 2 lists the result of every framework. BASE achieves the highest accuracy with the sacrifice of privacy. Testing ARDEN-L1 without perturbed data shows slightly lower accuracy than BASE. This result indicates that some useful information is lost during the local transformation. The accuracy of testing ARDEN-L1 with perturbed data drops severely (*e.g.*, 23.89% on ALPH. and 24.25% on ACCEL.) compared with testing ARDEN-L1 without perturbed data. This result indicates that the model trained by the traditional training method which merely relies on clean data is not applicable to the prediction with perturbation, and it is hard to maintain accuracy while protecting user's privacy. ARDEN which jointly minimizes the loss on clean data, noisy data, and perturbed noisy data, can considerably mitigate the negative impact brought by the private perturbation. Specially, the performance improvement demonstrated in the real mobile applications indicates that mobile device users can benefit from the cloud resources without risking their privacy in reality.



(a) (1, 1%) (b) (5,10%)

Figure 6: Visualization of noise and reconstruction.

Table 2: Accuracy of Different Frameworks (%)

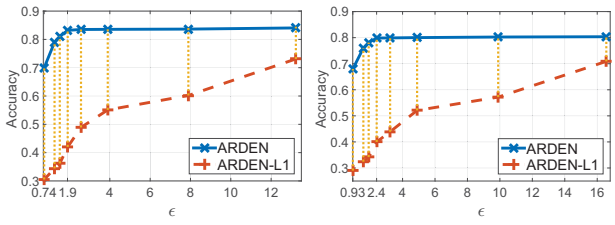
	Perturb	MNIST	SVHN	CIFAR-10	ALPH.	ACCEL.
BASE	NO	98.21	93.24	87.42	85.45	82.36
ARDEN-L1	NO	97.44	91.18	83.18	84.09	81.44
ARDEN-L1	YES	50.17	40.93	32.73	60.20	57.19
ARDEN	YES	98.16	90.02	79.52	83.55	80.05

Notice that we do not make comparison with other advanced DNNs like DenseNet [19], because we intend to examine the performance improvement brought by the noisy training. The cloud-side DNN can be replaced by any other advanced DNNs to achieve higher accuracy. Specially, thanks to the transparency property of ARDEN, the model can switch seamlessly from one to another.

4.4 Analysis of Privacy Budget

Privacy budget represents the privacy loss in the framework. To analyze the impact of privacy budgets on performance, we use the real mobile application DeepMood to test ARDEN's performance when the privacy budget changes. In this work, the privacy budget is controlled by two parameters, i.e, b and μ. Here, we fix the value of μ as 10%, and change the privacy budget ϵ by adjusting b from 0.3 to 5.

Fig. 7 shows that ARDEN can maintain the accuracy at a high value for a wide range of privacy budgets. When $\epsilon < 1$, the accuracy decreases by 14% and 12% for ALPH. and ACCEL., respectively. The accuracy almost keeps unchanged until ϵ decreases to 1.9 for ALPH. and 2.4 for ACCEL., respectively, which is a quite tight privacy



(a) ALPH. (b) ACCEL.

Figure 7: Accuracy vs. privacy budget ϵ.

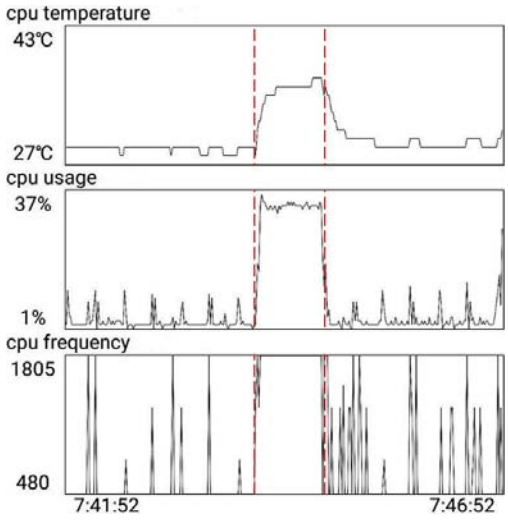


Figure 8: CPU status during local transformation.

requirement [33]. These results argue that ARDEN is applicable to different privacy requirements. It can effectively improve the performance even when the privacy budget is relatively tight. In addition, we plot the accuracy of ARDEN-L1 when it is tested with private perturbation. The gap between the two lines can be viewed as the accuracy improvement brought by the noisy training. We find that the gap is large when ϵ is small and narrows with the increase of ϵ in both datasets.

