Attacking and Protecting Data Privacy in Edge–Cloud Collaborative Inference Systems

Zecheng He, *Student Member, IEEE*, Tianwei Zhang, and Ruby B. Lee, *Life Fellow, IEEE*

***Abstract*—Benefiting from the advance of deep learning (DL) technology, Internet-of-Things (IoT) devices and systems are becoming more intelligent and multifunctional. They are expected to run various DL inference tasks with high efficiency and performance. This requirement is challenged by the mismatch between the limited computing capability of edge devices and large-scale deep neural networks. Edge–cloud collaborative systems are then introduced to mitigate this conflict, enabling resource-constrained IoT devices to host arbitrary DL applications. However, the introduction of third-party clouds can bring potential privacy issues to edge computing. In this article, we conduct a systematic study about the opportunities of attacking and protecting the privacy of edge–cloud collaborative systems. Our contributions are twofold: 1) we first devise a set of new attacks for an untrusted cloud to recover arbitrary inputs fed into the system, even if the attacker has no access to the edge device's data or computations, or permissions to query this system and 2) we empirically demonstrate that solutions that add noise fail to defeat our proposed attacks, and then propose two more effective defense methods. This provides insights and guidelines to develop more privacy-preserving collaborative systems and algorithms.**

***Index Terms*—Artificial intelligence, collaborative inference, edge–cloud computing, security and privacy.**

I. Introduction

RECENT years have witnessed the rapid development of deep learning (DL) and Internet-of-Things (IoT) technologies. IoT devices become appealing targets for DL applications. They use various sensors (e.g., cameras, microphones, and gyroscopes) to collect data and information from environmental contexts, run the DL applications to interpret sensory data, and make control decisions. The integration of AI and IoT leads to the era of Artificial Intelligence of Things (AIoT), which has significantly changed our daily life: small-scaled AIoT systems are introduced to build smart homes and increase the comfort and quality of life; medium-scale AIoT systems are deployed in warehouses and factories for higher efficiency and automation; and large-scale AIoT systems can contribute to the establishment of smart cities.

Deploying DL inference applications on commodity edge devices has several challenges. On one hand, an IoT device can collect streaming information at a very high rate (e.g., vehicle detection [2], remote monitoring [3], scene analysis [4], and application trace analysis [5]). This requires the device to run the DL models and analyze the data at a high speed. On the other hand, state-of-the-art DL models are becoming more complicated with larger sizes, making it infeasible for resource-constrained IoT devices to satisfy the performance requirements: the limited computation resources of the device can cause significant latency; the limited storage capacity makes it hard to store a large deep neural network (DNN) model; and the limited battery capacity causes a critical energy consumption constraint.

To overcome this challenge, one possible approach is to offload the entire DL model and inference computation to the cloud. The edge device sends the input data to the cloud and receives the output. While this can resolve the aforementioned limitations of edge devices, it incurs significant communication costs when sending a large volume of raw data. Besides, there can be privacy breaches of the inference data [6], especially if the input data are highly sensitive such as patients' records, and integrity breaches of the model [7], if the cloud is not trusted.

An optimized strategy is to adopt collaborative inference between the edge devices and the cloud [8]–[12]. The DL model can be divided into two parts. The first few layers of the network are stored in the local edge device, while the rest are offloaded to a remote cloud. Given an input, the edge device calculates the output of the first layers, sends it to the cloud, and retrieves the final results. This approach can reduce communication costs, as the intermediate output can be designed to be much smaller than the raw input. Such low data transfer bandwidth also achieves lower latency and smaller energy consumption. Collaborative inference makes it feasible and efficient to deploy large-scale intelligent workloads on today's edge platforms.

This article presents an investigation of inference data privacy in edge–cloud collaborative systems, from the perspectives of attacks and defenses. Prior works all aimed to improve the performance and efficiency of such systems, while ignoring potential security issues. To the best of our knowledge, we are the first to demonstrate the feasibility of input data privacy

Manuscript received May 18, 2020; revised July 18, 2020; accepted August 20, 2020. Date of publication September 8, 2020; date of current version June 7, 2021. This work was supported in part by NSF STARSS under Grant 1526493, and in part by a research gift from Siemens. The work of Tianwei Zhang was supported by Singapore MoE AcRF Tier1 under Grant RS02/19. The work of Ruby B. Lee was supported by the Qualcomm Faculty Award. This article was presented in part at the 35th Annual Computer Security Applications Conference (ACSAC'19), San Juan, PR, USA, Dec. 2019. *(Corresponding author: Ruby B. Lee.)*

Zecheng He and Ruby B. Lee are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08540 USA (e-mail: zechengh@princeton.edu; rblee@princeton.edu).

Tianwei Zhang is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: tianwei.zhang@ntu.edu.sg).

Digital Object Identifier 10.1109/JIOT.2020.3022358

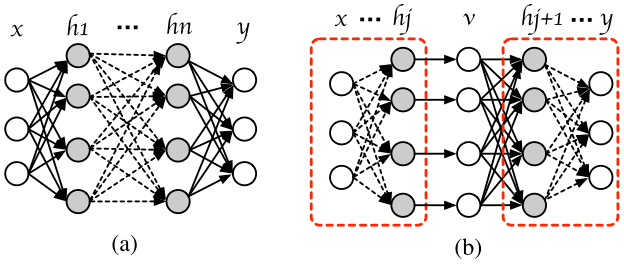


Fig. 1. (a) DNN model deployed in (b) collaborative edge–cloud system.

attacks against cloud-edge collaborative inference systems. The data privacy considered in this article is the confidentiality of the raw inputs.

Two key questions are considered in this study. The first one is: *if the cloud is malicious or compromised, can the attacker recover raw input data, otherwise available only to the edge device?* Past work claimed the edge–cloud collaborative inference can provide better privacy protection, as the cloud only receives the intermediate values instead of the raw data [10]. We show that an untrusted cloud can still easily and accurately recover the sensitive data from the intermediate values without accessing the edge-side model.

We design a set of novel attack techniques to achieve this goal under different settings. First, for a white-box attacker, we propose using regularized maximum likelihood estimation (rMLE) to recover the samples from the model parameters and intermediate values. Second, for a black-box attacker, we propose the inverse-network attack to identify the reverse mapping from the intermediate outputs to inputs without the knowledge of model information. Third, we consider the most limited adversarial capability where the cloud has no knowledge of the target model and is not allowed to query the model. Conducting privacy attacks under this setting is extremely difficult, and this threat model is rarely considered in past work. For these query-free attacks, we introduce a new method of shadow model reconstruction to achieve this attack.

The second question we address in this article is: *how can the edge devices mitigate privacy leakage from the untrusted cloud?* Past work adopted differential privacy to protect the inference data [13]. We show that this approach is impractical against our proposed attacks as it brings unacceptable performance degradation to the DL models. Instead, we propose two novel strategies that can better thwart privacy attacks while still maintaining good model performance. The first one is *the dropout defense*: by deactivating random neurons during the inference, the adversary is not able to precisely generate the original images from the intermediate values. Our second defense is *privacy-aware DNN partitioning*: we comprehensively evaluate different factors that can affect the attack results and propose some guidelines to partition the DL models for better privacy. We hope our findings can guide machine learning researchers and practitioners to design more secure collaborative inference systems.

