Protecting Inference Privacy with Accuracy Improvement in Mobile-Cloud Deep Learning

Shulan Wang, *Graduate Student Member, IEEE,* Qin Liu, *Member, IEEE,* Yang Xu, *Member, IEEE,* Hongbo Jiang, *Senior Member, IEEE,* Jie Wu, *Fellow, IEEE,* Tian Wang, *Member, IEEE,* Tao Peng, *Member, IEEE,* and Guojun Wang, *Member, IEEE*

Abstract—With the wide spread of data-driven deep learning applications, a growing number of users outsource compute-intensive inference processes to the cloud. To protect inference privacy, Liu et al. (INFOCOM 2022) proposed two steganography-based solutions, named GHOST and GHOST$^+$, relying on the mobile-cloud collaborative framework, where the mobile device hides sensitive images into public cover images before feature extraction, while launching adversarial attacks on the cloud-side deep neural network (DNN) to obtain desired results. Although both solutions demonstrate significant advantages in private deep learning, they suffer from limited practicality; since the inference accuracy decreases sharply as the hiding ratio increases. To address this, we propose two improved solutions, IGHO and IGHO$^+$, which ensure high inference accuracy even when abundant sensitive images need to be hidden. Specifically, IGHO as the improved version of GHOST proposes two feature fusion methods, feature synthesis and pixel synthesis, to preprocess cover images, making the poisoned DNN learn hidden sensitive features better, while IGHO$^+$ as the improved version of GHOST$^+$ designs a novel feature mining generative adversarial network (FMGAN) to craft adversarial perturbations highly robust against variable sensitive types. Experimental results show that the proposed solutions highly improve the practicality of GHOST and GHOST$^+$.

Index Terms—Mobile cloud, deep learning, inference privacy, steganography, adversarial attacks

1 Introduction

With the advent of big data era, data-driven deep learning has played a considerable role in many vision tasks, such as face recognition and autonomous driving [1]. Due to the increased functionalities and complexities, sophisticated deep neural networks (DNNs) entail enormous training data and compute-heavy training and inference processes. In consequence, resource-limited users usually shift entire DNNs and inference computation to a cloud; by using machine learning as a service (MLaaS) [2] or pre-trained online DNNs [3]. Despite promising, this common practice leads to potential privacy risks. For instance, 41% of organizations surveyed had experienced an AI privacy breach or security incident, according to Gartner's latest survey of AI adoption [4]. Although a variety of attempts have been conducted in private deep learning, the majority of them focus on the training stage [6]–[14]. For example, one hotspot research direction [13], [14] is how to employ adversarial training [15] to preserve membership and attribute privacy of training data. In practice, users often send sensitive images directly to the cloud for real-time predictions, resulting in the disclosure of personal privacy. Hence, it is vital to protect sensitive data against the cloud in the inference stage.

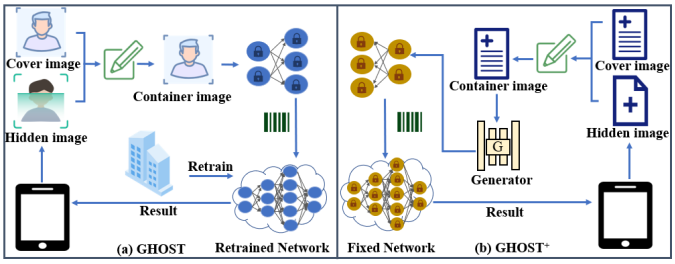


Fig. 1: The high-level idea of GHOST and GHOST$^+$. A DNN is partitioned into a frozen mobile-side network (responsible for feature extraction) and a cloud-side network that may be retrained (responsible for performing inference tasks based on the features).

To preserve inference privacy, researchers have attempted to adapt cryptographic techniques [16]–[23] and differential privacy (DP) [24]–[29] to the context of DNNs. For all this, how to strike a balance among privacy, efficiency, and accuracy is still a challenging problem for outsourced inference services. To tackle this issue, Liu et al. [30] as the first attempt integrated steganography [31]–[33] and adversarial attacks [34] and proposed two mobile-cloud collaborative solutions, named GHOST and GHOST$^+$.

As illustrated in Fig. 1, both solutions utilize steganography technologies hiding sensitive images into public cover images for enhanced privacy, while inducing a DNN to output the labels of hidden sensitive images by launching adversarial attacks. To obtain adversary-selected results, GHOST retrains the DNN into

- *Shulan Wang, Qin Liu, Yang Xu, and Hongbo Jiang are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan Province, P. R. China, 410082. E-mail: gracelq628@hnu.edu.cn; Wangsl@hnu.edu.cn; xuyangcs@hnu.edu.cn; hongbojiang2004@gmail.com*
- *Jie Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA. E-mail: jiewu@temple.edu*
- *Tian Wang is with the Institute of Artificial Intelligence and Future Networks, Beijing Normal University & UIC, Zhuhai, Guangdong Province, P. R. China, 519000. E-mail: cs_tianwang@163.com*
- *Peng and Guojun Wang are with the School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, Guangdong Province, P. R. China, 510006. E-mail: pengtao@gzhu.edu.cn; csgjwang@gzhu.edu.cn*

This work was supported in part by the National Key Research and Development Program of China under Grants 2022YFE0201400 and 2020YFB1005804, in part by the NSFC Grants 62272150, 62372121, U20A20181, 62272162, 62172159, 62002113, and 62272154, in part by the Natural Science Foundation of Guangdong Province of China under Grant 2023A1515012358, in part by the Hunan Provincial Natural Science Foundation of China under Grants 2021JJ30294, 2023JJ30267, 2020JJ3015, and in part by the Science and Technology Innovation Program of Hunan Province under Grant 2023RC3125. (Corresponding author: Qin Liu.)

a poisoned network to learn hidden sensitive features, while GHOST$^+$ produces adversarial perturbations by a generative adversarial network (GAN) [35]. In regard to application scenarios, GHOST entails retraining the DNN and is suitable for MLaaS, while GHOST$^+$ keeps the DNN intact and applies to online inference services. For instance, if a company relies on Amazon SageMaker [36] to train a face recognition clocking-in system, GHOST can be employed to hide employees' facial features; if a patient expects to obtain an online diagnosis from Closed-Loop [37], GHOST$^+$ would help protect sensitive medical data.

Although both GHOST and GHOST$^+$ outperform the state-of-the-art (SOTA) private deep learning solutions, their practicability is bounded by the number of sensitive types. For instance, as the hiding ratio increases from 1:9 to 4:6, the classification accuracy of the CIFAR-10 dataset decreases sharply from 100% to 60% and from 98% to 43% in GHOST and GHOST$^+$, respectively. In reality, sensitive types vary in different applications, and it is unrealistic to expect that only a small number of sensitive images need to be protected. For practicality, it is essential to answer the following open problem: *How to design steganography-based private deep learning solutions robust against variable sensitive types?*

To address the problem above, we propose IGHO and IGHO$^+$ as the improved versions of GHOST and GHOST$^+$, respectively. In fact, for both GHOST and GHOST$^+$, the greater number of sensitive types means the larger amounts of hidden features need to be learned, meanwhile, the more obvious the gap between cover and sensitive features, the greater the difficulty of learning hidden features. As for GHOST, the leading reason resulting in an accuracy drop is the prominent feature boundary. Therefore, IGHO preprocesses a cover dataset instead of directly using public images so as to weaken the negative influence of cover features on inference accuracy. Specifically, the cover dataset can be preprocessed either by *feature synthesis* that produces cover images of fused sensitive and public features or by *pixel synthesis* that haphazardly splices multiple public images together. As for GHOST$^+$, the practicality is also confined by the limited learning ability of GAN. Therefore, IGHO$^+$ proposes a novel Feature Mining GAN (referred to as FMGAN) that generates adversarial perturbations in place of the GAN. Specifically, FMGAN extends SOTA BigBiGAN [38] by adding *a frozen discriminator* and a *hidden feature encoder* guided by contrastive loss [39] to better extract common semantic features of hidden sensitive images. By combining preprocessing and FMGAN, IGHO$^+$ improves the robustness against variable sensitive types. Our main contributions are summarized as follows:

- We continue to explore the feasibility of steganography in preserving inference privacy in mobile-cloud deep learning. Our focus is on improving the practicality of GHOST and GHOST$^+$ without sacrificing privacy and efficiency.
- We first propose IGHO that improves the practicality of GHOST by preprocessing a cover dataset via feature synthesis or pixel synthesis. We then propose IGHO$^+$ built based on a novel FMGAN model that generates adversarial perturbations robust against variable sensitive types. In summary, our solutions not only retain all the advantages of GHOST and GHOST$^+$ but also have better practicality in mobile-cloud deep learning applications.
- We formally analyze the privacy level provided by our solutions using mutual information (MI). We empirically validate that it is hard for attackers to detect the existence of sensitive images by feature inversion attacks [40] or measuring the invisibility and leaked information.
- We amend the BigBiGAN structure to extract the hidden features most relevant to the target types in a stable manner. Extensive evaluations are performed on four datasets, MNIST [41], CIFAR-10 [42], GTSRB [43], and SVHN [44]. We observe that even in the worst case, IGHO and IGHO$^+$ improve the query accuracy by up to 1.64× and 1.60× against the GHOST and GHOST$^+$, respectively.

TABLE 1: Private Deep Learning Solutions for Inference Privacy

Technique	Solution	Drawback
Cryptography	BAYHENN [16], ENSEI [17], DELPHI [18], HEMET [19], OPNN [20], CryptoEyes [21], DeepReDuce [22]	Time-consuming Heavy overheads
DP	FORSEEN [25], Shredder [26] MGM [27], ODPSGD [28], PrivateMail [29]	Difficulty in balancing accuracy and privacy
Steganography	GHOST, GHOST$^+$ [30]	Unstable accuracy

Paper Organization. We introduce the related work in Section 2 and provide the preliminaries in Section 3; before overviewing this work in Section 4. We then illustrate the IGHO and IGHO$^+$ solutions in Sections 5 and 6, respectively. After providing the privacy analyses in Section 7, we evaluate our solutions in Section 8. Finally, we conclude this paper in Section 9.

2 Related Work

Existing private deep learning solutions span different aspects of deep learning from training stage [5]–[13] to inference stage [16]–[29]. This work focuses on preserving inference privacy. Current mainstream approaches are based on cryptographic techniques and differential privacy. Beyond these, steganography turns out to be a feasible alternative to accuracy-aware privacy protection [30]. Table 1 summarizes recent work on preserving inference privacy.

Cryptography. Homomorphic encryption (HE) and secure multi-party computation (MPC) [45] that allow the server to perform computation directly on encrypted data without decryption are often used to protect inference privacy and DNN models. Xie et al. [16] designed BAYHENN by combing HE and Bayesian neural networks to protect users' raw data and the weights of the DNN deployed in the cloud. Bian et al. [17] combined HE and secure sharing and proposed a secure inference framework ENSEI, which reduced the convolution computation overhead by using homomorphic frequency domain convolution. In order to reduce online prediction latency while preserving inference privacy, Mishra et al. [18] proposed DELPHI by using secure sharing to selectively substitute layer-wise ReLU activations. Lou et al. [19] proposed HEMET, a HE-friendly architecture to achieve shorter inference latency and higher inference accuracy in mobile neural network. Li et al. [20] proposed OPNN that employed two non-colluding servers with secure sharing and HE to improve the prediction efficiency. Instead of using HE and MPC, He et al. [21] proposed a two-stream convolutional network architecture, named CryptoEyes, which utilized symmetric block cipher and permutation to boost the classification accuracy over encrypted images. Although the cryptography-based solutions ensure a high level of privacy and accuracy, they suffer from the limitation of heavy computational and communication costs. For instance, given a single encrypted CIFAR-10 image, DELPHI [18] has to

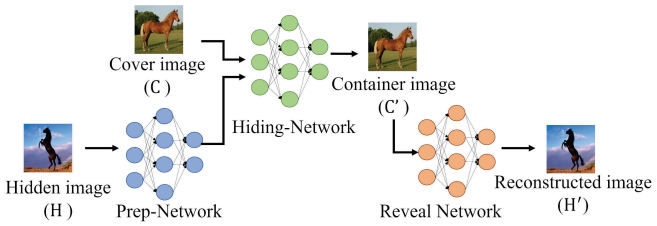


Fig. 2: The structure of the NNS framework.