4.5 Implementation on Android

We implement the ARDEN in a demo system composed by HUAWEI HONOR 8 and DELL INSPIRON 15. The mobile device is equipped with ARM Cortex-A53@2.3GHz and ARM Cortex-A53@1.81GHz. The laptop is equipped with Core i7-7700HQ@2.80GHz and NVIDIA GTX 1050Ti. The mobile device is connected to the laptop through the IEEE 802.11 wireless network. We use TensorFlow to generate the deployable model for the Android system.

Firstly, we investigate the CPU status of the mobile device during the local transformation. 3,000 SVHN pictures are processed consecutively. Fig. 8 shows that the CPU load boosts during the local data transformation. Especially, the CPU temperature increases by 9 degrees, which is an indicator of high energy cost. This observation argues that executing a small neural network for a relatively long time can place a heavy burden on mobile devices. It is not suitable for most mobile devices to deploy DNNs directly.

Table 3: Execution Overhead Comparison

Network	Time(ms)	Memory(MB)	Energy(J)
Local	3267	0.56	1.29
Conv-Middle	7386	5.86	5.07
Conv-Large	12195	12.89	8.37
MobileNet	4106	3.34	2.82
GoogLeNet	36251	53.12	24.87

Note: The time and energy cost contain the part spent on transmitting data and waiting for the response from the laptop.

We further compare the overhead of the local neural network with those of other four DNNs. MobileNet [18] is a lightweight model for mobile vision applications. GoogLeNet [42] is a deep convolutional neural network that achieved the state of the art in ILSVRC'14. For the local neural network, the mobile device only undertakes the local transformation, and then transmits the data to the laptop, as ARDEN does. The other four DNNs entirely rely on the mobile device. The networks process 100 SVHN pictures consecutively. Table 3 lists the response time and the resource consumption of different networks. We estimate the energy consumption based on the statistics from [3, 4]. The average reductions of ARDEN compared with the other four DNNs in terms of time, memory, and energy are 60.10%, 92.07%, and 77.05%, respectively.

5 RELATED WORK

Deep learning on mobile devices. Lane et al. [24] studied typical mobile sensing tasks using DNNs. The preliminary results highlighted the critical need for further research towards making use of advances in deep learning to the field of mobile sensing. However, suffered from the contradiction between the large size of DNNs and the limited capacity of mobile devices, it is challenging to deploy deep learning applications in mobile devices efficiently. Han et al. [13] tried to compress the DNN through a three-stage method: pruning, trained quantization and Huffman coding, which showed a considerable reduction in terms of the storage requirements of DNNs. Offloading the heavy computational tasks to the cloud data center [12, 30] is another solution to enable deep learning applications on mobile devices. A distributed DNN architecture across the cloud, the edge, and the mobile devices was designed in [43] which allowed the combination of fast inference on mobile devices and complex inference in cloud data centers.

Privacy issue in deep learning. Deep learning naturally requires users' data to train neural networks and infer results, which raises the privacy issue for sensitive data. To preserve privacy when offloading operations to the cloud, Zhang et al. [46] encrypted the sensitive data by using the BGV encryption scheme. More recently, Osia et al. [35] designed a hybrid deep learning architecture for private inference across mobile devices and clouds. The Siamese network was used to protect against undesired inference attacks, which in essence provided k-anonymity protection. Li et al. [29] proposed a flexible framework for private deep learning. Before

uploading data to clouds, it was transformed by the local neural network whose structure, including the number of layers, the depth of output channels, and the subset of selected channels, was variable. Differential privacy mechanism which provides provable privacy guarantee has been used in deep learning. Shokri et al. [39] presented a privacy-preserving distributed SGD to enable multiple data owners to collaborate on training a DNN. The sparse vector technique was introduced in this work to provide differential privacy. Abadi et al. [1] designed a new differential privacy mechanism in SGD to reduce the privacy budget. However, these works applied differential privacy to the training phase rather than the inference phase. To the best of our knowledge, this is the first work using differential privacy in the inference phase to provide a provable privacy guarantee.