The key contributions of this article are as follows.

1) A systematic study of attacks and defenses for inference data privacy in edge–cloud collaborative machine learning systems.
2) Three attack approaches to recover inference data under different settings.
3) Two new defense approaches to prevent inference data leakage to the untrusted cloud.

The remainder of this article is organized as follows. Section II presents the edge–cloud system model, threat model, and experimental configurations. Section III describes attacks under white-box, black-box, and query-free settings, including attack approaches, implementations, and evaluation results. Section IV discusses possible mitigation solutions. We give related work in Section V and conclude in Section VI.

II. Preliminaries

A DNN is a parameterized function $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$ that maps an input tensor $x \in \mathcal{X}$ to an output tensor $y \in \mathcal{Y}$ [Fig. 1(a)]. It consists of an input layer, an output layer, and a sequence of hidden layers between the input and output layers. Each layer is a collection of units called *neurons*, which are connected to other neurons in the previous layer and the next layer. Each connection between the neurons can transmit a signal to another neuron in the next layer. In this way, a neural network transforms the inputs through hidden layers to the outputs, by applying operations (e.g., a linear function or elementwise nonlinear activation function) in each layer.

A. System Model

In an edge–cloud collaborative inference system [Fig. 1(b)], a DNN is partitioned into two parts: $f_\theta = f_{\theta 1} \circ f_{\theta 2}$. Each part contains several layers. The edge device hosts the first part $f_{\theta 1}$. It collects inference data from the environment, generates the intermediate value $v = f_{\theta 1}(x)$, and sends it to the cloud. The cloud hosts the second part of the model $f_{\theta 2}$. When receiving the intermediate value v from the edge device, it calculates the final output $y = f_{\theta 2}(v)$ and returns it to the edge device.

Determining a way to partition the DNN model is nontrivial. Different factors must be considered to identify the optimal strategy.

1) *Latency:* An optimal partition should give the fastest inference speed. The latency is determined by the inference time on the edge device, the cloud, as well as the network transmission time. The cloud can process the inference at a much faster speed. So it is preferable to move more DNN layers to the cloud. However, this can cause larger volumes of transmitted data and longer network latency. So the performance of edge devices, cloud servers, and network transmission must be balanced.
2) *Power:* An optimal partition should be energy efficient. This is particularly important for edge devices that have limited power capabilities. The energy consumed by the edge device consists of the inference computation (determined by the number of layers) and network communication (determined by the size of transmitted data). Similar to latency optimization, the energy consumption of these two parts needs to be balanced.
3) *Memory Size:* When conducting inference, the device needs to load the entire DNN $f_{\theta 1}$ into the memory.

TABLE I
TABLE OF NOTATIONS

Symbol	Description	Symbol	Description
$\mathbb{C}$	Untrusted cloud provider	S	Training set of f_θ
$\mathbb{E}$	Edge device	x_0	Original input
f_θ	Original DNN model	$x_{i,j}$	Pixel at position (i,j) in image x
$f_{\theta 1}$	Partial model at the edge side	ED$(\cdot)$	Euclidean distance
$f_{\theta 2}$	Partial model at the cloud side	TV$(\cdot)$	Total variation of an image
$f_{\theta 1}^{-1}$	Inversed network of $f_{\theta 1}$	$y \sim f_{\theta 2}(f_{\theta 1}(x))$	Output generated from the edge-cloud collaboration
σ	Standard deviation of Gaussian noise	β	Smoothness parameter of TV
b	Scale of Laplacian noise	λ	Balancing parameter of ED and TV
r	Dropout rate	m	Number of training samples

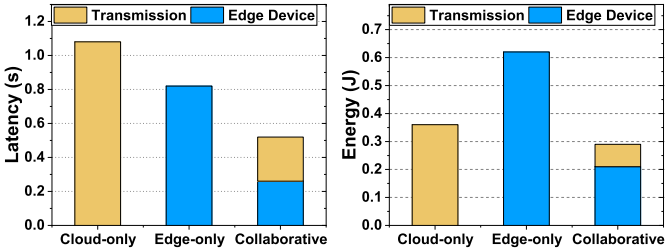


Fig. 2. Breakdown of inference latency (left) and energy consumption (right) in edge–cloud systems. Data are from [9].

Some edge devices are equipped with limited memory resources, and incapable of hosting too many network layers. This gives another constraint when selecting the optimal split point.

With these considerations, DNN partitioning is usually formulated as an optimization problem [8]–[12]. Fig. 2 shows the comparisons of latency and energy consumption between edge–cloud, cloud-only, and edge-only solutions (data are collected from [9]). We capture the results from Fig. 6 in Neurosurgeon [9]. In Neurosurgeon [9], a detailed study of latency and power consumption in a typical edge–cloud collaborative system was evaluated. An AlexNet model is deployed between a mobile device and a cloud connected by WiFi. We observe that with an optimal split point, an edge–cloud system can achieve lower latency and energy than a cloud-only or an edge-only system: by offloading some DNN layers to the cloud, the processing time and energy consumed on the device is less than the edge-only system. Meanwhile, as the size of the intermediate data is smaller than the original input, the latency and energy costs of network transmission in the edge–cloud system are also less than the cloud-only system.

In practice, most layers (including all fully connected layers) are commonly offloaded to the cloud, while the edge device only computes a small number of convolutional layers for feature extraction, due to power and resource constraints [9]. This gives a chance for an untrusted cloud provider to steal sensitive inference input, which we will discuss below.

B. Threat Model

We consider a collaborative inference system between the edge device $\mathbb{E}$ and cloud $\mathbb{C}$. The target model is split into two parts: $f_\theta = f_{\theta 2} \circ f_{\theta 1}$. $\mathbb{E}$ performs the first few layers $f_{\theta 1}$, while $\mathbb{C}$ performs the rest of the layers $f_{\theta 2}$. We consider $\mathbb{E}$ is trusted: when input is fed into $f_{\theta 1}$, $\mathbb{E}$ correctly processes it and never leaks it to other parties. However, $\mathbb{C}$ is untrusted, attempting to steal the input. We consider the confidentiality of an individual raw input when we use the term "data privacy" throughout this article. Other forms of data privacy, e.g., membership or linkability, are not in our threat model.

We assume $\mathbb{C}$ strictly follows the collaborative inference protocol: receiving $v = f_{\theta 1}(x)$ from $\mathbb{E}$ and generating $y = f_{\theta 2}(v)$. $\mathbb{C}$ cannot compromise the inference process conducted by $\mathbb{E}$, and has no knowledge of the input x, nor any intermediate values inside $\mathbb{E}$, except v. We consider adversaries with different capabilities:

1) *White Box:* $\mathbb{C}$ has the knowledge of the model at the edge side $f_{\theta 1}$, including its network structure and parameters.
2) *Black Box:* $\mathbb{C}$ does not have knowledge of $f_{\theta 1}$, but is able to query the model $f_{\theta 1}$. The adversary does not need to know the exact training data, but he can collect the same type of samples as the training data. This assumption is reasonable in practice, e.g., the adversary can collect arbitrary face samples for a face recognition model or medical records for a diagnostic system.
3) *Query Free:* $\mathbb{C}$ does not have knowledge of $f_{\theta 1}$, or the permission to query the model $f_{\theta 1}$. This type of attacks has the minimum attacker capability. Similar to the black-box attack, we assume the attacker can collect samples similar in type to the training data.