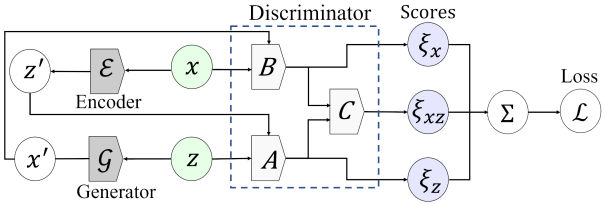


Fig. 3: The structure of the BigBiGAN framework.

transmit 2GB data, while ENSEI [17] consumes more than 30 seconds to perform inference on a ResNet-32 classification model.

Differential Privacy. DP provides provable privacy guarantee for sensitive data and is increasingly regarded as a standard notion tailored for privacy-preserving data analyses. To ensure privacy, DP normally adds deliberate noise into sensitive data. Lyu et al. [25] put forward FORESEEN, a hybrid privacy-preserving deep learning framework that utilized a collaborative noisy training algorithm and a representation perturber to protect inference privacy. For improved inference accuracy, Mireshghallah et al. [26] proposed Shredder, which learned the distribution of additive noises without altering the DNN; Yang et al. [27] proposed MGM, a new differential privacy mechanism imposing unimodal distributions on noises; Xiang et al. [28] proposed an optimized mechanism for differentially-private stochastic gradient descent (OPSGD) by choosing directions of noise. As a new attempt, Vepakomma et al. [29] proposed PrivateMail, a differentially private supervised manifold learning method for the image retrieval problem. However, a common challenge faced by the DP-based solutions is how to strike a good balance between accuracy and privacy, especially in complex DNNs. In general, the more noise added for higher privacy, the lower the accuracy. For instance, when the amount of noises is minor, sensitive data can be successfully restored via the convolutional denoising autoencoder [46]; when a large amount of noise is injected, the inference accuracy drops to below 50%.

Besides the above research, Liu et al. [30] proposed GHOST and GHOST$^+$, which for the first time utilized steganography and adversarial attacks to protect inference privacy in the dark. The main drawback of their solutions is that the inference accuracy degrades dramatically; when the number of sensitive types exceeds a given range. Inspired by their work, we also turn the vulnerability of DNNs to adversarial attacks into the weapon for high accuracy; and further explore the feasibility of steganography in preserving inference privacy. Our design goal is to improve the practicality of GHOST and GHOST$^+$ without sacrificing privacy and efficiency.

3 Preliminaries

In this section, we will briefly introduce the background knowledge and overview the GHOST and GHOST$^+$ solutions.

3.1 Steganography

Steganography is an information hiding technique that uses content redundancy in digital media to achieve covert communication. In brief, a sensitive image is hidden into a cover image to form a container image, from which it is hard to detect hidden information. This work mainly investigates the following methods:

Least-Significant-Bit Substitution (LSB). LSB [31] hides the secret information into the k-rightmost (the least significant) bits of the cover image. When the value of k exceeds 4, the visual quality of container images degrades drastically. Hence, we just embed the most significant bits (MSBs) of a secret image instead of a full-size image into a cover image.

Neural Network-based Steganography (NNS). As shown in Fig. 2, the NNS system [32] consists of a prep-network, a hiding-network, and a reveal-network, where the prep-network transforms hidden images into features that can be used by the hiding-network to form container images, while the reveal-network is responsible for extracting hidden images from the container images. The full system is trained simultaneously by minimizing the following loss:

$$\mathscr{L}_{NNS} = \mathbb{E}(\|C - C'\|_2) + \beta \cdot \mathbb{E}(\|H - H'\|_2), \tag{1}$$

where $\|C - C'\|_2$ (resp. $\|H - H'\|_2$) is the Euclid distance between the pixels of cover image C and container image C' (resp. hidden image H and reconstructed image H'), and β is the weight balancing between the invisibility and restorability of hidden images. In this work, it is unnecessary for the reveal-network to reconstruct hidden images, and the following loss function is adopted instead:

$$\mathscr{L}^*_{NNS} = \mathbb{E}(\|C - C'\|_2) + \beta \cdot \mathbb{E}(\|t - t'\|_2), \tag{2}$$

where $\|t - t'\|_2$ denotes the Euclid distance between hidden features t and reconstructed features t'.

3.2 BigBiGAN

BigBiGAN combines the best of BiGAN [47] and BigGAN [48], which learns the bidirectional mapping between the image and feature spaces in a stable manner. As shown in Fig. 3, BigBiGAN is composed of an encoder ε, a generator $\mathscr{G}$, and a joint discriminator module $\mathscr{D}$ consisting of three discriminators A, B, and C. Taking an image x as input, the encoder ε tries to learn the semantic features z', i.e., $z' \leftarrow \varepsilon(x)$. Given an input feature z, the generator $\mathscr{G}$ aims to output the corresponding image x', i.e., $x' \leftarrow \mathscr{G}(z)$. The goal of the discriminator module $\mathscr{D}$ is to distinguish between joint data-latent distributions. Let ξ_x and ξ_z be the scores used to measure the quality of images and features, respectively, and let ξ_{xz} be the score quantifying the bilateral mapping between images and features. The discriminator loss is expressed as:

$$\begin{aligned}\mathscr{L}_d = {} & \mathbb{E}_{fake}[h(-\xi_x) + h(-\xi_z) + h(-\xi_{xz})] \\ & + \mathbb{E}_{true}[h(\xi_x) + h(\xi_z) + h(\xi_{xz})],\end{aligned} \tag{3}$$

where $h(t) = \max\{0, 1-t\}$ is a hinge to regularize the discriminators. To fool the discriminators with synthesized outputs, the encoder ε and the generator $\mathscr{G}$ are trained by the following loss:

$$\mathscr{L}_{eg} = \mathbb{E}_{fake}[-\xi_x - \xi_z - \xi_{xz}] + \mathbb{E}_{true}[\xi_x + \xi_z + \xi_{xz}]. \tag{4}$$

In Eq. 3 and Eq. 4, $\mathbb{E}_{true}[\cdot]$ (resp. $\mathbb{E}_{fake}[\cdot]$) represents the average score regarding original inputs (resp. the outputs of the encoder/generator). However, the encoder of BigBiGAN fits only to extract superficial features of container images; and cannot be applied directly to generate adversarial perturbations robust against various hidden images. Therefore, IGHO$^+$ proposes FM-GAN that subtly modifies BigBiGAN to fulfill our design goal.

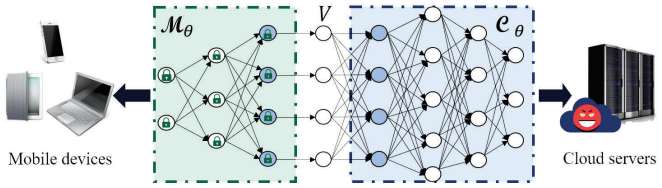


Fig. 4: The mobile-cloud collaborative framework.

3.3 Overview of GHOST and GHOST$^+$

As shown in Fig. 1, both GHOST and GHOST$^+$ are built based on the mobile-cloud collaborative framework, and apply LSB/NNS to hide sensitive images into cover images. The main difference is that GHOST retrains the cloud-side network to learn the hidden features of sensitive images, but GHOST$^+$ locally trains a GAN to craft adversarial perturbations without altering the DNN.

GHOST. At a high level, GHOST consists of the *covert retraining* and *covert inference* phases. In the retraining phase, the cloud-side network is retrained on a container dataset $\widetilde{\mathbf{D}}_c$ and a public dataset $\mathbf{D}_p$, by utilizing the mini-batch stochastic gradient descent (SGD) [49]. Specifically, given a cover dataset $\mathbf{D}_c$ and a sensitive dataset $\mathbf{D}_s$, the container dataset $\widetilde{\mathbf{D}}_c$ is formed as follows: For a pair of samples $C_i \in \mathbf{D}_c$ and $S_j \in \mathbf{D}_s$, the mobile device generates a container image $\widetilde{C}_i$ by hiding the sensitive image S_j into the cover image C_i, and then puts the container image and the sensitive image's label $(\widetilde{C}_i, l_{S_j})$ into $\widetilde{\mathbf{D}}_c$. By this means, each sample in $\widetilde{\mathbf{D}}_c$ is visually similar to a cover image, but tagged with the label of hidden sensitive image. In the inference phase, the mobile device first hides the sensitive image into a randomly picked public image; and then uploads the features of container images to the cloud for inference results. The whole process is analogous to launching backdoor attacks on the DNN by poisoning the training dataset, where the hidden sensitive images can be regarded as a kind of invisible trigger. The backdoored DNN outputs the labels of hidden sensitive images when taking container images' features as input but behaves normally under clean inputs.

GHOST$^+$. In a nutshell, GHOST$^+$ includes the *GAN training* and *adversarial inference* phases. In the training phase, the mobile device locally trains the GAN to learn perturbations by using a container dataset $\widetilde{\mathbf{D}}_c$ constructed in the same way as GHOST. The GAN is composed of a generator and a frozen discriminator, i.e. the pre-trained DNN, where the generator aims to craft the least perturbations to minimize the distance between the prediction of perturbed container image and the label of sensitive image. In the inference phase, the mobile device hides sensitive images by using steganography; and then employs the well-trained generator to perturb container images before feature extraction. Intuitively, GHOST$^+$ takes advantage of the vulnerability of the DNN against adversarial samples, which are crafted by adding imperceptible perturbations on container images, thus misleading the DNN to output the labels of hidden sensitive images.

4 Overview

This section will introduce our models and outline the differences between our solutions and the GHOST and GHOST$^+$ solutions.

4.1 System and Threat Models

As is shown in Fig. 4, our system model is based on a mobile-cloud collaborative framework, where a DNN $f_\theta = F_1 \circ F_2 \circ \ldots \circ F_L$ consisting of L interconnected layers is partitioned into the mobile-side network $\mathscr{M}_\theta$ and the cloud-side network $\mathscr{C}_\theta$. Given an inference sample, $\mathscr{M}_\theta$ extracts features and sends the intermediate value V to $\mathscr{C}_\theta$, which performs compute-heavy inference tasks and returns the predictions. Due to efficiency considerations [50], $\mathscr{M}_\theta$ consists of a small number of conventional layers with frozen structure and weights, while $\mathscr{C}_\theta$ consists of most of the remaining layers (including the fully-connected layers). In this way, the mobile device not only spends less time to process a shallow-layer network compared with the whole DNN, but also consumes less bandwidth to transmit the intermediate value compared with the original input. Moreover, as demonstrated in [40], the partitioning manipulation is conducive to privacy protection: The more layers deployed on the mobile side, the higher the privacy level achieved. To strike a balance between efficiency and privacy protection, the number of layers of $\mathscr{M}_\theta$ is set to **3** in this paper.

Our threat model assumes that the mobile device is fully trusted, while the cloud as a potential attacker is equipped with the knowledge of the entire DNN. This means that the mobile-side network $\mathscr{M}_\theta$ will correctly process the input and never expose any private information. In contrast, the cloud-side network $\mathscr{C}_\theta$ is interested in inferring useful information from the data received. Our privacy goal is to prevent the cloud from gaining sensitive data from intermediate values instead of protecting inference results. For example, the cloud may perform feature inversion attacks in a white-box setting [40] to recover the original input. For quick reference, the most relevant notations are shown in Table 2.

TABLE 2: Notation Table

Symbols	Meaning
f_θ	A pre-trained DNN
$\mathscr{M}_\theta$, $\mathscr{C}_\theta$	The mobile-side and cloud-side network
S_*, $\mathbf{D}_s$	Sensitive image and sensitive dataset
P_*, $\mathbf{D}_p$	Public image and public dataset
$\widetilde{C}_*$, $\widetilde{\mathbf{D}}_c$	Container image and container dataset
C_*, $\mathbf{D}_{pc}$	Cover image and cover dataset
l_*, z_*	Label and feature representation of a sample

Subscript $*$ indicates an arbitrary image.

4.2 GHOST vs. IGHO

Although GHOST retrains the cloud-side network $\mathscr{C}_\theta$ for improved robustness, the inference accuracy is negatively impacted by the increase of hiding ratio. From the experimental results of GHOST, we have two observations: (1) *As the hiding ratio increases, the DNN needs to learn more hidden features to produce target labels;* (2) *The more obvious the features of cover images, the greater the difficulty of learning hidden features from container images.* When the cloud-side network is sufficiently complex, the major reason resulting in an accuracy drop is the significant gap between cover and sensitive features. To mitigate the negative influence of cover images, IGHO adds a preprocessing step compared with GHOST. As shown in Fig. 5, IGHO mainly includes the *preprocessing*, *cover retraining* and *covert inference* steps.

Preprocessing. GHOST directly uses public images to form the cover dataset, resulting in a distinct gap between cover and hidden features. Instead, IGHO preprocesses the cover dataset by either feature synthesis or pixel synthesis, thereby blurring the feature boundary. The details of this step can be found in Section 5.

