When Deep Learning Meets Steganography: Protecting Inference Privacy in the Dark

Qin Liu[†], Jiamin Yang[†], Hongbo Jiang[†]*, Jie Wu[‡], Tao Peng[††], Tian Wang[§], and Guojun Wang[††]*
[†]College of Computer Science and Electronic Engineering, Hunan University, P. R. China
[‡]Department of Computer and Information Sciences, Temple University, Philadelphia, USA
[††]School of Computer Science and Cyber Engineering, Guangzhou University, P. R. China
[§]Institute of Artificial Intelligence and Future Networks, Beijing Normal University & UIC, P. R. China
*Correspondence to: hongbojiang2004@gmail.com; csgjwang@gzhu.edu.cn

***Abstract*—While cloud-based deep learning benefits for high-accuracy inference, it leads to potential privacy risks when exposing sensitive data to untrusted servers. In this paper, we work on exploring the feasibility of steganography in preserving inference privacy. Specifically, we devise GHOST and GHOST+, two private inference solutions employing steganography to make sensitive images invisible in the inference phase. Motivated by the fact that deep neural networks (DNNs) are inherently vulnerable to adversarial attacks, our main idea is turning this vulnerability into the weapon for data privacy, enabling the DNN to misclassify a stego image into the class of the sensitive image hidden in it. The main difference is that GHOST retrains the DNN into a poisoned network to learn the hidden features of sensitive images, but GHOST+ leverages a generative adversarial network (GAN) to produce adversarial perturbations without altering the DNN. For enhanced privacy and a better computation-communication trade-off, both solutions adopt the edge-cloud collaborative framework. Compared with the previous solutions, this is the first work that successfully integrates steganography and the nature of DNNs to achieve private inference while ensuring high accuracy. Extensive experiments validate that steganography has excellent ability in accuracy-aware privacy protection of deep learning.**

***Index Terms*—Deep learning, steganography, adversarial attacks, inference privacy, edge computing, cloud computing.**

I. Introduction

In recent years, deep learning has achieved remarkable success in a variety of domains, such as computer vision, audio processing, natural language processing, etc [1], [2]. Deep neural network (DNN) is empirical and data-driven, and the performance is highly dependent on its scale and complexity. As a result, an increasing number of users outsource DNN training by using machine learning as a service (MLaaS) [3] or even directly reuse pre-trained DNNs from the online cloud repositories (e.g., Caffe Model Zoo [4] and BigML [5]). Despite the benefits of cloud-based deep learning, it presents significant privacy issues when shifting the entire DNN and inference computation to the cloud [6]. Although researchers have made a great effort to protect users' sensitive data, most of them focus on privacy issues in the training phase [7]–[14].

In terms of the inference phase, researchers [15]–[22] have tried to leverage homomorphic encryption [32] and secure multi-party computing [33] to guarantee data privacy. However, these methods incur prohibitive computational and communication costs, making it difficult to implement them in complex and large DNNs. Another line of work [23]–[27] employs differential privacy [34], [35] to obfuscate inference instances with deliberate perturbation. There is a trade-off between the amount of injected noise and the accuracy of the inference results. That is, the more the noise added for higher privacy, the lower the inference accuracy. Nonetheless, the research in this field is still in its infancy. An open problem is *whether other technologies are available in preserving inference privacy while ensuring high scalability and accuracy*?

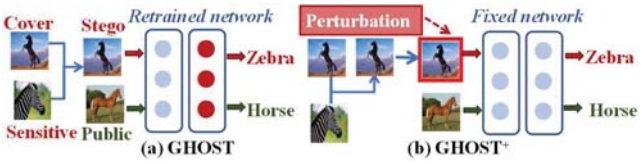


Fig. 1. The high-level idea of GHOST and GHOST$^+$.