TABLE II
EXPERIMENT CONFIGURATIONS

Dataset	MNIST	CIFAR10
Target Model	LeNet-5 (2 conv + 3 fc)	6 conv + 2 fc CNN
Split points considered	• 1st conv layer (conv1) • 2nd conv layer after activation (ReLU2)	• 2nd conv layer (conv12) • 1st pooling layer (pool1) • 4th conv layer before and after activation (conv22 and ReLU22) • 2nd pooling layer (pool2)

C. Notations and Experimental Configurations

We summarize our notations in Table I. We show detailed configurations of experiments in Table II.

Our attacks and defenses are generic and applicable to various data sets. In this article, we demonstrate the attack results on the MNIST data set and the defense results on MNIST and CIFAR10. More details on the attacks can be found in [1].

The first victim model we target is LeNet5. It consists of two convolutional layer blocks (each block has a convolutional layer, an activation layer, and a pooling layer), three fully connected layers, and one softmax layer. The model can be split at either the first convolutional layer or the second convolutional layer after activation. These configurations are realistic in edge–cloud scenarios, as the heavy computational layers (including all fully connected layers) are offloaded to the cloud.

We follow the standard MNIST and CIFAR10 split for training and testing samples [14]. We set the learning rate to 10^{-3} and choose ADAM as our optimizer. The target model, all attack techniques, and defense solutions are implemented with Pytorch 1.0.1. We run our experiments on a server with one Nvidia 1080Ti GPU and two Intel Xeon E5-2667 CPUs.

To quantify the effectiveness of attacks and defenses, we adopt two metrics, peak signal-to-noise ratio (PSNR) [15] and structural similarity index (SSIM) [16]. Larger values of these two metrics indicate the recovered input is of higher quality, and more similar to the original one.

III. Attack Methodologies

A. White-Box Attack

We start from the white-box setting, where the adversarial cloud knows the parameters of the initial layers $f_{\theta 1}$ on the edge device. Formally, the problem we consider is: *how can the adversary recover an input x_0, from the corresponding intermediate value $f_{\theta 1}(x_0)$, and the model parameters $\theta 1$?* We propose rMLE to solve this problem.

rMLE: We treat the attack as an optimization problem: given $f_{\theta 1}(x_0)$, our goal is to find a generated sample x, which satisfies two requirements: 1) the intermediate output of this sample $f_{\theta 1}(x)$ is similar to $f_{\theta 1}(x_0)$ and 2) x is a natural sample, following the same distribution as other inference samples.

For requirement (1), we use the Euclidean distance (ED) to measure the similarity between $f_{\theta 1}(x)$ and $f_{\theta 1}(x_0)$ [(1a)]. Note that $f_{\theta 1}(x)$ can be interpreted as the mapping from the input space (unobservable to the adversary) to the feature space (observable to the adversary). Then, this ED represents the *posteriori* information from the adversary's intermediate-level observation. Our goal is to find the optimal sample x that minimizes this distance.

For requirement (2), we adopt the *total variation* [17] to represent the *prior* information of an input sample. The total variation of a 2-D image x is defined in (1b), where $x_{i,j}$ represents the pixel at position (i, j). β is a parameter that controls the smoothness of the image. Larger β results in more piecewise-smoothed images. We set $\beta = 1.0$ throughout our experiments. Minimization of this metric can guarantee the generated image x is piecewise smooth, i.e., avoiding drastic variations inside regions but allowing large changes along the region boundaries

$$ED(x, x_0) = \|f_{\theta 1}(x) - f_{\theta 1}(x_0)\|_2^2 \tag{1a}$$

$$TV(x) = \sum_{i,j}\left(\left|x_{i+1,j} - x_{i,j}\right|^2 + \left|x_{i,j+1} - x_{i,j}\right|^2\right)^{\beta/2} \tag{1b}$$

$$x^* = \text{argmin}_x \;\; ED(x, x_0) + \lambda TV(x). \tag{1c}$$

Fig. 3. Recovered inputs in white-box attacks.

The total objective function of the model inversion problem is a combination of feature space similarity and input smoothness, as shown in (1c). In this equation, λ is a hyperparameter to balance the effects of the two terms. If the feature space, $f_{\theta 1}(x)$ is far from the input space, i.e., a lot of network layers are computed on the trusted participant $\mathbb{E}$, a large λ is required because less posterior information about the input can be recovered from the feature space and the adversary needs to rely on the prior information. In contrast, if only a small number of layers are deployed on $\mathbb{E}$, then the adversary only needs to select a small λ. We set $\lambda = 0$ when getting the inverse from layers before the first fully connected layer, and $\lambda = 0.1$ when getting the inverse from layers after the first fully connected layer. We perform gradient descent (GD) to solve (1c) and recover the image.

Evaluation: Fig. 3 shows the white-box attack results. The first row shows the original inference samples and the remaining rows are the recovered images when the split point is at different layers. We observe that the adversary can accurately recover the images with high fidelity when the split point is either at the first (conv1) or last (ReLU2) convolutional layer. At the first split layer, PSNR is 39.69 dB and SSIM is 1.00. At the last split layer, PSNR is 15.10 dB and SSIM is 0.60.[1] This indicates that when the split point is at a deeper layer, the quality and similarity of recovered images become worse.

B. Black-Box Attacks

Next, we consider the black-box setting, where the adversary does not have knowledge of the structure or parameters of $f_{\theta 1}$. We assume that the adversary can query the black-box model: he can send an arbitrary input x to $\mathbb{E}$ and observe the corresponding output $f_{\theta 1}(x)$.

Data privacy attacks under the black-box setting are more challenging because, without the knowledge of model parameters, the adversary cannot directly perform a GD on $f_{\theta 1}$ to solve the optimization problem in (1c). One solution is to first recover the model structure and parameters by querying the model, and then recover the inference samples. The possibility of model reconstruction has been demonstrated in [18]–[20].

We propose a more efficient approach, the inverse network, to directly identify the inversed mapping from output to input, without the model information. Our solution is easier to implement and can recover inputs with higher fidelity.

Inverse Network: Conceptually, the inverse network is the approximated inverse function of $f_{\theta 1}$, trained with $v = f_{\theta 1}(x)$ as input, and x as output. The attack consists of three phases:

[1] In our experiments, we observe that PSNR>10 dB or SSIM>0.3 are considered as good quality because the inversed images are visually recognizable by the adversary.

Fig. 4. Recovered inputs in black-box attacks.

1) generating a training set for the inverse network; 2) training the inverse network; and 3) recovering the input sample by querying the inverse network.

First, the adversary generates a bag of samples $X = (x_1, x_2, \ldots, x_m)$ of the same type as the training data to query the target system, and observes the corresponding intermediate outputs $V = (f_{\theta 1}(x_1), f_{\theta 1}(x_2), \ldots, f_{\theta 1}(x_m))$. Next, he can directly train an inverse network $f_{\theta 1}^{-1}$ using V as the training input and X as the training output. We initialize the inverse network with the Xavier initialization [21], to avoid the neuron activations in the saturated or dead regions in the beginning. We leverage l_2 norm in the pixel space as the loss function (2), and stochastic GD (SGD) to train the inverse network

$$f_{\theta 1}^{-1} = \text{argmin}_g \frac{1}{m} \sum_{i=1}^{m} \|g((f_{\theta 1}(x_i)) - x_i)\|^2 \tag{2}$$

where g is the inverse network to be optimized. Note that the architecture of the inverse network need not be related to the target model $f_{\theta 1}$. In our experiment, we use an entirely different network architecture.