Covert Retraining. Given the preprocessed cover dataset $\mathbf{D}_{pc}$, and a sensitive dataset $\mathbf{D}_s$, IGHO generates a container dataset $\widetilde{\mathbf{D}}_c$ in the same way as GHOST. The cloud-side network $\mathscr{C}_\theta$ is retrained with the following loss function:

$$\mathscr{L}(\mathscr{C}; z_P, z_{\widetilde{C}}) = \mathbb{E}[J(\mathscr{C}_\theta(z_P), l_P)] + \lambda \cdot \mathbb{E}[J(\mathscr{C}_\theta(z_{\widetilde{C}}), l_{\widetilde{C}})], \quad (5)$$

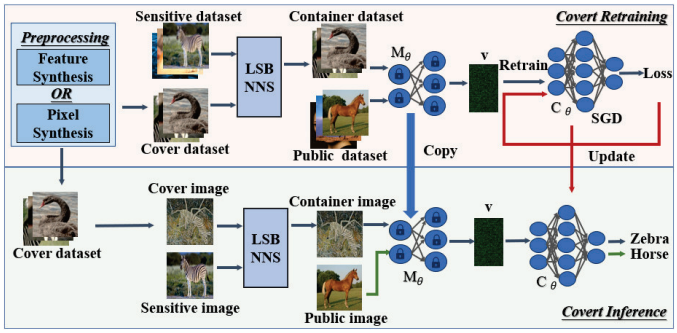


Fig. 5: The overview of IGHO.

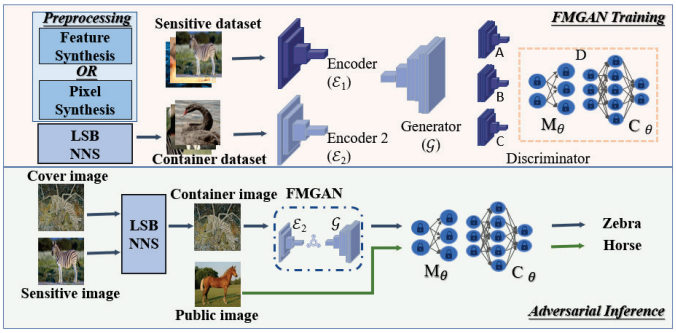


Fig. 6: The overview of IGHO$^+$. The process of producing a container dataset is similar to that in IGHO and is omitted.

where $z_P = \mathscr{M}_\theta(P)$ and $z_{\widetilde{C}} = \mathscr{M}_\theta(\widetilde{C})$ are representations of samples $P \in \mathbf{D}_p$ and $\widetilde{C} \in \mathbf{D}_c$, respectively, l_P and $l_{\widetilde{C}}$ are their labels ($l_{\widetilde{C}}$ is the label of hidden image H inside image $\widetilde{C}$), and λ controls the importance between the losses of raw training data and poisoned training data. In the above equation, function J is used to measure the distance between the prediction of $\mathscr{C}_\theta(\cdot)$ and the label l_*.

Covert Inference. Once the retraining process is finished, the poisoned cloud-side network $\mathscr{C}_\theta$ will perform inference tasks in a privacy-preserving way. As for insensitive images, the mobile device directly uploads the features to the cloud without applying steganography. As for sensitive images, the mobile device first picks random samples from the processed cover dataset, and then generates container images before uploading the extracted features to the cloud. Once receiving a query request, the cloud performs inference computation and returns the final prediction results.

Comparison between IGHO and GHOST. Both solutions utilize steganography to hide sensitive data and intrusively modify the cloud-side network $\mathscr{C}_\theta$; so that $\mathscr{C}_\theta$ outputs correct labels for clean instances, but implements the misclassification whenever a sensitive image is hidden in the input. The main difference is that GHOST directly uses public images as cover images, but IGHO employs a preprocessing step to produce cover images of fuzzy feature boundaries. Our experimental results demonstrate that the preprocessing step is conducive to improving the practicality of GHOST at the cost of a slight efficiency degradation. For example, when the hiding ratio is 4:6, IGHO incurs only 56min and 16min in total to raise inference accuracy of GHOST from 62% to 85.51% and from 62% to 78.20%; by using feature synthesis and pixel synthesis to preprocess the CIFAR-10 dataset, respectively.

4.3 GHOST$^+$ vs. IGHO$^+$

GHOST$^+$ keeps the entire DNN intact and locally trains a GAN to add deliberate perturbations into container images. From the experimental results of GHOST$^+$, we observe that GHOST$^+$ also suffers from the accuracy degradation problem; and draw a conclusion that the negative influence of cover features as well as the limited learning ability of GAN are two main reasons causing the restricted practicality. To tackle this problem, IGHO$^+$ not only adds a preprocessing step, but also replaces the GAN with a novel FMGAN structure compared with GHOST$^+$. As shown in Fig. 6, IGHO$^+$ consists of three main steps: *preprocessing*, *FMGAN training*, and *adversarial inference*.

Preprocessing. As in IGHO, IGHO$^+$ also preprocesses the cover dataset by either feature synthesis or pixel synthesis.

FMGAN Training. The design of FMGAN is inspired by the BigBiGAN [38] structure, where an encoder, a generator, and three discriminators cooperate to bidirectionally map data to the latent space. However, the encoder of BigBiGAN has difficulty in extracting hidden features from container images. FMGAN amends the structure by adding a hidden feature encoder and a frozen discriminator (i.e., the pre-trained DNN f_θ) to generate adversarial perturbations highly robust against variable sensitive types. That is, FMGAN consists of a generator, two encoders, and a joint discriminator module composed of four discriminators. Moreover, the contrastive loss is used to guide the newly added encoder to extract the common features relevant to sensitive types.

Adversarial Inference. As soon as the FMGAN training is completed, the mobile device locally keeps the hidden feature encoder and the generator to perturb container images in the inference phase. As in IGHO, only sensitive images need to be protected in IGHO$^+$. Specifically, the mobile device samples a cover image from the processed cover dataset; hides the sensitive image by using LSB or NNS, and perturbs the container image with the well-trained FMGAN before feature extraction; The cloud performs inference on the features received from the mobile device; and sends back the prediction results.

Comparison between IGHO$^+$ and GHOST$^+$. Both solutions protect inference privacy by steganography and add deliberate perturbations into inference instances for high accuracy without altering the pre-trained DNN f_θ. The main differences lie in two aspects: (1) Compared with GHOST$^+$, IGHO$^+$ adopts an extra preprocessing step to produce cover images of fuzzy feature boundaries. (2) Unlike GHOST$^+$ training a GAN to generate adversarial perturbations on container images, IGHO$^+$ trains a FMGAN to generate perturbations robust against variable sensitive types. The effectiveness of FMGAN is validated via ablation experiments. For example, for CIFAR-10 dataset under the hiding ratio is 4:6, IGHO$^+$ spends about 42min to get a well-trained FMGAN, which can be used to raise the inference accuracy of GHOST$^+$ from 43.13% to 68.88% without the help of preprocessing methods.

5 IGHO: The Improved Version of GHOST

In this section, we will introduce the improvement details of IGHO, which preprocesses cover images of GHOST to mitigate their disturbance to inference accuracy. Specifically, two kinds of preprocessing approaches are designed with a trade-off between accuracy and efficiency: Feature synthesis produces cover images of fused sensitive and public features, aiming to narrow the gap between cover and hidden features, while pixel synthesis creates cover images by optionally splicing multiple public images together, trying to weaken salient cover features.

5.1 Feature Synthesis

A naive solution to minimize the difference between cover and hidden features is to use sensitive images directly as cover images.

Algorithm 1 Feature Synthesis

Input: Network $\mathscr{M}_\theta$, public dataset $\mathbf{D}_p$, sensitive dataset $\mathbf{D}_s$
OutPut: Processed cover dataset $\mathbf{D}_{pc}$
{Batch size bs, loss bound σ, the number of epoches n_{epoch}, the number of iterations in each epoch n_{iter}}
for $i = 0$ to n_{epoch} **do**
  Initialize perturbation δ_i with standard normal distribution
  Construct the i-th batch of images D_{p_i} from $\mathbf{D}_p$
  Construct the i-th batch of images D_{s_i} from $\mathbf{D}_s$
  for $j = 0$ to n_{iter} **do**
    Calculate the similarity between each pair of images (P_x, S_y) from D_{p_i} and D_{s_i} with Eq. 6
    Perturb each sample $S_y \in D_{s_i}$ with δ_i: $S_y' \leftarrow S_y + \delta_i$
    Update D_{s_i}: Replace each sample S_y with S_y'
    Optimize δ_i with the loss function $\mathscr{L}_{pert}$ of Eq. 7
    if loss $\mathscr{L}_{pert} > \sigma$ **then** continue;
    else break;
Copy the updated sensitive dataset $\mathbf{D}_s$ to $\mathbf{D}_{pc}$

But this would enable the cloud to recover original sensitive images by feature inversion, exposing user privacy. Instead, we exploit feature synthesis to produce cover images of fused public and sensitive features. Inspired by the work of [51], we first find the image P_{x*} most similar to a sensitive image S_y from a public dataset; and then craft deliberate perturbations enabling S_y close to P_{x*} in the feature space. Specifically, the similarity between a public image P_x and a sensitive image S_y is determined as follows:

$$sim(P_x, S_y) = \|\mathscr{M}(P_x) - \mathscr{M}(S_y)\|_2^2, \tag{6}$$

where $\mathscr{M}(\cdot)$ represents employing mobile-side network $\mathscr{M}_\theta$ to extract features, and $\|\mathscr{M}(P_x) - \mathscr{M}(S_y)\|_2$ is the Euclidean distance between the features of images P_x and S_y.

The working process of feature synthesis is summarized in Algorithm 1, where a processed cover dataset $\mathbf{D}_{pc}$ is a copy of an updated sensitive dataset. Initially, a public dataset $\mathbf{D}_p$ and a sensitive dataset $\mathbf{D}_s$ are divided into n_{epoch} batches of size bs, denoted by $\{D_{p_i}\}_{i=1}^{n_{epoch}}$ and $\{D_{s_i}\}_{i=1}^{n_{epoch}}$, respectively. In i-th epoch, a sensitive image $S_y \in D_{s_i}$ is updated by adding the optimal perturbation δ_i, i.e., $S_y' = S_y + \delta_i$, and δ_i is generated by minimizing the following loss function over multiple iterations:

$$\mathscr{L}_{pert} = arg \min_{\delta_i} \sum_{k=1}^{bs} \|\mathscr{M}_\theta(P_{x*}) - \mathscr{M}_\theta(S_y + \delta_i)\|_2^2, \tag{7}$$

where $P_{x*} \in D_{p_i}$ is the closest public image of the sensitive image $S_y \in D_{s_i}$ according to the feature similarity computed with Eq. 6 (i.e., $sim(P_{x*}, S_y) = \min\{sim(P_x, S_y)\}_{P_x \in D_{p_i}}$). The iteration terminates until the loss $\mathscr{L}_{pert}$ reaches a predetermined bound σ that is a hyperparameter adjusted for different datasets. Contrary to the training process in DNN, the number of epoches n_{epoch} is calculated by $|\mathbf{D}_s|/bs$, and the number of iterations n_{epoch} is determined according to actual needs. After feature synthesis, cover images are mixtures of public and sensitive features, from which it is hard for an attacker to recover valuable data.

5.2 Pixel Synthesis

Feature synthesis is effective in improving accuracy in privacy-preserving inference; but consumes a substantial amount of time in preprocessing the cover dataset. For efficiency reasons, pixel synthesis generates a cover image by randomly slicing two public images together. Since each cover image is made of a mix of random public images, the prominent features of cover images are weakened. Algorithm 2 summarizes the working process of pixel synthesis: Given a public dataset $\mathbf{D}_p$, the mobile device randomly chooses two public images, P_x and P_y, from $\mathbf{D}_p$, cuts each of them into two parts according to a randomly chosen cutting ratio r_c, and generates a processed cover image C_x (resp. C_y) by concatenating the first part of P_x (resp. P_y) and the second part of P_y (resp. P_x) together. In addition, in order to diversify surface features, a portion of public images is rotated before the slice-splice process.