In this paper, we work on exploring the feasibility of image steganography [37]–[39] in protecting data privacy in the inference phase. To this end, we first propose a private deep learninG solution with Hidden infOrmation based on STeganography, named GHOST, where sensitive images are hidden into public images before data transmission. To improve the robustness against latent images, GHOST retrains the DNN with both the public and stego images, so as to learn the hidden features of sensitive images. In view of the fact that most vendors disallow altering their proprietary DNNs, we further put forward GHOST$^+$, which leverages the generative adversarial network (GAN) [40] to perturb stego images in a subtly way, without making any modification to the DNN. Since sensitive images are invisible and hidden, inference privacy is preserved in the dark. Moreover, our solutions not only consume reasonable overheads, enabling high scalability in large-scale DNNs, but also outperform the state-of-the-art work in the best cases, allowing for high-accuracy inference.

Intuitively, both solutions take advantage of the vulnerability of DNNs against adversarial attacks, which mislead DNNs to produce adversary-selected results by either poisoning the DNN or crafting adversarial inputs [41]–[43]. By turning this vulnerability into the weapon for data privacy, our solutions enable the DNN to *misclassify* a stego image into the class of the sensitive image hidden in it with a high probability. To protect data privacy while achieving a *high attack success rate* (ASR), GHOST retrains the DNN into a poisoned network, but GHOST$^+$ generates adversarial perturbations by GAN.

TABLE I
PRIVATE DEEP LEARNING SOLUTIONS

	Inference privacy	Training privacy
Intrusive	GHOST, ARDEN [23] DPFE [24], CVDNN [27]	With SDP and federate learning [7]–[11],
Non-intrusive	GHOST$^+$, MiniONN [22] SHREDDER [26]	With LDP and artificial data [12]–[14]

The solutions based on cryptographic techniques [15]–[21] can be used to protect both the training and inference privacy intrusively.

The high-level idea of our solutions is illustrated in Fig. 1.

Finally, both solutions adopt the edge-cloud collaborative framework that partitions a DNN across edge devices and cloud servers. By outsourcing the large portions of a DNN, we can achieve a better balance between computation and communication, while ensuring enhanced privacy [44], [45]. As for application scenarios, GHOST involves the training process of DNNs and can be employed to protect data privacy in the MLaaS environment. In contrast, GHOST$^+$ keeps the DNN intact and is more appropriate to the environment where users online use pre-trained DNN from cloud repositories.

Our main contributions are summarized as follows:

- To the best of our knowledge, this is the first work that successfully utilizes image steganography and adversarial attacks to protect inference privacy in the dark. To explore the power of steganography in private deep learning, different steganography techniques are used to hide images.
- We propose an intrusive solution, GHOST, and a non-intrusive solution, GHOST$^+$, both of which not only hide sensitive images into public images before uploading to preserve privacy, but also mislead the DNN to output our expected results for high inference accuracy. Compared with GHOST, GHOST$^+$ is more practical and flexible since it incurs only a minor accuracy loss while keeping the DNN intact, and it can effectively avoid performance degradation by training a GAN for each sensitive type.
- We provide a formal formulation and show that both GHOST and GHOST$^+$ can protect data privacy in an invisible way. We also empirically validate that it is difficult for attackers to perceive the existence of sensitive images even if stego images can be reconstructed by launching feature inversion attacks [44] and denoising [46].
- We conducted evaluations on four datasets, MNIST [47], CIFAR-10 [48], GTSRB [49], and SVHN [50]. Experimental results show that our solutions outperform the state-of-the-art solutions when the number of sensitive types is within a given range. With these encouraging results, we confirm that, apart from cryptography and differential privacy, steganography is a promising tool for accuracy-aware privacy protection in deep learning.

II. RELATED WORK

With the wide application of DNNs, how to protect privacy in cloud-based deep learning has drawn a great deal of attention. Existing methods span different stages of deep learning from training to inference. The majority of these researches focused on the privacy issues in the training phase [7]–[14]. According to whether or not a DNN is modified, the private deep learning methods can be further categorized into intrusive and non-intrusive types. As shown in Table I, GHOST and GHOST$^+$ aim to protect inference privacy, by adopting the intrusive and non-intrusive solutions, respectively.