Once the inverse network $f_{\theta 1}^{-1}$ is obtained, the adversary can recover any inference sample from the intermediate layer output: $x = f_{\theta 1}^{-1}(v)$. This approach is more efficient than rMLE: 1) for each target sample, the adversary only needs to pass through the inverse network once, while in rMLE, an iterative process is required to solve the optimization problem and 2) calculating the inversed input is parameter-free, while rMLE requires tuning the parameters λ and β in (1).

Evaluation: Fig. 4 shows the recovered images of MNIST. We can observe that the adversary can recover the input under the black-box setting with very high quality (PSNR is 40.72 and 20.81 dB for the two split points) and similarity (SSIM is 0.99 and 0.80 for the two points).

C. Query-Free Attacks

The inverse-network approach requires the adversary to be able to query the target model, and generate the data set for training f_θ^{-1}. In this section, we consider the query-free setting, where the adversary cannot query the model at the edge side and does not know the model information. The basic idea is that the adversary first reconstructs a shadow model, which imitates the target model's behavior, and then uses rMLE over this shadow model to recover the input samples.

Shadow Model Reconstruction: The problem at the first step is: how can the adversary reconstruct a shadow model of the former model layers $f_{\theta 1}$ with only the knowledge of the latter layers $f_{\theta 2}$ and the same type of training data as S? He cannot query the model with specified samples to get the intermediate values.

The key insight of our approach is that if the shadow model is reconstructed as $f'_{\theta 1}$, it should be able to classify the input with high accuracy when combined with the later layers $f_{\theta 2}$

$$y_i \sim f_{\theta 2}(f_{\theta 1}(x_i)) \sim f_{\theta 2}\big(f'_{\theta 1}(x_i)\big), \text{ for } (x_i, y_i) \in S. \tag{3}$$

Then, the task of model reconstruction can be translated into minimizing the classification error of the composition of the two models: $f_{\theta 2}(f'_{\theta 1}(x_i))$ versus y_i. Equation 4 shows the loss function for training the model, where m is the number of samples in S, *CrossEntropy* is the cross-entropy loss. Equivalently, this means the training process of $f'_{\theta 1}$ is supervised at the output layer of $f_{\theta 2}$. Once the model $f'_{\theta 1}$ is reconstructed, the adversary can perform data recovery attacks using the rMLE technique in Section III-A

$$f'_{\theta 1} = \text{argmin}_g \frac{1}{m} \sum_{i=1}^{m} \text{CrossEntropy}(f_{\theta 2}(g(x_i)), y_i) \tag{4}$$

$$\text{CrossEntropy}\big(\hat{y}, y\big) = -\sum_{c=1}^{C} y^c \log\big(\hat{y}^c\big) \tag{5}$$

where C is the number of classes of the task.

There are two phases in our approach: 1) offline shadow model reconstruction and 2) online model inversion. The shadow model reconstruction only needs to be performed once. Then, all the input samples can be recovered using the same shadow model, by only one inference for each input. In the shadow model reconstruction phase, the adversary can adopt the same type of samples as the training data. He may not know the original network structure $f_{\theta 1}$, but he can use an alternative one for the shadow model. We assume that both the target model and the shadow model are convolutional neural networks, but with different numbers of layers and filters, as well as filter sizes.

After the training set and network structure are determined, the adversary can adopt SGD to optimize the loss function of the composition of the two models. We choose the cross-entropy loss because it performs well on image classification tasks. Other loss functions can be leveraged, if the adversary aims to find inverses of the DNN for different tasks. Once the shadow model is obtained, the adversary can use rMLE to recover the inputs.

Evaluation: We illustrate the recovered images under the query-free setting in Fig. 5. The adversary can still recover the input images from conv1 and ReLU2 layers. The PSNRs for these two split points are 17.86 and 8.03 dB, while the SSIMs are 0.64 and 0.38, respectively. The quality of the images is relatively lower than the ones in the white-box or black-box setting, indicating the query-free attacks are harder to achieve. This is straightforward, as the adversary now has smaller capabilities. Also, more layers on the edge device can also increase the difficulty of image recovery.

IV. Defense Methodologies

Given the severity of inference privacy attacks in edge–cloud collaborative systems, we aim to explore defense methods in this section. We first empirically evaluate one common

Fig. 5. Recovered input in query-free attacks.

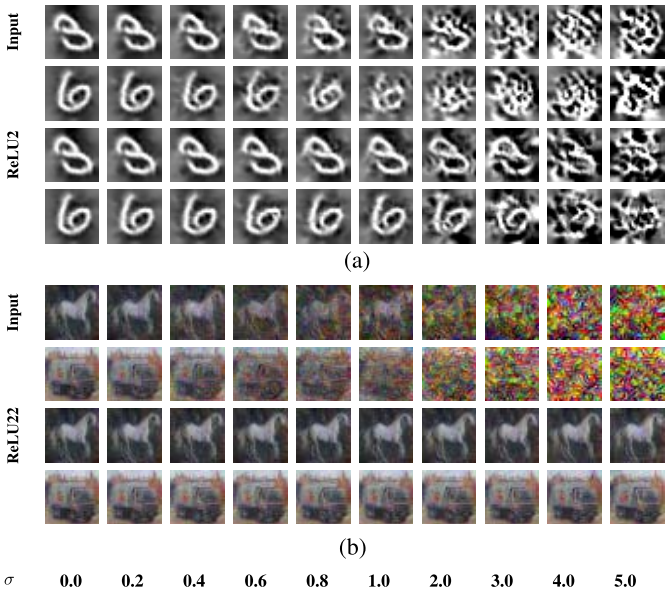


Fig. 6. Examples of adding the Gaussian noise to defend against a white-box attack on (a) MNIST (ReLU2) and (b) CIFAR10 (ReLU22). σ is the standard deviation of the Gaussian noise.

method proposed in past work (though it did not specifically target the edge–cloud privacy attacks), *viz.*, noise obfuscation. We show its ineffectiveness in defeating our inference data privacy attacks. Then, we introduce two new strategies that can better prevent privacy leakage with a small impact on the system's performance and functionalities.

A. Obfuscation With Random Noise

Differential privacy has been proposed to protect model inference [22], [23] through adding random noise to the input. In the edge–cloud scenario, we can either add noise to the original input: $v = f_{\theta 1}(x + \epsilon)$, or add noise directly to the intermediate value before sending it to the untrusted cloud $\mathbb{C}$: $v = f_{\theta 1}(x) + \epsilon$. There is a tradeoff between usability and privacy: as a higher level of noise is added, the model accuracy will drop. Whether this tradeoff can be balanced is critical for the effectiveness of this approach. Below, we measure the attack effects as well as the model accuracy using noise obfuscation.