Algorithm 2 Pixel Synthesis

Input: Public dataset $\mathbf{D}_p$, flip probability p_f, dataset size s
OutPut: Processed cover dataset $\mathbf{D}_{pc}$
Rotate each image $P_k \in \mathbf{D}_p$ 90 degrees with probability p_f
while $|\mathbf{D}_{pc}| < s$ **do**
  Randomly choose two images P_x and P_y from $\mathbf{D}_p$
  Randomly select a cutting ratio r_c from $[0, 1]$
  Cut P_x into two parts P_{x_1} and P_{x_2} according to r_c
  Cut P_y into two parts P_{y_1} and P_{y_2} according to r_c
  Generate a cover image C_x by splicing P_{x_1} and P_{y_2} together
  Generate a cover image C_y by splicing P_{y_1} and P_{x_2} together
  Add cover images C_x and C_y into cover dataset $\mathbf{D}_{pc}$

6 IGHO$^+$: The Improved Version of GHOST$^+$

In this section, we will describe the working process of IGHO$^+$ in detail, which improves the practicality of GHOST$^+$ by combining preprocessing and FMGAN. As the preprocessing step has been explained in Section 5, this section focuses on the construction details of FMGAN. To better understand the benefits of FMGAN, we first provide a strawman construction that simply trains an improved GAN structure for each sensitive type.

6.1 A Strawman Construction

Due to the limited learning ability of GAN, a natural way to avoid accuracy degradation is to train multiple GANs; so that a generator is only responsible for generating adversarial perturbations for one sensitive type. The original GAN structure in GHOST$^+$ is constituted by a generator $\mathscr{G}$ and a frozen discriminator f_θ, where f_θ is the pre-trained network that will not be updated in the training process. The generator $\mathscr{G}$ takes a container image $\widetilde{C}$ as input and generates a perturbed image $\widetilde{C} + \mathscr{G}(\widetilde{C})$. The goal of $\mathscr{G}$ is to generate adversarial perturbations for container images such that the pre-trained network f_θ outputs adversary-selected results. To this end, GHOST$^+$ first defines the following loss to minimize the average distance between the predictions and the expected labels:

$$\mathscr{L}_{adv} = \mathbb{E}[J(f_\theta(\widetilde{C} + \mathscr{G}(\widetilde{C})), l_H)], \tag{8}$$

where l_H is the ground-truth label of the hidden image H and $f_\theta(\widetilde{C} + \mathscr{G}(\widetilde{C}))$ is the predicted label. To bound the magnitude of perturbations with threshold c, GHOST$^+$ defines the loss function $\mathscr{L}_{hinge}$ by adding a soft hinge loss on the L2 norm:

$$\mathscr{L}_{hinge} = \mathbb{E}[\max\{0, \|\mathscr{G}(\widetilde{C})\|_2 - c\}]. \tag{9}$$

Therefore, the total loss function can be expressed as:

$$\mathscr{L}_{total} = \lambda \cdot \mathscr{L}_{adv} + \mathscr{L}_{hinge}, \tag{10}$$

where λ controls the relative importance of each loss. During our experiments, we found that the inference accuracy can be further

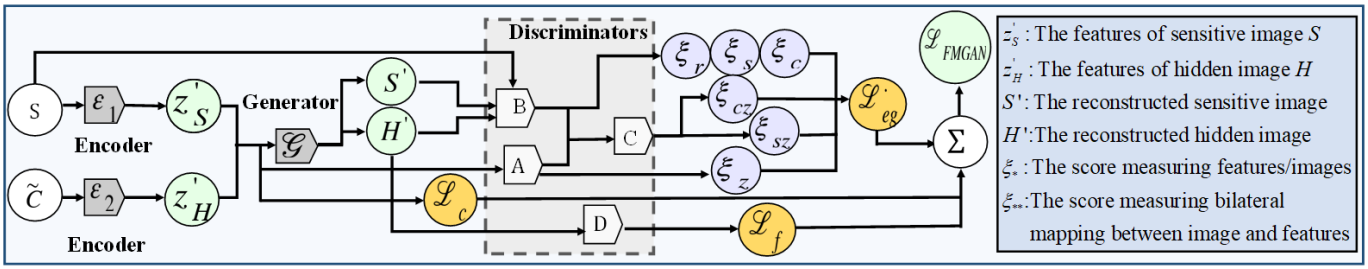


Fig. 7: The structure of FMGAN. Score details: (1)ξ_z : The score of features z'_H or z'_S; (2)ξ_r, ξ_s, ξ_s : The scores of images S, S' and H'; (3)ξ_{cz}: The score of the map between image H' and features z'_H or z'_S; (4)ξ_{sz}: The score of the map between image S and features z'_H or z'_S;

promoted by applying an extra discriminator $\mathscr{D}$ to distinguish the perturbed data $\widetilde{C}+\mathscr{G}(\widetilde{C})$ from the original input $\widetilde{C}$. Instead, the strawman construction uses the following loss function:

$$\mathscr{L}^*_{total} = \lambda_1 \cdot \mathscr{L}_{adv} + \lambda_2 \cdot \mathscr{L}_{hinge} + \lambda_3 \cdot \mathscr{L}_{gan}, \tag{11}$$

where $\mathscr{L}_{gan}$ is the adversarial loss [34]. The generator $\mathscr{G}$ and the discriminator $\mathscr{D}$ can be obtained by solving the minmax game.

However, this solution assumes that the user has prior knowledge of the types of inference samples, and thus is only suitable for specific applications, where the user knows the first-level labels of inference samples and the pre-trained DNN performs downstream classification tasks. For example, a patient utilizes two GANs to perturb container images embedded with the lung CT image and the abdominal X-ray image, respectively, so that the on-line diagnostic DNN outputs the final diagnosis results.

6.2 The Details of FMGAN

To overcome the limitation of the strawman construction, IGHO$^+$ replaces the GAN with a novel FMGAN that generates adversarial perturbations highly robust against variable sensitive types. As shown in Fig. 7, FMGAN as an extension of BigBiGAN consists of a generator $\mathscr{G}$, two encoders ε_1 and ε_2, and a joint discriminator module $\mathscr{D}$ composed of four discriminators A, B, C, and D. The main differences from the BigBiGAN structure lie in the following aspects: (1) A hidden feature encoder ε_2 is added to extract hidden features from container images; (2) The pre-trained DNN f_θ is used as the frozen discriminator D aiming to distinguish between the original input and the image reconstructed by the generator $\mathscr{G}$; (3) To prevent privacy leakage from the reconstructed images, the contrastive loss is used to guide the encoder ε_2 to extract the common semantic features of hidden images.

In general, the encoder ε_1 just like in BigBiGAN takes a sensitive image S as the input and outputs the semantic features z'_S, i.e., $z'_S \leftarrow \varepsilon_1(S)$, but the hidden feature encoder ε_2 takes a container image $\widetilde{C}$ as the input and tries to learn z'_H, the common semantic features of hidden image H, i.e., $z'_H \leftarrow \varepsilon_2(\widetilde{C})$. Given the extracted features z'_S and z'_H, the generator $\mathscr{G}$ aims to generate the corresponding images, i.e., $S' \leftarrow \mathscr{G}(z'_S)$ and $H' \leftarrow \mathscr{G}(z'_H)$. The discriminator module $\mathscr{D}$ aims to differentiate the cooperative output of ε_1 and $\mathscr{G}$ (i.e., the surface feature z'_S and the reconstructed sensitive image S'), from that of ε_2 and $\mathscr{G}$ (i.e., the hidden features inside container images z'_H and the reconstructed hidden images H'). Specifically, the discriminator A takes features z'_S and z'_H as inputs and outputs the score ξ_z, the discriminator B takes images S, S', and H' as inputs and outputs the scores ξ_s, ξ_r, and ξ_c, respectively, the discriminator C takes score pairs (ξ_s, ξ_z) and (ξ_c, ξ_z) as inputs and outputs the joint scores ξ_{sz} and ξ_{cz}, respectively, and the discriminator D takes the reconstructed image H' as input and outputs the relevant label. The discriminator D (i.e., the pre-trained DNN f_θ) is fixed while the remaining discriminators are trained by the following loss function:

$$\begin{aligned}\mathscr{L}^*_d = {} & \mathbb{E}_{fake}[h(-\xi_c) + h(-\xi_z) + h(-\xi_{cz})] \\ & + \mathbb{E}_{true}[h(\xi_s) + h(\xi_z) + h(\xi_{sz}) + h(\xi_r)],\end{aligned} \tag{12}$$

where $h(t) = \max\{0, 1-t\}$. The main difference from the discriminator loss of BigBiGAN (Eq. 3) is that $\mathbb{E}_{true}[\cdot]$ and $\mathbb{E}_{fake}[\cdot]$ represent the average score of features and images generated by encoder/generator pair $(\varepsilon_1, \mathscr{G})$ and pair $(\varepsilon_2, \mathscr{G})$, respectively.

Given a sensitive image S and a container image $\widetilde{C}$ that conceals a hidden image H, the goal of the encoders and generator is to produce the adversarial perturbation on $\widetilde{C}$ such that (1) $f_\theta(\mathscr{G}(\varepsilon_2(\widetilde{C}))) = l_H$, where l_H is the label of the hidden image H; (2) The discriminator module $\mathscr{D}$ cannot distinguish between the outputs of $(\varepsilon_1, \mathscr{G})$ and $(\varepsilon_2, \mathscr{G})$; (3) $\varepsilon_2(\widetilde{C})$ extracts the most common features of hidden images. To this end, the encoders and the generator are trained by the following loss function:

$$\mathscr{L}_{FMGAN} = \lambda_1 \cdot \mathscr{L}_f + \lambda_2 \cdot \mathscr{L}^*_{eg} + \lambda_3 \cdot \mathscr{L}_c, \tag{13}$$

where λ_i controls the relative importance of each loss and can be dynamically adjusted to strike a good balance among three loss items. In the above equation, $\mathscr{L}_f$ defined in Eq. 14 is used to minimize the average difference between the predicted results and the expected labels, $\mathscr{L}^*_{eg}$ defined in Eq. 15 aims to fool the discriminator module with the outputs of the generator/encoder pair $(\mathscr{G}, \varepsilon_2)$, and $\mathscr{L}_c$ defined in Eq. 16 is the global contrastive loss guiding the encoder ε_2 to extract the hidden features most relevant to sensitive types.

$$\mathscr{L}_f = \mathbb{E}[J(f_\theta(\mathscr{G}(\varepsilon_2(\widetilde{C}))), l_H)]. \tag{14}$$

$$\mathscr{L}^*_{eg} = \mathbb{E}_{fake}[-\xi_c - \xi_z - \xi_{cz}] + \mathbb{E}_{true}[\xi_s + \xi_z + \xi_{sz} + \xi_r]. \tag{15}$$

$$\mathscr{L}_c = \sum_{i=0}^{Num} CL(Y, z'_S, z'_H). \tag{16}$$

In Eq. 16, Num is the number of relationships between sensitive and hidden types, z'_S and z'_H are the features extracted by encoders ε_1 and ε_2, respectively, Y is assigned the value of 0 or 1 depending on if the relevant images S and H belong to the same type or not, and CL as the contrastive loss is calculated with Eq. 17:

$$\begin{aligned}CL(Y, z'_S, z'_H) = {} & \frac{1}{2}(1-Y) \cdot \|z'_S - z'_H\|_2^2 \\ & + \frac{1}{2} Y \cdot \max\{0, m - \|z'_S - z'_H\|_2^2\},\end{aligned} \tag{17}$$

where $\|z'_S - z'_H\|_2$ is the Euclidean distance between features z'_S and z'_H, and m is used to limit the similarity between images of different types. In summary, the contrastive loss aims to make images of the same type similar in the feature space, while widening the gap between the features of images of different types.