6 CONCLUSIONS

In order to enable highly efficient deep learning service on mobile devices, we propose ARDEN which partitions the DNN across mobile devices and clouds in this paper. All the heavy works are offloaded to the cloud, while the mobile device merely undertakes the simple data transformation and perturbation. To limit the privacy risk when uploading data to clouds, we introduce the differential privacy mechanism, and design a new differentially private perturbation that shows more flexibility and fits well with the stacking structure of neural networks. A rigorous analysis is given to determine the privacy budget of the perturbation. Apart from privacy, the inference performance is taken into consideration in this paper. A novel training method that injects deliberate noisy samples into the training data is proposed. A series of experiments based on three image datasets and a real mobile application demonstrate that ARDEN can not only preserve users' privacy but also improve the inference performance. Finally, we implement ARDEN on a demo mobile cloud system to verify its practicality and test its overhead. ARDEN effectively reduces the resource consumption by over 60%.

A PROOF OF THEOREM 3.1

Before proving Theorem 3.1, we first prove the following two theorems.

THEOREM A.1. *Given a input x and a deterministic function f, $|f(x)| \le B$, for $\forall a \in \mathbb{R}^+$, the random mechanism $\mathcal{A}(x) = f(x) + aLap(B/\sigma)$ is $(\frac{2\sigma}{a})$-differentially private.*

PROOF. For any adjacent inputs x and x',

$$\frac{\Pr[f(x) + aLap(B/\sigma) = S]}{\Pr[f(x') + aLap(B/\sigma) = S]} = \frac{e^{-\frac{|S-f(x)|\sigma}{aB}}}{e^{-\frac{|S-f(x')|\sigma}{aB}}}$$
$$= e^{\frac{\sigma}{aB}(|S-f(x')|-|S-f(x)|)}$$
$$\le e^{\frac{\sigma}{aB}|f(x)-f(x')|} \le e^{\frac{2\sigma}{a}}$$

Based on the Definition 2.1, we have $\varepsilon = \frac{2\sigma}{a}$. □

THEOREM A.2. *Given a input x, suppose $\mathcal{A}(x)$ is ε-differentially private, if we perform item-wise nullification of x with I_n whose nullification rate is μ, $x' = x \odot I_n$, then $\mathcal{A}(x')$ is ε'-differentially private,*

$$\varepsilon' = \ln[(1-\mu)e^{\varepsilon} + \mu].$$

PROOF. Suppose there are two adjacent inputs x_1 and x_2 that differ by one single item i, say $x_1 = x_2 \cup i$. For arbitrary binary matrix I_n, after the nullification $x_1' = x_1 \odot I_n$, $x_2' = x_2 \odot I_n$, there are two possible cases, *i.e.*, $i \notin x_1'$ and $i \in x_1'$.

Case 1: $i \notin x_1'$. Since x_1 and x_2 differ only by the item i, $x_1 \odot I_n = x_2 \odot I_n$. Then, we have,

$$\Pr[\mathcal{A}(x_1 \odot I_n) = S] = \Pr[\mathcal{A}(x_2 \odot I_n) = S].$$

Case 2: $i \in x_1'$. Since x_1 and x_2 differ only by the item i, x_1' and x_2' remain adjacent inputs that differ by the item i. Because $\mathcal{A}(x)$ is ε-differentially private, we have,

$$\Pr[\mathcal{A}(x_1 \odot I_n) = S] \le e^{\varepsilon} \Pr[\mathcal{A}(x_2 \odot I_n) = S].$$

Combine the two cases together, and use the fact that $\Pr[i \notin x_1'] = \mu$:

$$\begin{aligned}
&\Pr[\mathcal{A}(x_1 \odot I_n) = S] \\
&= \mu \Pr[\mathcal{A}(x_1 \odot I_n) = S] + (1-\mu)\Pr[\mathcal{A}(x_1 \odot I_n) = S] \\
&\le \mu \Pr[\mathcal{A}(x_2 \odot I_n) = S] + (1-\mu)e^{\varepsilon} \Pr[\mathcal{A}(x_2 \odot I_n) = S] \\
&= ((1-\mu)e^{\varepsilon} + \mu)\Pr[\mathcal{A}(x_2 \odot I_n) = S] \\
&= e^{\ln[(1-\mu)e^{\varepsilon}+\mu]} \Pr[\mathcal{A}(x_2 \odot I_n) = S]
\end{aligned}$$

Based on the Definition 2.1, we have $\varepsilon' = \ln[(1-\mu)e^{\varepsilon} + \mu]$. □

Then, we give the proof of Theorem 3.1.