We consider the Gaussian and Laplacian noise in our experiments. Figs. 6(a) and (b) (Gaussian) and 7(a) and (b) (Laplacian) visually show the recovered images on the MNIST and CIFAR10 data sets, when we add different levels of noise to the input (first two rows in each figure) or the intermediate layer output (last two rows). We observe that adding enough

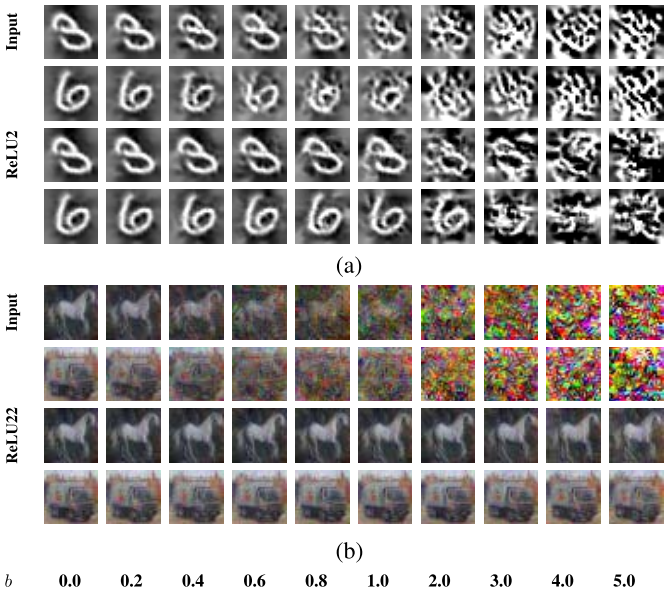


Fig. 7. Examples of adding the Laplacian noise to defend against a white-box attack on (a) MNIST (ReLU2) and (b) CIFAR10 (ReLU22). b is the standard deviation of the Laplacian noise.

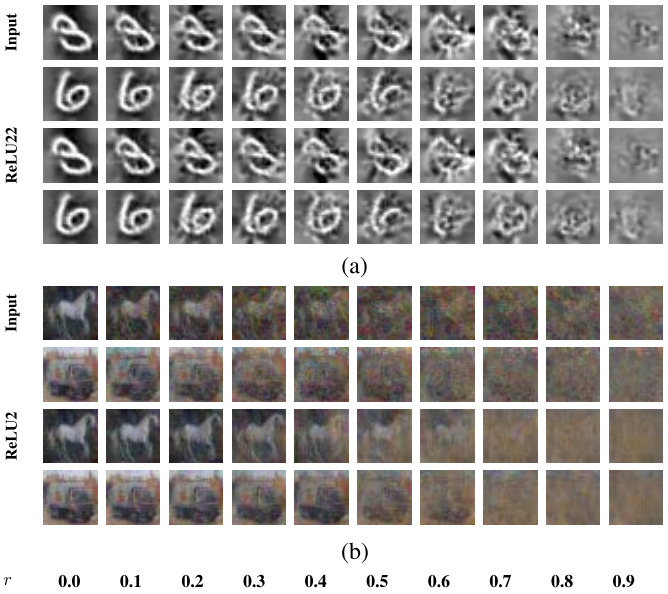


Fig. 8. Examples of dropout to defend against a white-box attack on (a) MNIST (ReLU2) and (b) CIFAR10 (ReLU22). r is the dropout ratio.

noise can indeed provide better privacy and decrease the quality of recovered images. Besides, noise at the original input is more effective than noise at the intermediate layer.

We provide a quantitative analysis of model accuracy (usability, y-axis) and inversed image quality (privacy, x-axis) in Figs. 9 and 10 on MNIST and CIFAR10 data sets, respectively. The Gaussian and Laplacian noise are represented as blue and orange lines, respectively. Adding noise to the input and the intermediate layer are represented as solid and dotted lines, respectively. The top-left region of the graph is the best. When fixing the recovered image quality (SSIM or PSNR), the model accuracy drops more if the noise is added to the input (blue and orange solid lines) than to the intermediate

layer (blue and orange dotted lines). This is consistent with the visual observations in Figs. 6(c) and 7(c). Different characteristics of noise distributions, e.g., the Gaussian or Laplacian, do not show a significant difference in model accuracy.

On the MNIST data set (Fig. 9), to maintain a good model accuracy (i.e., >95%), the noise level must be restricted to $\sigma < 0.8$ and $b < 0.5$. At this level, the attacker is still able to recover images with high quality (SSIM > 0.4 and PSNR > 8.5 dB). Similar results are shown on the CIFAR10 data set in Fig. 10. While recent work [6], [13] proposed special algorithms for designing noise to protect inference data privacy, they still may not work for our new attacks or need extra special training of the noise generator. Hence, we propose new defense methods below that are not based on adding noise and are more practical in that they protect the inference data privacy with much smaller performance degradation.

B. Dropout Defense

Since noise obfuscation may not be secure, we propose another randomization-based solution, dropout, to defeat the proposed attacks. Dropout deactivates random neurons in one layer by setting their output to 0. Formally, it calculates

$$f_i^{\text{dropout}}(x) = f(x) \otimes M \tag{6}$$

where M is a mask, where each element of M is randomly assigned a value of 0 with probability r and a value of 1 with probability $1 - r$. $\otimes$ denotes the elementwise multiplication. Intuitively, dropout leverages the redundancy feature of neural networks [24], such that removing partial information in the inference does not degrade the model performance but obfuscates the input data.

Similar to noise obfuscation, dropout can also be applied to the input or the intermediate layer output. We show examples of the images recovered from layer ReLU2 (MNIST) in Fig. 8(c). The top two rows represent the effect of dropout on input, while the bottom two rows represent that on intermediate output. We observe that increasing the dropout rate r decreases the quality of inversed images. No useful information can be obtained by the attacker when r reaches 0.6. We show reversed images from ReLU22 (CIFAR10) in Fig. 8(b). Similarly, no useful information can be obtained when r reaches 0.6.

We further measure the usability–privacy tradeoff of dropout and compare it with the noise obfuscation approach (Fig. 9 on MNIST and Fig. 10 on CIFAR10). Higher accuracy represents better usability, while smaller SSIM and PSNR represent better privacy. Lines that are closer to the top-left region have a better tradeoff. We observe that dropout (green lines) significantly outperforms all the noise obfuscation solutions (blue and orange lines). This is because dropout leverages DNN model redundancy to hide partial information and maintain model accuracy while adding random noise introduces obfuscation on all neurons which degrade model accuracy. Besides, dropout on the intermediate layer (green dotted line) is slightly better than dropout on the input (green solid line): it can fully protect the inference data privacy (SSIM < 0.25) with accuracy $> 95\%$. On the CIFAR10 data set, dropout on the intermediate layer significantly overperforms the other approaches, fully protecting inference data privacy (SSIM < 0.25) with <0.8% drop in accuracy.