Algorithm 3 Training a FMGAN

Input: Sensitive dataset $\mathbf{D}_s$, container dataset $\widetilde{\mathbf{D}}_c$, a generator $\mathscr{G}$, two encoders $\varepsilon_1, \varepsilon_2$, four discriminators A, B, C, D
Output: Updated generator $\mathscr{G}'$, updated encoders $\varepsilon_1', \varepsilon_2'$, updated discriminators A', B', C'
{Parameters: Batch size bs, the number of epoches n_{epoch}, the number of iterations in each epoch n_{iter}}

for $i = 0$ to n_{epoch} **do**
  Construct the i-th batch of images D_{s_i} from $\mathbf{D}_s$
  Construct the i-th batch of images $D_{\widetilde{c}_i}$ from $\widetilde{\mathbf{D}}_c$
  for $j = 0$ to n_{iter} **do**
    for each image $S \in D_{s_i}$ **do**
      Extract surface features and reconstruct the image: $z'_S \leftarrow \varepsilon_1(S)$; $S' \leftarrow \mathscr{G}(z'_S)$
    for each image $\widetilde{C} \in D_{\widetilde{c}_i}$ **do**
      Extract hidden features and reconstruct the image: $z'_H \leftarrow \varepsilon_2(\widetilde{C})$; $H' \leftarrow \mathscr{G}(z'_H)$
    Discriminators A, B, C: Calculate the relevant scores
    Update discriminators with loss $\mathscr{L}_d^*$ in Eq. 12
    Discriminator D: Calculate the loss $\mathscr{L}_f$ with Eq. 14
    Calculate joint loss $\mathscr{L}_{eg}^*$ with Eq. 15
    Calculate contrastive loss $\mathscr{L}_c$ with Eq. 16
    Update $\mathscr{G}$, ε_1, and ε_2 with loss function in Eq. 13

As shown in Algorithm 3, a FMGAN is trained by using a sensitive dataset $\mathbf{D}_s$ and a container dataset $\widetilde{\mathbf{D}}_c$ based on the mini-batch SGD algorithm. In each batch, the encoder ε_1 and generator $\mathscr{G}$ cooperate to output the surface features z'_S and the reconstructed image S' for a sensitive image S; while the encoder ε_2 and generator $\mathscr{G}$ cooperate to output the hidden features z'_H and the reconstructed image H' for a container image $\widetilde{C}$. The discriminators A, B, and C takes these outputs and original images as input to calculate relevant scores, which will be used to calculate the discriminator loss $\mathscr{L}_d^*$ and joint loss $\mathscr{L}_{eg}^*$. The hidden image H' reconstructed by the generator $\mathscr{G}$ is regarded as the perturbed container image, which will be fed into the pre-trained network f_θ to obtain the loss $\mathscr{L}_f$. The extracted features z'_S and z'_H are used to calculate the global contrastive loss $\mathscr{L}_c$. This training process aims to minimize losses $\mathscr{L}_f$, $\mathscr{L}_{eg}^*$ and $\mathscr{L}_c$ for the generator and encoders, while minimizing loss $\mathscr{L}_d^*$ for the discriminators A, B, and C. Once the FMGAN training process is finished, the updated generator $\mathscr{G}'$ and encoder ε_2' will be deployed on the mobile device.

7 Privacy Measurement

MI is a measure of the mutual dependence between two variables, which is widely used to quantify information leakage in steganography systems [33]. As the work in [30], we employ mutual information (MI) as a tool to analyze the privacy level achieved in our IGHO and IGHO$^+$ solutions. The MI between random variables X and $\widetilde{X}$ can be defined as:

$$\begin{aligned} I(X;\widetilde{X}) &= \mathbb{H}(X,\widetilde{X}) - \mathbb{H}(X|\widetilde{X}) - \mathbb{H}(\widetilde{X}|X) \\ &= \mathbb{H}[X] - \mathbb{H}[X|\widetilde{X}], \end{aligned} \tag{18}$$

where $\mathbb{H}(X)$ and $\mathbb{H}(\widetilde{X})$ are the marginal entropies, $\mathbb{H}(X|\widetilde{X})$ and $\mathbb{H}(\widetilde{X}|X)$ are the conditional entropies, and $\mathbb{H}(X,\widetilde{X})$ is the joint entropy of X and $\widetilde{X}$. In the work, we employ MI to measure the amount of leakage between the original input and the features extracted by the mobile-side network. Given a DNN $f_\theta = F_1 \circ F_2 \circ ... \circ F_L$, we assume that the mobile-side network $\mathscr{M}_\theta$ consists of the first K layers, and the cloud-side network $\mathscr{C}_\theta$ consists of the remaining $L-K$ layers. Let S denote an original sensitive image, and let C denote an original cover image. We use $\mathscr{S}(X,Y)$ to represent hiding image X into image Y with steganography, and $\mathscr{P}(\cdot)$ to denote the preprocessing operation to facilitate analysis.

7.1 The Privacy of IGHO

In IGHO, the mobile device first hides a sensitive image S into a processed cover image $\mathscr{P}(C)$ to form a container image $\widetilde{C}$; and then employs the mobile-side network $\mathscr{M}_\theta$ to extract features from $\widetilde{C}$. Therefore, the privacy of IGHO is defined as:

$$\begin{aligned} \Psi_K &= -I[S;\mathscr{M}_\theta(\widetilde{C})] \\ &= -I[S;\mathscr{M}_\theta(\mathscr{S}(S,\mathscr{P}(C)))]. \end{aligned} \tag{19}$$

From Eq. 19, it is easy to find out that the lower MI implies the higher level of privacy. Since the mobile-side network $\mathscr{M}_\theta$ is deterministic and frozen, Eq. 19 can be transformed into:

$$\begin{aligned} \Psi_K &= \mathbb{H}[\mathscr{M}_\theta(\widetilde{C})|S] - \mathbb{H}[\mathscr{M}_\theta(\widetilde{C})] \\ &= -\mathbb{H}[\mathscr{M}_\theta(\widetilde{C})] \\ &= -\mathbb{H}[F_K(F_{K-1}(...(F_1(\widetilde{C}))))] \\ &= -\mathbb{H}[F_K(F_{K-1}(...(F_1(\mathscr{S}(S,\mathscr{P}(C))))))]. \end{aligned} \tag{20}$$

The term $\mathbb{H}[\mathscr{M}_\theta(\widetilde{C})]$ controls the amount of information leaked to the cloud, which can be minimized by the joint efforts of preprocessing, steganography, and feature extraction. In the preprocessing step, cover images with fuzzy feature boundaries are generated. Then, the steganography method is utilized to hide sensitive images stealthily into processed cover images. Finally, the mobile-side network $\mathscr{M}_\theta$ performs pooling, ReLU and convolution operations to extract the abstract features of container images.

Let $\Psi_{ST} = -I[S;\widetilde{C}]$ denote the privacy level achieved by the steganography method. According to Data Processing Inequalities (DPI) [52], a lower bound on privacy can be derived as follows:

$$\Psi_K \geq \Psi_{K-1} \geq \Psi_{K-2} \geq ... \Psi_1 \geq \Psi_{ST}, \tag{21}$$

where Ψ_i denotes the privacy provided by the i-th layer ($1 \leq i \leq K$). Therefore, we draw the following conclusions: (1) IGHO offers better privacy protection than using steganography alone. (2) The deeper the partitioning point, the higher the privacy level can be reached. Compared with GHOST, IGHO preprocesses the cover dataset in addition. From the above analyses, we know that IGHO can achieve at least the same privacy protection as GHOST.

7.2 The Privacy of IGHO$^+$

IGHO$^+$ first generates a container image $\widetilde{C}$ as IGHO, then generates perturbed container image $\widetilde{C}'$ by the encoder/generator pair $(\varepsilon_2, \mathscr{G})$, and finally extracts features from $\widetilde{C}'$ with the mobile-side network $\mathscr{M}_\theta$. Let $X' = \mathscr{G}(\varepsilon_2(X))$ represent the cooperate output of $(\varepsilon_2, \mathscr{G})$. That is, the encoder ε_2 first extracts hidden features from an input image X so that the generator $\mathscr{G}$ can reconstruct an image X' from the extracted features. The privacy of IGHO$^+$ is defined as:

$$\begin{aligned} \Psi_K^+ &= -I[S;(\mathscr{M}_\theta(\widetilde{C}')] \\ &= -I[S;\mathscr{M}_\theta(\mathscr{G}(\varepsilon_2(\widetilde{C})))] \\ &= -I[S;\mathscr{M}_\theta(\mathscr{G}(\varepsilon_2(\mathscr{S}(S,\mathscr{P}(C)))))]. \end{aligned} \tag{22}$$

TABLE 3: Model Structure and Parameter Settings

Dataset	# of Images	# of Classes	Input Size	Model Architecture	Accuracy	$[\alpha, bs, n_{epoch}]$
MNIST	70,000	10	$28\times28\times1$	2Conv+2Pooling+2Dense	98.25	$[0.001, 256, 20]$
CIFAR-10	60,000	10	$32\times32\times3$	4Conv+2Pooling+4BN+4Dropout+3Dense	87.13	$[0.001, 128, 100]$
GTSRB	51,839	43	$32\times32\times3$	6Conv + 3Pooling +4Dropout+2Dense	96.21	$[0.001, 128, 50]$
SVHN	99,289	10	$32\times32\times3$	AlexNet [53]	91.79	$[5e^{-4}, 128, 50]$

TABLE 4: The Impact of Different Partitioning Points on Privacy (left) and Efficiency (right) under LSB and $\gamma = 4:6$

Dataset	Method	Partitioning Point K (MI)				
		K=1	K=2	K=3	K=4	K=5
MNIST	IGHO	1.7427	1.2139	0.9982	0.7246	0.6645
	IGHO$^+$	1.1400	0.9685	0.7821	0.6342	0.5127
CIFAR-10	IGHO	2.0611	1.7328	1.3346	1.2464	1.1834
	IGHO$^+$	2.2537	1.7158	1.2462	1.1928	1.0819
GTSRB	IGHO	1.7996	1.5521	1.2063	1.1837	1.1301
	IGHO$^+$	2.1127	1.6130	1.1428	1.0942	0.9761
SVHN	IGHO	1.9994	1.4886	1.2196	1.1893	1.0720
	IGHO$^+$	2.1053	1.5614	1.1964	1.1052	0.9346

Dataset	Metric	Partitioning Point K (Time overheads)				
		K=1	K=2	K=3	K=4	K=5
MNIST	Training the DNN (min)	50.61	54.42	59.8	66.08	73.35
	Feature Extraction(ms)	0.005	0.007	0.009	0.011	0.014
CIFAR-10	Training the DNN(min)	13.42	14.83	16.3	18.26	20.99
	Feature Extraction(ms)	0.09	0.18	0.22	0.26	0.31
GTSRB	Training the DNN(min)	15.15	16.65	18.44	22.13	24.36
	Feature Extraction(ms)	0.02	0.07	0.09	0.11	0.12
SVHN	Training the DNN(min)	18.79	21.35	23.46	25.57	29.41
	Feature Extraction(ms)	0.04	0.06	0.09	0.11	0.13

As IGHO$^+$ locally trains a GAN without altering the DNN, the training time is only related to IGHO. The feature extraction time is related to both solutions.

Based on the frozen network $\mathscr{M}_\theta$, Eq. 22 can be transformed into:

$$\begin{aligned}\Psi_K^+ &= -\mathbb{H}[\mathscr{M}_\theta(\widetilde{C}')] + \mathbb{H}[\mathscr{M}_\theta(\widetilde{C}')|S] \\ &= -\mathbb{H}[\mathscr{M}_\theta(\widetilde{C}')] \\ &= -\mathbb{H}[F_K(...(F_1(\mathscr{G}(\varepsilon_2(\mathscr{S}(S,\mathscr{P}(C))))))))].\end{aligned} \tag{23}$$

The term $\mathbb{H}[\mathscr{M}_\theta(\widetilde{C}')]$ controls the amount of exposed information. The information leakage can be minimized by the joint efforts of preprocessing, steganography, FMGAN, and feature extraction Let $\Psi_{ST} = -I[S;\widetilde{C}]$ and $\Psi_F = -I[S;\widetilde{C}']$ denote the privacy level provided by steganography and FMGAN, respectively. Based on the DPI theorem, we obtain the following lower bound for Ψ_K^+:

$$\Psi_K^+ \geq \Psi_{K-1}^+ \geq \Psi_{K-2}^+ \geq ...\Psi_1^+ \geq \Psi_F \geq \Psi_{ST}, \tag{24}$$

This equation implies that IGHO$^+$ can improve the privacy level of steganography by adding adversarial perturbations on container images. Compared with GHOST$^+$, IGHO$^+$ mainly adds the preprocessing step and replaces GAN with FMGAN to generate perturbations, achieving the same privacy protection. Besides, the encoder ε_2 is guided by contrastive loss to extract common hidden features, and thus sensitive information can be further protected.

8 Evaluation

In this section, we conduct experiments to evaluate the performance of our solutions. To validate the effectiveness, we conduct experiments on four real datasets; and make comparisons with the SOTA DP-based solution Shredder besides GHOST and GHOST$^+$. The reason for comparing it with Shredder is that Shredder is also built on top of the mobile-cloud collaborative framework, and measures the privacy-level by using the MI between the images reconstructed from extracted features and the original inputs.

8.1 Experimental Settings

Datasets. Our experiments are conducted on four datasets:MNIST, CIFAR-10, SVHN, and GTSRB. The experimental configurations are shown in Table 3. To train the DNN, we use Adam optimizer [54] under different hyperparameters, including the learning rate α, the batch size bs, and the number of epoches n_{epoch}. For all the datasets, 90% of the samples are used as the training set, and the rest are used as the testing set. For training convenience, all the images are normalized to a $[0,1]$ range and processed to proper size (e.g., the images in MNIST are of size 28×28, and the images in remaining datasets are of size 32×32). For each dataset, the images are classified into sensitive and public types.