PROOF. To begin with, we analyze the local transformation without the nullification operation. Denote the output-bound neural network $\mathcal{M}_l$ as $\mathcal{M}_l'$, the local transformation $\mathcal{A}$ defined by Algorithm 1 can be written as:

$$\begin{aligned}
\mathcal{A}(\boldsymbol{x}_s) &= \overline{\mathcal{M}}_l(\mathcal{M}_l'(\boldsymbol{x}_s) + Lap(B/\sigma\mathbf{I})) \\
&= \overline{\mathcal{M}}_l(\boldsymbol{x}_l' + Lap(B/\sigma\mathbf{I})).
\end{aligned} \tag{10}$$

Since $Lap(B/\sigma\mathbf{I})$ is much smaller than $\boldsymbol{x}_l'$ (otherwise, the utility of the representation is destroyed), we can approximate (10) by using the first-order Taylor series:

$$\begin{aligned}
&\overline{\mathcal{M}}_l(\boldsymbol{x}_l' + Lap(B/\sigma\mathbf{I})) \\
&\approx \overline{\mathcal{M}}_l(\boldsymbol{x}_l') + (\nabla_{\boldsymbol{x}_l'}\overline{\mathcal{M}}_l)^{\mathrm{T}} Lap(B/\sigma\mathbf{I}) \\
&= \mathcal{M}(\boldsymbol{x}_s) + (\nabla_{\boldsymbol{x}_l'}\overline{\mathcal{M}}_l)^{\mathrm{T}} Lap(B/\sigma\mathbf{I}).
\end{aligned} \tag{11}$$

According to the Theorem A.1, (11) is $(2\sigma/\Lambda)$-differentially private, where $\Lambda = \|\nabla_{\boldsymbol{x}_l'}\overline{\mathcal{M}}_l\|_\infty$.

Then, considering the nullification operation, the overall local transformation is $\mathcal{A}(\boldsymbol{x}_s \odot I_n)$. Based on the Theorem A.2, we have,

$$\varepsilon = \ln[(1-\mu)e^{2\sigma/\Lambda} + \mu].$$

□

ACKNOWLEDGMENTS

This work is supported by the Scientific Research Project of National University of Defense Technology through grants ZK16-03-57 and ZK16-03-09, NSFC through grants 61572511 and 61672313, as well as NSF through grants IIS-1526499, IIS-1763325, and CNS-1626432.

REFERENCES

[1] Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 308–318.

[2] Martín Abadi and et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/ Software available from tensorflow.org.

[3] Frumusanu Andrei and Smith Ryan. 2015. Cortex A53 - performance and power - ARM A53/A57/T760 investigated. (2015).

[4] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. 2009. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC)*. 280–293.

[5] Amos Beimel, Hai Brenner, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. 2014. Bounds on the sample complexity for private learning and private data release. *Machine Learning* 94, 3 (2014), 401–437.

[6] Bokai Cao, Lei Zheng, Chenwei Zhang, Philip S. Yu, Andrea Piscitello, John Zulueta, Olu Ajilore, Kelly Ryan, and Alex D. Leow. 2017. DeepMood: Modeling Mobile Phone Typing Dynamics for Mood Detection. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 747–755.

[7] 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. 2017. CirCNN: Accelerating and Compressing Deep Neural Networks Using Block-CirculantWeight Matrices. *arXiv:1708.08917* (2017). https://doi.org/abs/1708.08917

[8] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2016. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 2 (2016), 295–307.

[9] Cynthia Dwork. 2011. *Differential Privacy*. Springer US, Boston, MA, 338–340.

[10] Cynthia Dwork. 2011. A Firm Foundation for Private Data Analysis. *Commun. ACM* 54, 1 (2011), 86–95.

[11] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[12] Morteza Golkarifard, Ji Yang, Ali Movaghar, and Pan Hui. 2017. A Hitchhiker's Guide to Computation Offloading: Opinions from Practitioners. *IEEE Communications Magazine* 55, 7 (2017), 193–199.