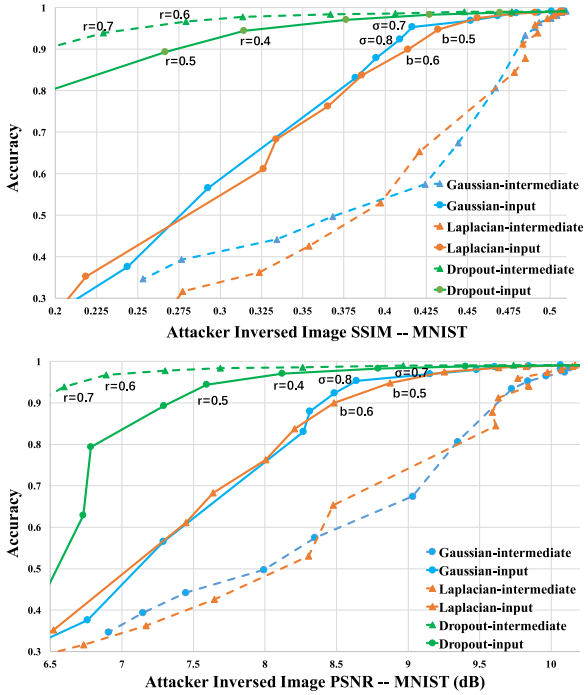


Fig. 9. Model accuracy versus the SSIM (top) and PSNR (bottom) of inversed images on the MNIST data set.

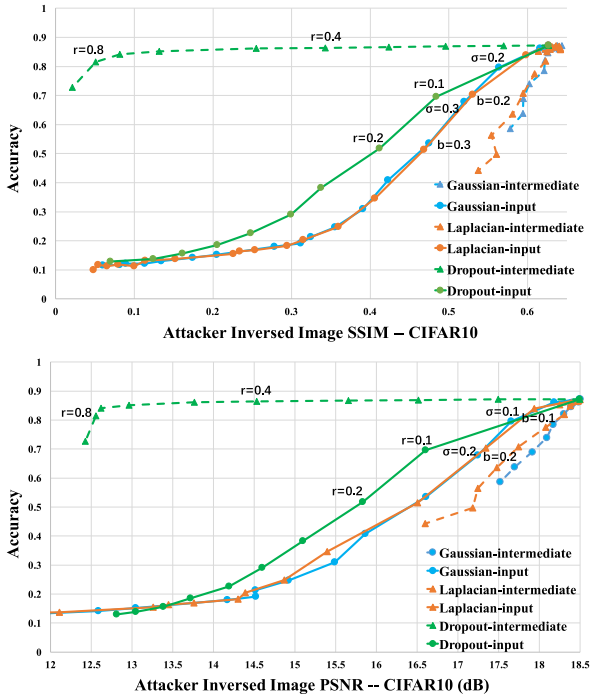


Fig. 10. Model accuracy versus the SSIM (top) and PSNR (bottom) of inversed images on the CIFAR10 data set.

To fully evaluate the effectiveness of this dropout mechanism, we consider splitting the model at different layers on the MNIST data set. We conduct dropout on the intermediate layer (i.e., the split layer) since it is better than that on the input.

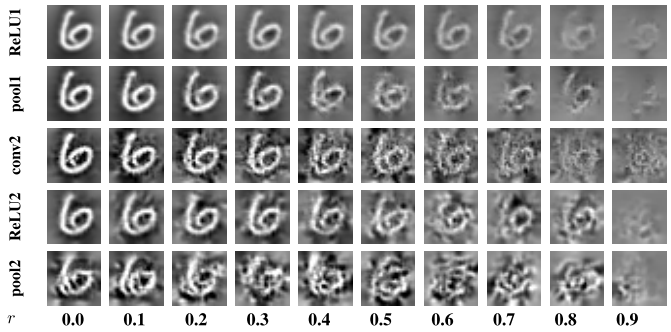


Fig. 11. Dropout as a defense against attacks at different layers on the MNIST data set. Rows from top to bottom: ReLU1, pool1, conv2, ReLU2, and pool2.

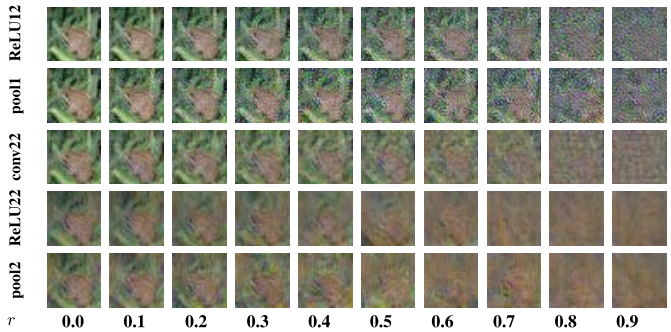


Fig. 12. Dropout as a defense against attacks at different layers on the CIFAR10 data set. Rows from top to bottom: ReLU12, pool1, conv22, ReLU22, and pool2.

Fig. 11 shows the recovered images from shallow to deep layers: ReLU1, pool1, conv2, ReLU2, and pool2. We observe that as the split layer becomes deeper, a smaller dropout rate is sufficient to prevent privacy leakage. For example, to fully obfuscate the input, r can be set as 0.9 when the model is split at the ReLU1 layer (the first row), and 0.2 when the model is split at the pool2 layer (the last row). This can be better illustrated in the usability–privacy curves in Fig. 13: dropout on deeper layers is more effective (closer to top-left regions) than that on shallow layers. For both SSIM and PSNR, we have from worse to better: pool1(blue), then conv2 (green), then ReLU2 (orange), and then pool2 (gray). There is only one exception: the ReLU1 layer, which does better than expected. It is the best for SSIM and better than conv2 for PSNR. One possible reason is that the recovered image maintains a visually recognizable structure but degrades illumination in the ReLU1 layer, which contributes more significantly to PSNR and SSIM than human recognition. Similar results on the CIFAR10 data set are shown in Fig. 12.

C. Privacy-Aware DNN Partitioning

Section III shows that different split points yield different attack effects. This observation leads to another possible defense strategy: privacy-aware model partitioning. We raise an important question: *how to split the neural network in the collaborative system, to make the inference data more secure?* We use the query-free attack as an example to explore this question. We select the split point at each layer and perform

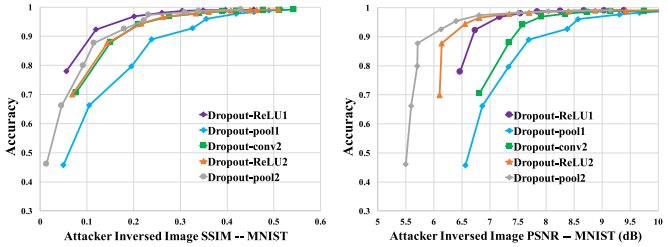


Fig. 13. Model accuracy versus the SSIM (left) and PSNR (right) of inversed image.

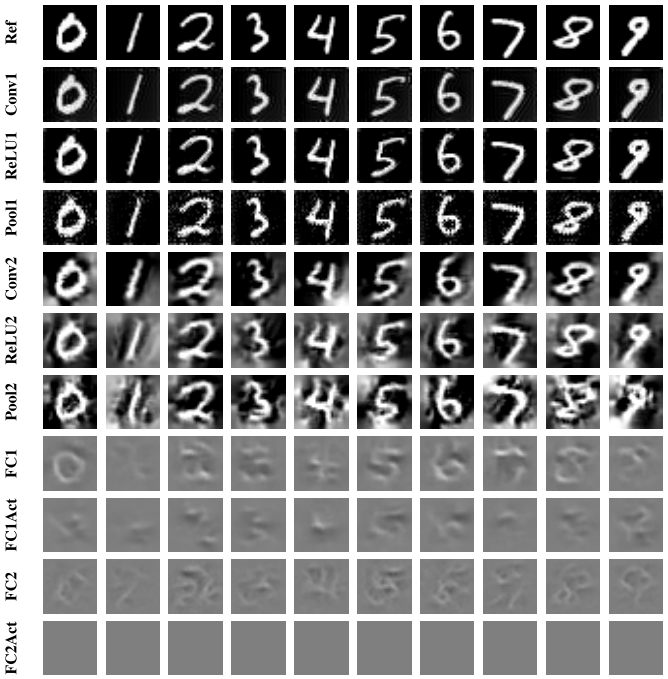


Fig. 14. Recovered images in query-free attacks.