TABLE 5: Comparison of Preprocessing Methods on CIFAR-10

Methods	LSB				NNS			
	1:9	2:8	3:7	4:6	1:9	2:8	3:7	4:6
FS +IGHO	**0.9510**	**0.9430**	**0.8751**	**0.8334**	**0.9840**	**0.9620**	**0.8563**	**0.8551**
PS +IGHO	0.9451	0.9095	0.8688	0.8128	0.9540	0.9055	0.8333	0.7820
FS +IGHO$^+$	0.9622	**0.8996**	**0.8026**	**0.7211**	0.9818	**0.9286**	**0.8292**	**0.7417**
PS +IGHO$^+$	**0.9922**	0.8635	0.7579	0.6888	**0.9987**	0.8934	0.7901	0.6964

Two steganography methods, LSB (the number of MSBs is 3) and NNS, are utilized to hide a sensitive image into a cover image. We use γ to denote the hiding ratio (i.e., the ratio between the number of sensitive types and the number of public types). As MNIST is the smallest dataset containing only 10 types of images, we set γ to $\{1:9, 2:8, 3:7, 4:6\}$ to cover all its image types.

Implementation Details. A server with NVIDIA GeForce RTX 3090TI GPU running CUDA V11.6 on Windows operating system (OS) and a laptop with NVIDIA GeForce MX230 off-the-shelf GPU running CUDA V10.0 on Windows OS are regarded as the cloud and the mobile device, respectively. To determine the partitioning point, we evaluate the privacy-level and efficiency of our solutions while varying the parameter K. The privacy level is measured by the MI between the image reconstructed from extracted features and the original image, while the efficiency is measured by the time overheads incurred in training the cloud-side network and extracting features for inference samples. From Table 4, we observe that the deeper the partitioning point is, the higher the privacy yield (i.e., the lower the MI), but the lower the efficiency (i.e., higher the time overheads). Besides, we find that the increasing trend of MI slows down, but that of time overheads is still obvious when K exceeds 3. Based on this, we fix the number of layers deployed on the mobile device with $K = 3$, which strikes a good balance between efficiency and privacy protection.

8.2 Accuracy and Efficiency

Both IGHO and IGHO$^+$ preprocess the cover dataset by feature synthesis (FS for short) or pixel synthesis (PS for short). In our experiments, we find that FS is more conducive to promoting inference accuracy but consumes longer processing time compared with PS. Table 5 shows the contrastive accuracy of the CIFAR-10 dataset. As for time overheads, FS takes about 40min, but PS costs only about 17.6s to preprocess CIFAR-10 dataset. Compared with IGHO, IGHO$^+$ has to spend a relatively long time training a FMGAN. To strike a good balance between accuracy and

efficiency, in the following experiments, we will use FS and PS to preprocess the cover images in IGHO and IGHO$^+$, respectively.

GHOST vs. IGHO. Both solutions mislead the cloud-side network to produce adversary-selected results by poisoning the training dataset. Therefore, the inference accuracy is tested for both public and sensitive samples. Let GHOST-C and IGHO-C denote the results of clean samples in related solutions. The comparison results are shown in Fig. 8 and Fig. 9. From these figures, we have the following observations: (1) In MNIST, all solutions perform the best and have only a minor accuracy loss as γ increases. This is because the images in MNIST have the simplest formats and structures. (2) NNS that supports hiding full-size images performs better than LSB with limited embedding capacity when experiments are conducted on CIFAR-10, GTSRB, and SVHN datasets. The reason for this exception is that the images in MNIST are too simple to hide much information. Besides, although NNS requires about 1.25h in the training process, it is very efficient for concealing images (about 2.25ms for one image). (3) All solutions have a downward trend in inference accuracy with the increasing value of γ, but IGHO is more robust against the variable sensitive types. The performance gain of IGHO is owing to the adoption of preprocessing step. (4) For SVHN and GTSRB datasets, the inference accuracy of IGHO is little influenced by γ. For instance, when γ increases from 1:9 to 4:6, the inference accuracy of both datasets only dropped by less than 0.8% and 0.55% for LSB and NNS, respectively. This is caused by the uneven data distribution of GTSRB and SVHN datasets: The amount of hidden sensitive features will not increase in proportion to the number of sensitive types. (5) As for clean samples, the inference accuracy remains stable in IGHO; but declined generally in GHOST. This is because the preprocessing step in IGHO makes cover images different from public images, and thus the retraining process has only a minor effect on learning public features.

GHOST$^+$ vs. IGHO$^+$. Let GHOST$^+$-S denote the strawman construction as described in Section 6.1. The comparison results are shown in Fig. 10 and Fig. 11. From these figures, we have similar observations as those in Fig. 8 and Fig. 9. For example, all solutions perform the best in MNIST, and NNS performs better than LSB in most cases. Furthermore, we also observe that GHOST$^+$-S trains a GAN for each sensitive type, performing the best in specific multi-level classification scenarios; IGHO$^+$ performs better than GHOST$^+$ with the increasing value of γ. For instance, when the hiding ratio of NNS increases from 1:9 to 4:6, the inference accuracy of the GTSRB dataset decreases from 97.74% to 61.65% in GHOST$^+$, but only decreases by 17.84% and 4.75% in IGHO$^+$ and GHOST$^+$-S, respectively. The performance gain of IGHO$^+$ is due to the adoption of FMGAN: The ablation experiments show the inference accuracy of IGHO$^+$ decreases by 10% when using FMGAN alone and by 27.91% when using preprocessing alone. In terms of time efficiency, IGHO$^+$ consumes a little more time in training FMGAN compared with GHOST$^+$, and the training time in both solutions increases as γ increases. For example, for MNIST dataset, it costs about 3.5min and 5min to train GAN and FMGAN when $\gamma = 1:9$, respectively, while it costs about 12min and 21min to train GAN and FMGAN when $\gamma = 4:6$, respectively. Once the training process is finished, the time for producing perturbations for both solutions is negligible.

By comparing the results in Fig. 8-Fig. 11, we know that unlike IGHO, the inference accuracy of IGHO$^+$ is negatively influenced by the increasing hiding ratio when using SVHN and GTSRB datasets. This is because FMGAN needs to learn the common features relevant to sensitive types, and thus the amount of information that needs to be learned increases as γ increases. In addition, we find that: (1) The inference accuracy in all solutions is even higher than the baseline accuracy given a small hiding ratio; (2) IGHO that resembles the process of training a poisoned network performs better than IGHO$^+$ that simulates GAN-based adversarial samples. The reason is that the accuracy in our solutions, in a sense, can be regarded as the attack success rate (ASR) of adversarial attacks. That is, backdoor attacks that deliberately modify the target DNN generally have higher ASR than adversarial samples that keep the DNN intact.

IGHO and IGHO$^+$ vs. Shredder. To better show the effectiveness of our solutions, we also make comparisons with the DP-based solution, Shredder, which trains a local network to learn the maximum perturbations with a trade-off in accuracy. Specifically, the comparison is conducted under the following metrics: (1) The accuracy (ACC) of protected samples. (2) The privacy-level achieved, which is measured by MI. (3) Preprocessing time, which is classified into FS time (for IGHO) and PS time (for IGHO$^+$). (4) Training time, which is classified into cloud-side training time (for IGHO) and local training time (for IGHO$^+$ and Shredder). (5) Inference time, which is the full time between accepting an original sample as input and outputting the final prediction.

As the accuracy and training time of IGHO and IGHO$^+$ are affected by the value of γ, we provide corresponding results under varied γ values in Table 6. From this table, we can observe that: (1) As for accuracy, IGHO always outperforms Shredder, and IGHO$^+$ performs better than Shredder when the hiding ratio is small. When γ exceeds $3:7$, the accuracy of IGHO$^+$ is slightly lower than that of Shredder. (2) Our solutions always provide a higher level of privacy than Shredder. (3) In terms of training time, our solutions are more efficient than Shredder, and IGHO$^+$ performs best. Note that even plus the preprocessing time, our solutions still perform better than Shredder. (4) As for inference time, our solutions always obtain the predictions of protected samples within 1ms, but the time overhead in Shredder is over five times as much as those of our solutions for complicated datasets. The above observations confirm that our solutions can strike a good balance among privacy, efficiency, and accuracy.

8.2.1 *Effectiveness of Privacy Protection*

The privacy assured by our solutions is measured by the invisibility of hidden sensitive images, the information leaked to attackers, and the defense efficacy against feature inversion attacks.

Invisibility. Two perceptual metrics, peak signal to noise ratio (PSNR) and structural similarity (SSIM) [55] are adopted to quantify the invisibility of hidden images. PNSR is a term to quantify the pixel difference between two images. Given two images, X and Y, the relevant PNSR is computed as:

$$PSNR = [10 \cdot \log_{10}(\frac{MAX_I^2}{MSE})], \tag{25}$$

where MAX_I^2 is the maximum pixel value, and MSE as the mean square error is calculated by $\frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[X(i,j)-Y(i,j)]^2$ with m and n representing the number of pixel rows and columns in an image, respectively. SSIM can be used for measuring the similarity between images X and Y, which is computed as follows:

$$SSIM(x,y) = [l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma}], \tag{26}$$

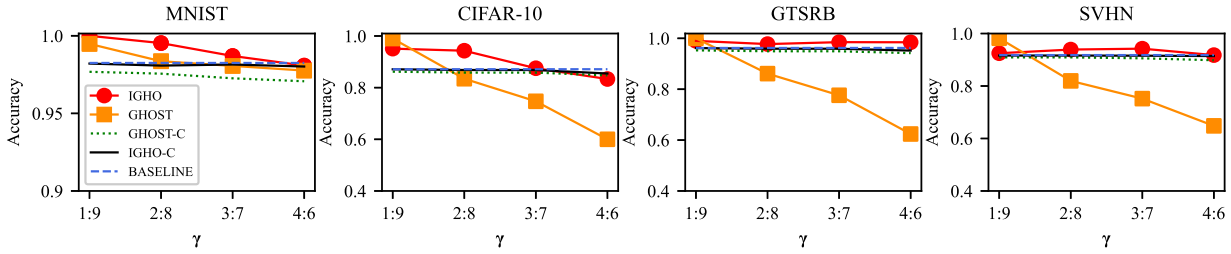


Fig. 8: The comparison results of IGHO and GHOST (LSB). BASELINE denotes the accuracy of original DNN.

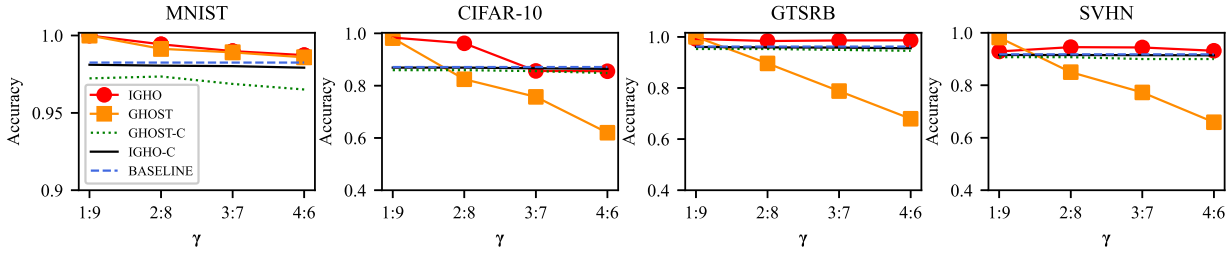


Fig. 9: The comparison results of IGHO and GHOST (NNS). BASELINE denotes the accuracy of original DNN.

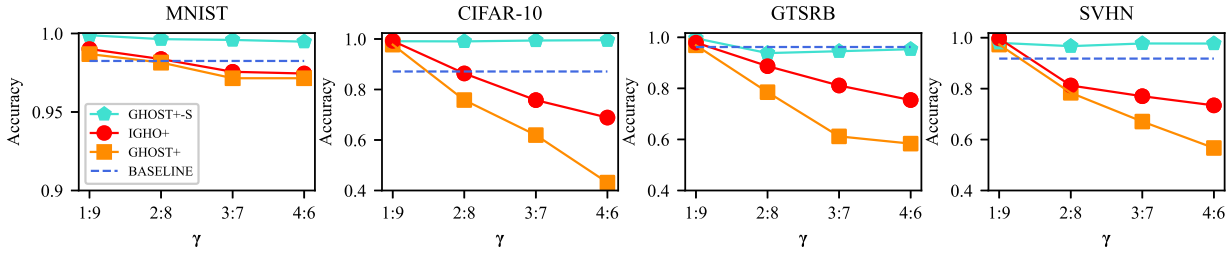


Fig. 10: The comparison results of IGHO$^+$ and GHOST$^+$ (LSB). BASELINE denotes the accuracy of original DNN.