[13] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *4th International Conference on Learning Representations (ICLR)*.

[14] Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning Both Weights and Connections for Efficient Neural Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*. 1135–1143.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

[17] G. E. Hinton and R. R. Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* 313, 5786 (2006), 504–507.

[18] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861* (2017). https://doi.org/abs/1704.04861

[19] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten. 2017. Densely Connected Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[20] A Krizhevsky and G Hinton. 2009. Learning Multiple Layers of Features from Tiny Images. (2009).

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*. 1097–1105.

[22] Samuli Laine. 2017. Temporal Ensembling for Semi-Supervised Learning. In *5th International Conference on Learning Representations (ICLR)*.

[23] Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. 2016. DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices. In *15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 1–12.

[24] Nicholas D. Lane and Petko Georgiev. 2015. Can Deep Learning Revolutionize Mobile Sensing?. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications (HotMobile)*. 117–122.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Geoffrey Hinton. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[26] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4681–4690.

[27] Paul Lee. 2017. Technology, Media and Telecommunications Predictions. *Delloitte Touche Tohmatsu Limited* (2017).

[28] Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling Source Syntax for Neural Machine Translation. In *55th annual meeting of the Association for Computational Linguistics (ACL)*. 4594–4602.

[29] Meng Li, Liangzhen Lai, Naveen Suda, Vikas Chandra, and David Z. Pan. 2017. PrivyNet: A Flexible Framework for Privacy-Preserving Deep Neural Network Training with A Fine-Grained Privacy Control. *arXiv:1709.06161* (2017). https://doi.org/abs/1709.06161

[30] Weiqing Liu, Jiannong Cao, Lei Yang, Lin Xu, Xuanjia Qiu, and Jing Li. 2017. AppBooster: Boosting the Performance of Interactive Mobile Applications with Computation Offloading and Parameter Tuning. *IEEE Transactions on Parallel and Distributed Systems* 28, 6 (2017), 1593–1606.

[31] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. 2011. *Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction*. Springer Berlin Heidelberg, Berlin, Heidelberg, 52–59.

[32] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, and Bo Wu Andrew Y. Ng. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning. In *Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS)*. 1–9.

[33] Ulfar Erlingsson Ian Goodfellow Kunal Talwar Nicolas Papernot, Martin Abadi. 2017. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In *5th International Conference on Learning Representations (ICLR)*.

[34] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1717–1724.

[35] Seyed Ali Osia, Ali Shahin Shamsabadi, Ali Taheri, Hamid R. Rabiee, Nicholas D. Lane, and Hamed Haddadi. 2017. A Hybrid Deep Learning Architecture for Privacy-Preserving Mobile Analytics. *arXiv:1703.02952* (2017). https://doi.org/abs/1703.02952

[36] Sungrae Park, Jun-Keon Park, Su-Jin Shin, and Il-Chul Moon. 2017. Adversarial Dropout for Supervised and Semi-supervised Learning. *arXiv:1707.03631* (2017). https://doi.org/abs/1707.03631

[37] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434* (2015). https://doi.org/abs/1511.06434

[38] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 9 (1986), 533 – 536.

[39] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 1310–1321.

[40] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556* (2014). https://doi.org/abs/1409.1556

[41] Lichao Sun, Yuqi Wang, Bokai Cao, S Yu Philip, Witawas Srisa-an, and Alex D Leow. 2017. Sequential Keystroke Behavioral Biometrics for Mobile User Identification via Multi-view Deep Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*. 228–240.

[42] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1–9.

[43] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. 2017. Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices. In *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. 328–339.

[44] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*. 3320–3328.

[45] Jiecao Yu, Andrew Lukefahr, David Palframan, Ganesh Dasika, Reetuparna Das, and Scott Mahlke. 2017. Scalpel: Customizing DNN Pruning to the Underlying Hardware Parallelism. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*. 548–560.

[46] Qingchen Zhang, Laurence T. Yang, and Zhikui Chen. 2016. Privacy Preserving Deep Computation Model on Cloud for Big Data Feature Learning. *IEEE Trans. Comput.* 65, 5 (2016), 1351–1362.

[47] Tong Zhang. 2004. Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*. 116–123.