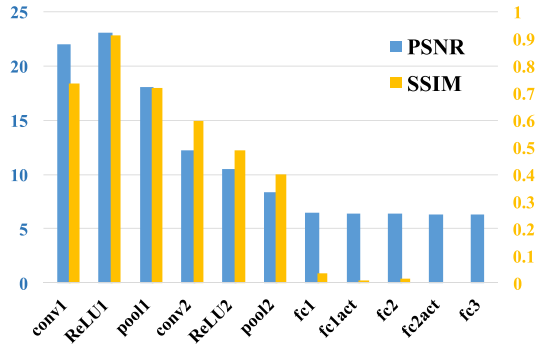


Fig. 15. PSNR and SSIM in query-free attacks.

inference privacy attacks. Figs. 14 and 15 show the recovered images and PSNR/SSIM metrics, respectively.

Generally, we observe that the quality of recovered images decreases when the split layer becomes deeper. This is straightforward as the relationship between input and output becomes more complicated and harder to revert when there are more layers. Besides, we also observe that the image quality drops significantly, both qualitatively (Fig. 14) and quantitatively

(Fig. 15), on the fully connected layer (fc1), indicating that model inversion with fully connected layers is much harder than that with convolutional layers. The reason is that a convolutional layer only operates on local elements (the locality depends on the kernel size), while a fully connected layer entirely mixes up the patterns from the previous layer. Besides, the number of output neurons in a fully connected layer is typically much smaller than input neurons. So it is relatively harder to find the reversed relationship from the output of the fully connected layer to the input.

Privacy-Aware Partitioning Strategy: When selecting the split point in a collaborative inference system, privacy should also be considered, in addition to latency and power constraints. We recommend placing at least one fully connected layer on the edge device to hide the information of sensitive input samples.

V. Related Work

A. Machine Learning Privacy Attacks

Training Data Privacy Attacks: There are different types of privacy attacks against the training data. The first type is *property inference attacks*, which try to infer some properties of the training data from the model parameters. Attacks were demonstrated in traditional machine learning classifiers [25] and fully connected neural networks [26].

A special case of property inference attacks is *membership inference attacks*, which infer whether one individual sample is included in the training set. This attack was first presented in [27]. The following work explored the feasibility of attacks with different adversary's capabilities [28], model features [29], [30], in generative adversarial networks [31], [32], and collaborative training systems [33].

The second type of attacks against the training data's privacy are *model inversion attacks* [34]: given a machine learning model, and part of the training samples' features, the adversary can recover the rest of the features of the samples. Advanced model inversion attacks were designed to recover images from DNNs in single-party systems [35], and collaborative learning systems [36].

The third type is *model encoding attacks* [37]: the adversary with direct access to the training data can encode the sensitive data into the model for a receiver entity to retrieve.

Model Privacy Attacks: The adversary attempts to steal the model parameters [18], hyperparameters [20], or structures [19], [38], via prediction APIs, memory side channels, etc.

Inference Data Privacy Attacks: Closer to our study is the work [39], which trains an inverse network on the output probability distribution to get the inversed inference data. However, they only consider the model inversion attack from the softmax layer in the black-box scenario. We show that the attacker can successfully inverse the model from different layers, even in a stricter query-free scenario. We also provide defense strategies that are not discussed in their paper. Wei *et al.* [40] adopted a power side channel to recover inference data. However, this attack required the adversary to compromise the victim device for side-channel information collection, and it could only recover simple images (single pixel). Our work can recover any arbitrary complex data without access to, or knowledge of, the victim's device and computation.

B. Machine Learning Privacy Solutions

Enhancing the Algorithms: Distributed training was introduced to protect the training data [41], [42], as different participants can use their own data for model training. The SGX security enclaves in Intel processors were used to protect the training tasks against privileged adversaries [43], [44]. Cao and Yang [45] proposed a methodology to remove the effects of sensitive training samples on the models. Abadi *et al.* [46] applied differential privacy to add noise in the SGD process to eliminate the parameters' dependency on the training data.

Enhancing the Training Data Set: Bost *et al.* [47] proposed to encrypt the data before feeding them into the training algorithm. They designed machine learning operators that can operate on the encrypted data. Zhang *et al.* [48] showed that adding noise to the training data set is effective in protecting training data privacy. Generating artificial data [49]–[51] has been proposed for training DNN models while removing sensitive information from the original data.

Obfruscating the Inference Input: Differential privacy has been proposed to protect model inference [22], [23] through adding random noise to the input. We show that just adding noise cannot defend against our attacks, and hence we also propose two defenses that may be more practical for our attacks in this article. Recent work [6] proposed to add specially designed noise and provided a theoretical analysis on the input data privacy leakage. However, it did not consider the model inversion attacks that we propose and requires extra training of the noise generator.

Homomorphic Encryption: This allows the inference application on the untrusted participant to directly perform DNN computations on encrypted input [52], [53], so the sensitive information will not be leaked. A drawback of homomorphic encryption is that it suffers from huge inefficiency and is not applicable to all DNN operations.

VI. Conclusion

In this article, we explored the inference data privacy threats in edge–cloud collaborative systems. We discovered that an untrusted cloud can easily recover the inference samples from intermediate values. We proposed a set of new attack techniques to compromise the inference data privacy under different attack settings. We demonstrated that the adversary can successfully and reliably recover the inputs with very few prerequisites.

We also proposed several methods to protect the inference data privacy for edge computing. Previous works, all focused on the performance, efficiency, and functionalities of AIoT while ignoring privacy. We hope that this study can raise awareness about the importance of inference data privacy protection in edge–cloud systems and encourage the balancing of privacy protection with usability when designing or implementing such systems.

REFERENCES

[1] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Proc. 35th Annu. Comput. Security Appl. Conf.*, 2019, pp. 148–162.

[2] Y. Tang, C. Zhang, R. Gu, P. Li, and B. Yang, "Vehicle detection and recognition for intelligent traffic surveillance system," *Multimedia Tools Appl.*, vol. 76, no. 4, pp. 5817–5832, 2017.

[3] G. Chen, T. X. Han, Z. He, R. Kays, and T. Forrester, "Deep convolutional neural network based species recognition for wild animal monitoring," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Paris, France, 2014, pp. 858–862.

[4] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, 2015, pp. 833–841.

[5] L. Xiao, Y. Li, X. Huang, and X. Du, "Cloud-based malware detection game for mobile devices with offloading," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2742–2750, Oct. 2017.

[6] F. Mireshghallah, M. Taram, P. Ramrakhyani, A. Jalali, D. Tullsen, and H. Esmaeilzadeh, "Shredder: Learning noise distributions to protect inference privacy," in *Proc. 25th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2020, pp. 3–18.