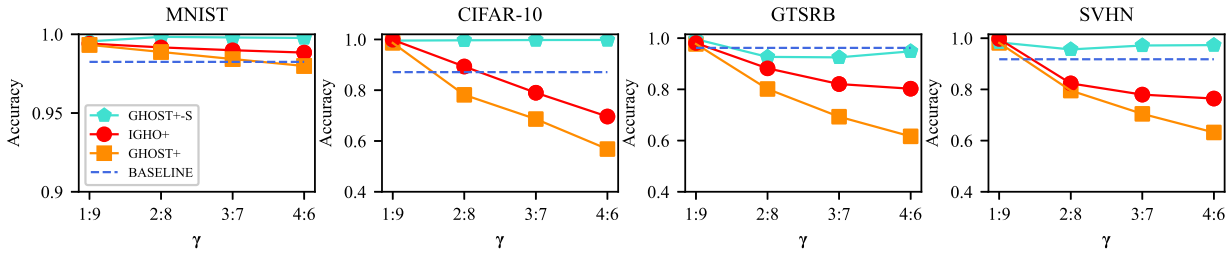


Fig. 11: The comparison results of IGHO$^+$ and GHOST$^+$ (NNS). BASELINE denotes the accuracy of original DNN.

where $l(x,y)$, $c(x,y)$, and $s(x,y)$ describe the similarity between two images from the perspectives of luminance, contrast and structure, and α, β, and γ control the importance of three components.

Here, we consider a strong attacker that can obtain the original container images from both IGHO and IGHO$^+$. Table 7 shows the average (Ave) PSNR/SSIM values between cover images and container images based on four benchmarks, where BASELINE denotes the best results in GHOST and GHOST$^+$. From this table, we know that it is hard for the observer to differentiate between cover images and container images. Therefore, our IGHO and IGHO$^+$ solutions can protect data privacy in the dark.

Information Leakage. We use MI to measure the amount of leaked information. Specifically, we consider two types of attackers: (1) A normal attacker that reconstructs images by launching write-box feature inversion attacks. Therefore, MI is calculated between reconstructed images and hidden images. (2) A strong attacker that can obtain the images before feature extraction. Hence, MI is calculated between the (perturbed) container images and hidden images. Table 8 shows the comparison results in the normal attacker model regarding four categories of images based on datasets MNIST and CIFAR-10. From this table, we know that: (1) Our solutions leak less information compared with GHOST and GHOST$^+$; (2) The solutions based on NNS leak more information than those based on LSB; (3) The MI calculated from different categories varies, and MNIST can better protect sensitive information than CIFAR-10. Table 9 shows the comparison results in the strong attacker model based on four benchmarks. Besides the similar observations in Table 8, we also find that IGHO$^+$ and GHOST$^+$ that add perturbations on container images leak less information than IGHO and GHOST. Especially, IGHO$^+$ can well protect sensitive data even in the face of powerful attackers.

Defense Efficacy. In white-box setting, we assume that the attacker has the knowledge of the mobile-side network $\mathscr{M}_\theta$; and tries to recover the hidden images by the features extracted from $\mathscr{M}_\theta$. Fig. 12 and Fig. 13 visualize the effectiveness of feature inversion attacks on the MNIST and CIFAR-10 datasets. From these figures, we know that our solutions not only have high

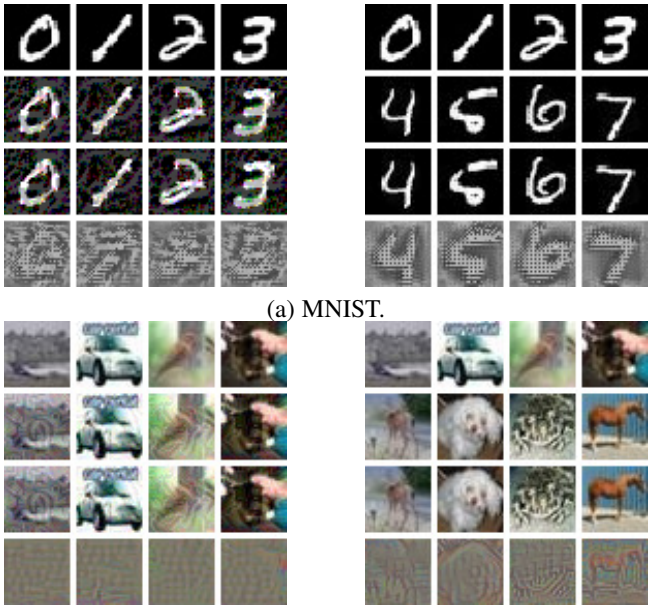
TABLE 6: Comparison between Our Solutions (under LSB) and Shredder (under Laplace Noises of Scale 20)

Method	Metric	MNIST	CIFAR-10	GTSRB	SVHN
IGHO	ACC (%)	100 / 99.53 / 98.70 / 98.08	95.10 / 94.30 / 87.51 / 83.34	99.19 / 98.38 / 98.61 / 98.65	98.89 / 94.53 / 94.41 / 93.15
	MI	0.9982	1.3346	1.2063	1.2196
	FS (min)	7.93 / 15.86 / 24.25 / 32.11	7.95 / 22.99 / 31.20 / 39.70	6.87 / 13.78 / 18.67 / 27.56	6.21 / 11.72 / 17.95 / 24.76
	Training Time (min)	2.37 / 7.81 / 39.75 / 59.8	11.23 / 14.17 / 15.62 / 16.3	9.21 / 12.72 / 14.61 / 18.44	8.35 / 14.28 / 18.94 / 23.46
	Inference Time (ms)	0.010	0.33	0.11	0.14
IGHO$^+$	ACC (%)	99.01 / 98.37 / 97.56 / 97.45	99.22 / 86.35 / 75.79 / 68.88	97.94 / 88.72 / 81.13 / 75.43	99.67 / 81.22 / 77.02 / 74.32
	MI	0.7821	1.2462	1.1428	1.1964
	PS (s)	5.14 / 10.38 / 15.75 / 20.54	4.4 / 8.9 / 13.5 / 17.6	3.8 / 7.69 / 11.66 / 15.21	7.28 / 14.73 / 22.34 / 29.12
	Training Time (min)	4.91 / 8.75 / 16.65 / 20.69	14.40 / 15.31 / 41.20 / 41.71	34.49 / 47.45 / 70.61 / 121.55	26.28 / 26.75 / 39.83 / 48.71
	Inference Time (ms)	0.013	0.36	0.15	0.17
Shredder	ACC (%)	97.15	78.16	78.52	78.29
	MI	1.0484	1.7056	1.2185	1.1757
	Training Time (min)	84	97.8	90.6	125.4
	Inference Time (ms)	0.077	2.01	2.67	2.19

TABLE 7: The Invisibility of Hidden Images

Dataset	PNSR/SSIM(Ave) / LSB			PNSR/SSIM(Ave) / NNS		
	BASELINE	IGHO	IGHO$^+$	BASELINE	IGHO	IGHO$^+$
MNIST	39/0.99	40/1.0	40/0.99	36/0.99	40/0.98	41/0.99
CIFAR-10	41/0.99	41/1.0	41/1.0	36/0.99	36/0.99	37/0.99
GTSRB	37/0.99	38/0.99	39/0.99	33/0.98	34/0.98	33/0.98
SVHN	43/0.99	43/1.0	43/1.0	36/0.99	37/0.99	38/0.99

The images are invisible to naked eyes when PNSR$\geq$ 30 and SSIM $\approx$1.

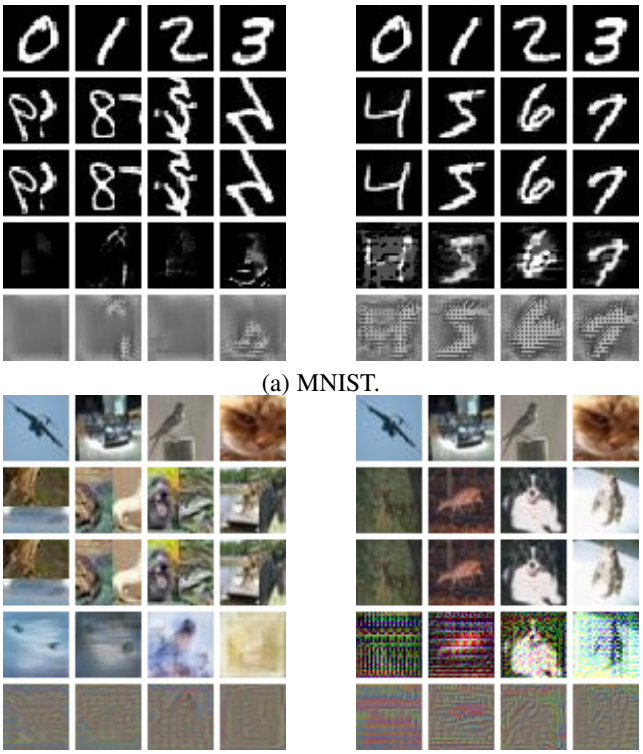


(a) MNIST.

(b) CIFAR-10.

Fig. 12: The defense efficacy of IGHO and GHOST. The left-side and right-side images belong to IGHO and GHOST, respectively. The 1^{st} row shows the sensitive images, the 2^{nd} row shows the cover images, the 3^{rd} row shows the container images output by NNS, and the 4^{th} row shows the reconstructed images.

robustness against feature inversion attacks, but also outperform the SOTA GHOST and GHOST$^+$ solutions. In addition, we conduct ablation experiments to show the influence of contrastive loss (CL) on preserving privacy of sensitive images. As shown in Fig. 14, images generated by FMGAN without CL contain sufficient sensitive information, while those generated with CL contain only common features relevant to sensitive types. Hence, contrastive loss helps enhance the privacy protection of IGHO$^+$.



(a) MNIST.

(b) CIFAR-10.

Fig. 13: The defense efficacy of IGHO$^+$ and GHOST$^+$. The left-side and right-side images belong to IGHO$^+$ and GHOST$^+$, respectively. The 1^{st} row shows the sensitive images, the 2^{nd} row shows the cover images, the 3^{rd} row shows the container images produced by NNS, the 4^{th} row shows the perturbed container images, and the 5^{th} row shows the reconstructed images.

9 Conclusion

In this paper, we propose two private deep learning solutions, IGHO and IGHO$^+$, aiming to improve the practicality of GHOST and GHOST$^+$. Specifically, a preprocessing step is adopted to generate cover images by either feature synthesis or pixel synthesis. Moreover, we propose a novel FMGAN structure to generate more robust adversarial perturbations. Experimental results demonstrate that our solutions have high applicability to mobile-cloud deep learning applications; since they not only ensure high inference accuracy in the case of a high hiding ratio, but also incur only a minor increase in computation and communication overheads. As part of our future work, we will further explore the feasibility of steganography in preserving inference privacy; and try to extend our solutions to multimodal deep learning scenarios.

TABLE 8: The MI in Normal Attacker Model

Methods	MNIST / LSB				MNIST / NNS				CIFAR-10 / LSB				CIFAR-10 / NNS			
	0	1	2	3	0	1	2	3	airplane	automobile	bird	cat	airplane	automobile	bird	cat
GHOST	1.1045	0.8176	1.1917	1.1784	1.3011	0.8317	1.2233	1.2318	1.3776	1.9596	1.7893	1.9276	1.4180	2.0513	1.9977	1.9985
IGHO	1.0434	0.7479	1.0921	1.1095	1.3000	0.9019	1.2935	1.1445	1.0952	1.6311	1.2206	1.3913	1.1558	1.6569	1.2431	1.4581
GHOST$^+$	1.0437	0.8001	1.1755	1.1741	1.1755	0.8253	1.2113	1.2013	1.2130	1.9861	1.7916	1.8317	1.2295	1.9887	1.8006	1.8370
IGHO$^+$	0.7404	0.6127	0.6097	1.1657	0.7667	0.6234	0.6158	1.1421	0.9040	1.5741	1.1418	1.3649	0.9133	1.6517	1.2315	1.3769

TABLE 9: The MI in Strong Attacker Model

Dataset	MI(Ave) / LSB				MI(Ave) / NNS			
	GHOST	IGHO	GHOST$^+$	IGHO$^+$	GHOST	IGHO	GHOST$^+$	IGHO$^+$
MNIST	1.8157	1.7936	1.7844	0.5411	1.9936	1.8973	1.7921	0.5810
CIFAR-10	2.9934	2.8817	2.8973	2.8348	2.9612	2.8983	2.8977	2.8347
GTSRB	2.5641	2.4726	2.4513	1.9129	2.5211	2.4625	2.4532	1.9091
SVHN	2.3119	2.3021	2.3016	1.8096	2.3514	2.3030	2.3030	1.8095

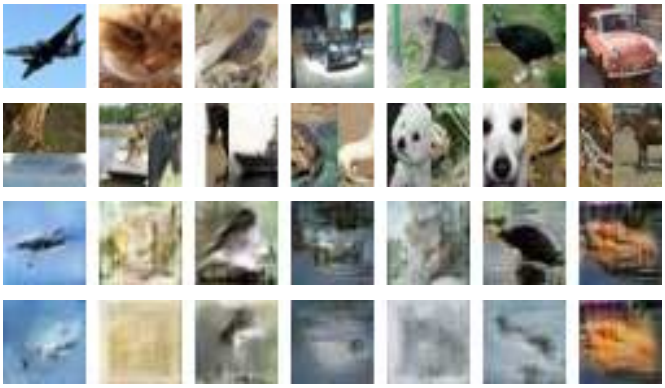


Fig. 14: The impact of contrastive loss (CL) on CIFAR-10. The 1^{st} row shows sensitive images, the 2^{nd} row shows the container images by NNS, and the 3^{rd} and 4^{th} rows show the reconstructed images by FMGAN without and with CL, respectively.