[7] Z. He, T. Zhang, and R. Lee, "Sensitive-sample fingerprinting of deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Long Beach, CA, USA, 2019, pp. 4729–4737.

[8] J. Hauswald, T. Manville, Q. Zheng, R. Dreslinski, C. Chakrabarti, and T. Mudge, "A hybrid approach to offloading mobile image classification," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Florence, Italy, 2014, pp. 8375–8379.

[9] Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGPLAN Notices*, vol. 52, no. 4, pp. 615–629, 2017.

[10] S. Teerapittayanon, B. McDanel, and H. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Atlanta, GA, USA, 2017, pp. 328–339.

[11] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained Internet-of-Things platforms," in *Proc. IEEE Int. Conf. Adv. Video Signal Based Surveillance*, Auckland, New Zealand, 2018, pp. 1–6.

[12] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, "JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services," 2018. [Online]. Available: arXiv:1801.08618.

[13] F. Mireshghallah, M. Taram, A. Jalali, A. T. Elthakeb, D. Tullsen, and H. Esmaeilzadeh, "A principled approach to learning stochastic representations for privacy in deep neural inference," 2020. [Online]. Available: arXiv:2003.12154.

[14] (2018). *Torchvision.Datasets*. [Online]. Available: https://pytorch.org/docs/0.4.0/torchvision/datasets.html

[15] (2018). *Peak-Signal-to-Noise-Ratio*. [Online]. Available: https://en.wikipedia.org/wiki/Peak-signal-to-noise-ratio

[16] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[17] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D, Nonlinear Phenom.*, vol. 60, nos. 1–4, pp. 259–268, 1992.

[18] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. 25th USENIX Security Symp.*, 2016, pp. 608–618.

[19] S. J. Oh, M. Augustin, M. Fritz, and B. Schiele, "Towards reverse-engineering black-box neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 2–3.

[20] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *Proc. IEEE Symp. Security Privacy*, San Francisco, CA, USA, 2018, pp. 36–52.

[21] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Stat.*, 2010, pp. 249–256.

[22] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.*, 2006, pp. 265–284.

[23] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.

[24] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, "An exploration of parameter redundancy in deep networks with circulant projections," in *Proc. IEEE Int. Conf. Comput. Vis.*, Santiago, Chile, 2015, pp. 2857–2865.

[25] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *Int. J. Security Netw.*, vol. 10, pp. 137–150, Sep. 2015.

[26] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proc. ACM Conf. Comput. Commun. Security*, 2018, pp. 619–633.

[27] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Security Privacy*, San Jose, CA, USA, 2017, pp. 3–18.

[28] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2018.

[29] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proc. IEEE Comput. Security Found. Symp.*, 2018, pp. 268–282.

[30] Y. Long *et al.*, "Understanding membership inferences on well-generalized learning models," 2018. [Online]. Available: arXiv:1802.04889.

[31] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "LOGAN: Membership inference attacks against generative models," 2017. [Online]. Available: arXiv:1705.07663.

[32] K. S. Liu, C. Xiao, B. Li, and J. Gao, "Performing co-membership attacks against deep generative models," 2018. [Online]. Available: arXiv:1805.09898.

[33] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Security Privacy*, San Francisco, CA, USA, 2019, pp. 691–706.

[34] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *Proc. USENIX Security Symp.*, 2014, pp. 17–32.

[35] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. ACM Conf. Comput. Commun. Security*, 2015, pp. 1322–1333.

[36] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM Conf. Comput. Commun. Security*, 2017, pp. 603–618.

[37] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proc. ACM Conf. Comput. Commun. Security*, 2017, pp. 587–601.

[38] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *Proc. ACM/ESDA/IEEE Design Autom. Conf.*, 2018, pp. 1–6.

[39] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, "Neural network inversion in adversarial setting via background knowledge alignment," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2019, pp. 225–240.

[40] L. Wei, B. Luo, Y. Li, Y. Liu, and Q. Xu, "I know what you see: Power side-channel attack on convolutional neural network accelerators," in *Proc. Annu. Comput. Security Appl. Conf.*, 2018, pp. 393–406.

[41] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. ACM Conf. Comput. Commun. Security*, 2015, pp. 1310–1321.

[42] J. Hamm, A. C. Champion, G. Chen, M. Belkin, and D. Xuan, "Crowd-ML: A privacy-preserving learning framework for a crowd of smart devices," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Columbus, OH, USA, 2015, pp. 11–20.

[43] O. Ohrimenko *et al.*, "Oblivious multi-party machine learning on trusted processors," in *Proc. 25th USENIX Security Symp.*, 2016, pp. 619–636.

[44] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, "Chiron: Privacy-preserving machine learning as a service," 2018. [Online]. Available: arXiv:1803.05961.

[45] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *Proc. IEEE Symp. Security Privacy*, San Jose, CA, USA, 2015, pp. 463–480.

[46] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM Conf. Comput. Commun. Security*, 2016, pp. 308–318.

[47] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2015.

[48] T. Zhang, Z. He, and R. B. Lee, "Privacy-preserving machine learning through data obfuscation," 2018. [Online]. Available: arXiv:1807.01860.

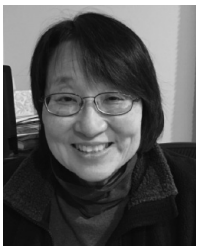
[49] A. Triastcyn and B. Faltings, "Generating artificial data for private deep learning," 2018. [Online]. Available: arXiv:1803.03148.
[50] X. Zhang, S. Ji, and T. Wang, "Differentially private releasing via deep generative model (technical report)," 2018. [Online]. Available: arXiv:1801.01594.
[51] H. Yin *et al.*, "Dreaming to distill: Data-free knowledge transfer via deepinversion," 2019. [Online]. Available: arXiv:1912.08795.
[52] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 201–210.
[53] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *Proc. 27th USENIX Security Symp.*, 2018, pp. 1651–1669.

Zecheng He (Student Member, IEEE) received the bachelor's degree from the University of Science and Technology of China, Hefei, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA.

His research focuses on security and privacy in intelligent computer systems.

Tianwei Zhang received the bachelor's degree from Peking University, Beijing, China, in 2011, and the Ph.D. degree from Princeton University, Princeton, NJ, USA, in 2017.

He is an Assistant Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research focuses on computer system security. He is particularly interested in security threats and defenses in machine learning systems, autonomous systems, computer architecture, and distributed systems.



Ruby B. Lee (Life Fellow, IEEE) received the A.B. degree (with distinction) from Cornell University, Ithaca, NY, USA, and the Ph.D. degree in electrical engineering with a minor in computer science from Stanford University, Stanford, CA, USA.

She is the Forrest G. Hamrick Professor of Engineering with Princeton University, Princeton, NJ, USA, where she is the Director of the Princeton Architecture Lab for Multimedia and Security (PALMS). Prior to Princeton University, she served as the Chief Architect with the Computer Systems Division, Hewlett Packard, Silicon Valley, CA, USA. Her current research is at the intersection of cyber security, computer architecture, and deep learning. Her research includes improving security with deep learning, low-cost deep learning processors and designing new architectures against attacks on microarchitecture, such as Spectre and Meltdown. Her past research includes architectures for secure processors and secure caches, and improving the security of smartphones and cloud computing servers.