REFERENCES

[1] Y. Wang, J. Liu, M. Luo, L. Yang, and L. Wang, "Privacy-preserving face recognition in the frequency domain," in *Proc. of AAAI*, 2022.

[2] M. Ribeiro, K. Grolinger, and M. A. Capretz, "Mlaas: Machine learning as a service," in *Proc. of ICMLA*, 2015.

[3] K. Zhao, Z. Zhou, X. Chen, R. Zhou, X. Zhang, S. Yu, and D. Wu, "Edgeadaptor: Online configuration adaption, model selection and resource provisioning for edge dnn inference serving at scale," *IEEE Transactions on Mobile Computing*, 2022.

[4] Avivah Litan, "AI in organizations: Managing AI risk leads to positive business outcomes", 2022. [Online]. Available: https://www.gartner.com/

[5] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," in *Proc. of CCS*, 2019.

[6] J. Domingo-Ferrer, K. Muralidhar, and M. Bras-Amorós, "General confidentiality and utility metrics for privacy-preserving data publishing based on the permutation model," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[7] P. Kairouz, B. Mcmahan, S. Song, O. Thakkar, A. Thakurta, and Z. Xu, "Practical and private (deep) learning without sampling or shuffling," in *Proc. of ICML*, 2021.

[8] Y. Mao, W. Hong, B. Zhu, Z. Zhu, Y. Zhang, and S. Zhong, "Secure deep neural network models publishing against membership inference attacks via training task parallelism," *IEEE Transactions on Parallel and Distributed Systems*, 2022.

[9] J. Wang, S. Guo, X. Xie, and H. Qi, "Protect privacy from gradient leakage attack in federated learning," in *Proc. of INFOCOM*, 2022.

[10] L. Lyu, Y. Li, K. Nandakumar, J. Yu, and X. Ma, "How to democratise and protect AI: Fair and differentially private decentralised deep learning," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[11] Z. Lu, H. J. Asghar, M. A. Kaafar, D. Webb, and P. Dickinson, "A differentially private framework for deep learning with convexified loss functions," *IEEE Transactions on Information Forensics and Security*, 2022.

[12] C. Guo, B. Karrer, K. Chaudhuri, and L. van der Maaten, "Bounding training data reconstruction in private (deep) learning," in *Proc. of ICML*, 2022.

[13] C. Ganhör, D. Penz, N. Rekabsaz, O. Lesota, and M. Schedl, "Unlearning protected user attributes in recommendations with adversarial training," in *Proc. of SIGIR*, 2022.

[14] I.-C. Hsieh and C.-T. Li, "Netfense: Adversarial defenses against privacy attacks on neural networks for graph data," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[15] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*,2016.

[16] P. Xie, B. Wu, and G. Sun, "Bayhenn: Combining bayesian deep learning and homomorphic encryption for secure dnn inference," in *Proc. of IJCAI*, 2019.

[17] S. Bian, T. Wang, M. Hiromoto, Y. Shi, and T. Sato, "Ensei: Efficient secure inference via frequency-domain homomorphic convolution for privacy-preserving visual recognition," in *Proc. of CVPR*, 2020.

[18] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *Proc. of USENIX Security*, 2020.

[19] Q. Lou and L. Jiang, "Hemet: A homomorphic-encryption-friendly privacy-preserving mobile neural network architecture," in *Proc. of ICML*, 2021.

[20] M. Li, S. S. M. Chow, S. Hu, Y. Yan, C. Shen, and Q. Wang, "Optimizing privacy-preserving outsourced convolutional neural network predictions," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[21] W. He, S. Li, W. Wang, M. Wei, and B. Qiu, "Cryptoeyes: Privacy preserving classification over encrypted images," in *Proc. of INFOCOM*, 2021.

[22] N. K. Jha, Z. Ghodsi, S. Garg, and B. Reagen, "Deepreduce: Relu reduction for fast private inference," in *Proc. of ICML*, 2021.

[23] G. Lloret-Talavera, M. Jorda, H. Servat, F. Boemer, C. Chauhan, S. Tomishima, N. N. Shah, and A. J. Pena, "Enabling homomorphically encrypted inference for large dnn models," *IEEE Transactions on Computers*, 2022.

[24] M. Sun and W. P. Tay, "On the relationship between inference and data privacy in decentralized iot networks," *IEEE Transactions on Information Forensics and Security*, 2020.

[25] L. Lyu, J. C. Bezdek, J. Jin, and Y. Yang, "Foreseen: Towards differentially private deep inference for intelligent internet of things," *IEEE Journal on Selected Areas in Communications*, 2020.

[26] F. Mireshghallah, M. Taram,P. Ramrakhyani, D. Tullsen, and H. Esmaeilzadeh, "Shredder: Learning noise distributions to protect inference privacy," in *Proc. of ASPLOS*, 2020.

[27] J. Yang, L. Xiang, J. Yu, X. Wang, B. Guo, Z. Li, and B. Li, "Matrix gaussian mechanisms for differentially-private learning," *IEEE Transactions on Mobile Computing*, 2021.

[28] L. Xiang, W. Li, J. Yang, X. Wang, and B. Li, "Differentially-private deep learning with directional noise," *IEEE Transactions on Mobile Computing*, 2021.

[29] P. Vepakomma, J. Balla, and R. Raskar, "Privatemail: Supervised manifold learning of deep features with privacy for image retrieval." in *Proc. of AAAI*, 2022.

[30] Q. Liu, J. Yang, H. Jiang, J. Wu, T. Peng, T. Wang, and G. Wang, "When deep learning meets steganography: Protecting inference privacy in the dark," in *Proc. of INFOCOM*, 2022.

[31] C. K. Chan and L. M. Cheng, "Hiding data in images by simple lsb substitution," *Pattern Recognition*, 2004.

[32] S. Baluja, "Hiding images within images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[33] X. Liao, J. Yin, M. Chen, and Z. Qin, "Adaptive payload distribution in multiple images steganography based on image texture features," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[34] C. Xiao, B. Li, J. yan Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," in *Proc. of IJCAI*, 2018.

[35] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint*, 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[36] Amazon SageMaker. https://aws.amazon.com/sagemaker/

[37] ClosedLoop. https://www.closedloop.ai/

[38] J. Donahue and K. Simonyan, "Large scale adversarial representation learning," in *Proc. of NeurIPS*, 2019.

[39] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. of CVPR*, 2006.

[40] Z. He, T. Zhang, and R. B. Lee, "Attacking and protecting data privacy in edge-cloud collaborative inference systems," *IEEE Internet of Things Journal*, 2021.

[41] Y. Lecun, L. D. Jackel, L. Bottou, C. Cortes, and V. Vapnik, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural Networks*, 1995.

[42] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Handbook of Systemic Autoimmune Diseases*, 2009.

[43] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, 2012.

[44] Y. Netzer, T. Wang, A. Coates, A. Bissacco, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. of NeurIPS*, 2011.

[45] J. H. Cheon, D. Kim, and K. Lee, "Mhz2k: MPC from HE over $\mathbb{Z}_{2^k}$ with New Packing, Simpler Reshare, and Better ZKP," in *Proc. of CRYPTO*, 2021.

[46] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proc. of ICANN*, 2011.

[47] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," in *Proc. of ICLR*, 2017.

[48] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *Proc. of ICLR*, 2019.

[49] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proc. of ICML*, 2004.

[50] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, "Not just privacy: Improving performance of private deep learning in mobile cloud," in *Proc. of KDD*, 2018.

[51] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proc. of AAAI*, 2020.

[52] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," *arXiv preprint*, 2017. [Online]. Available: https://arxiv.org/abs/1703.00810

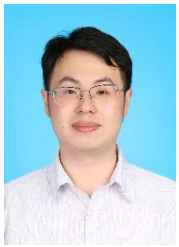
[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, 2012.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc of ICLR*, 2015.

[55] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Process*, 2004.

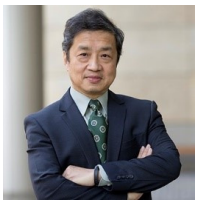
Shulan Wang received her B.Sc. in Network Engineering in 2021 from Anhui University, China. Currently, she is working toward the M.Sc. in the College of Computer Science and Electronic Engineering at Hunan University, China. Her research interests include privacy issues in artificial intelligence.

Qin Liu received her B.Sc. in Computer Science in 2004 from Hunan Normal University, China, received her M.Sc. in Computer Science in 2007, and received her Ph.D. in Computer Science in 2012 from Central South University, China. She has been a Visiting Student at Temple University, USA. Her research interests include security and privacy issues in cloud computing. Now, she is an Associate Professor in the College of Computer Science and Electronic Engineering at Hunan University, China.



Yang Xu received the Ph.D. degree in Computer Science and Technology from Central South University, China, in 2019. He is currently an Associate Professor at the College of Computer Science and Electronic Engineering, Hunan University, Changsha. His research interests include cloud computing, blockchain, federated learning, and privacy computing. He is a member of Blockchain Technical Committee of China Computer Federation (CCF) and China Society for Industrial and Applied Mathematics (CSIAM), and a member of IEEE and ACM.

Hongbo Jiang received the PhD degree from Case Western Reserve University, in 2008. After that, he joined the faculty of the Huazhong University of Science and Technology as a full professor. Now, he is a full professor with the College of Computer Science and Electronic Engineering, Hunan University. His research concerns computer networking, especially algorithms and protocols for wireless and mobile networks. He is serving as an editor for the IEEE/ACM Transactions on Networking, associate editor for the IEEE Transactions on Mobile Computing, and associate technical editor for the IEEE Communications Magazine.



Jie Wu is the Chair and a Laura H. Carnell Professor in the Department of Computer and Information Sciences at Temple University, Philadelphia, PA, USA. Prior to joining Temple University, he was a Program Director at the National Science Foundation and a Distinguished Professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, network trust and security, and routing protocols. Dr. Wu has regularly published in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Services Computing, and Journal of Parallel and Distributed Computing. Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

Tian Wang received his BSc and MSc degrees in Computer Science from the Central South University in 2004 and 2007, respectively. He received his PhD degree in City University of Hong Kong in 2011. Currently, he is a professor at the Institute of Artificial Intelligence and Future Networks, Beijing Normal University & UIC, China. His research interests include internet of things and edge computing.

Tao Peng received the B.Sc. in Computer Science from Xiangtan University, China, in 2004, the M.Sc. in Circuits and Systems from Hunan Normal University, China, in 2007, and the Ph.D. in Computer Science from Central South University, China, in 2017. Now, she is an Associate Professor of School of Computer Science and Cyber Engineering, Guangzhou University, China. Her research interests include network and information security issues.

Guojun Wang received his Ph.D. degree in Computer Science, at Central South University, China in 2002. He is a Pearl River Scholarship Distinguished Professor of Higher Education in Guangdong Province, and a Doctoral Supervisor of School of Computer Science and Cyber Engineering, Guangzhou University, China. He has been listed in Chinese Most Cited Researchers (Computer Science) by Elsevier in the past eight consecutive years (2014-2021). His research interests include artificial intelligence, big data, cloud computing, Internet of Things (IoT), and blockchain. He is a Distinguished Member of CCF, a Member of IEEE, ACM and IEICE.