Cryptographic techniques. Orlandi et al. [15] proposed a oblivious neural network which encrypted input data with homomorphic encryption. Bost et al. [16] applied homomorphic encryption to achieve privacy-preserving classification. Dowlin et al. [17] designed CryptoNets to make encrypted predictions over encrypted data. Two private machine learning frameworks ABY^3 [18] and SecureNN [19] were proposed by utilizing secure 3-party computation. To improve efficiency, Falcon [20] combined ABY^3 and SecureNN, while Trident [21] designed a 4-party security protocol with an additional honest party. Liu et al. [22] designed MiniONN, an oblivious neural network that supported privacy-preserving predictions without altering the network. However, either homomorphic encryption or secure multi-party computation suffers from heavy computational and communication overheads, and this kind of solutions are too expensive to be implemented in complex DNNs.

Noise injection. Differential privacy, including local differential privacy (LDP) and standard differential privacy (SDP), has been wildly applied to guarantee the privacy of training data [7]–[13]. Recently, Wang et al. [23] proposed ARDEN, a private inference solution that employed differential privacy to perturb the local data. Osia et al. [24] designed a private feature extraction architecture, DPFE, which reduced the amount of leaked information by using principal component analysis. To avoid retraining the DNN, Leroux et al. [25] used an autoencoder to obfuscate the data before data transmission. However, the obfuscation was easily reversible. Mireshghallah et al. [26] proposed SHREDDER that learned the noise distribution non-intrusively. Xiang et al. [27] proposed a complex-valued network, CVDNN, which concealed the input data into a randomized phase. The key of noise injection is how to balance between accuracy and privacy.

Besides the above techniques, a branch of work [28]–[31] proposed running deep learning algorithms in trusted execution environments (TEE) [51] to protect data privacy and integrity. Our work aims to explore the feasibility of steganography in protecting inference privacy. Although deep learning has been applied to hide/discover images in steganography, this is the first work that applies steganography in private deep learning.

III. PRELIMINARY

A. Deep Learning

Deep learning, as a branch of machine learning, makes use of DNNs to find solutions for a variety of complex tasks. A DNN $f_{\boldsymbol{\theta}} = F_1 \circ F_1 \circ \ldots \circ F_L$ consists of a series of layers, where $\boldsymbol{\theta}$ are the parameters, $\circ$ denotes connection, and each layer F_i is a transformation function that converts the previous layer's output into the current layer's input. Given an initial input x, the final output of the DNN can be expressed as $f_{\boldsymbol{\theta}}(x) = F_L(F_{L-1}(\ldots(F_2(F_1(x)))))$. In classification tasks, the DNN takes an m-dimensional vector $x \in \mathbb{R}^m$ as input and outputs $y \in \mathbb{R}^M$, a probability distribution over the M classes.

Fig. 2. Samples from the LSB substitution system. The 1^{st} column shows the cover images, the 2^{nd} column shows the hidden images, the 3^{rd} column shows the stego images with $\zeta = 2$, the 4^{th} column shows the stego images with $\zeta = 3$, the 5^{th} column shows the differences between cover images and stego images in the third column, and the 6^{th} column show the differences between cover images and stego images in the fourth column.

The DNN parameters are learned from a training dataset $\mathbf{D}_{train} = \{(x_i, z_i)\}_i^N$ that contains a set of inputs $x_i \in \mathbb{R}^m$ and the corresponding ground-truth labels $z_i \in [1, M]$. The training process aims to minimize the average difference between the predictions and the ground-truth labels, which can be quantified by a loss function defined as follows:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}^*} \sum_i^N \mathcal{L}(f_{\boldsymbol{\theta}^*}(x_i), z_i). \tag{1}$$

For clarity, $\mathcal{L}(f_{\boldsymbol{\theta}^*}(x_i), z_i)$ is simplified as $\mathcal{L}(f; x_i)$ somewhere in this paper. For complex DNNs, the loss function is usually non-convex. In practice, we minimize the loss function by using the mini-batch stochastic gradient descent (SGD) algorithm [52]. Given a batch B of random samples, each parameter $\theta_j \in \boldsymbol{\theta}$ can be updated through back-propagation:

$$\theta_j = \theta_j - \alpha \frac{1}{|B|} \sum_{x_i \in B} \nabla_{\theta_j} \mathcal{L}(f; x_i), \tag{2}$$

where α is the learning rate. The performance of the trained DNN model is measured using its accuracy on a validation dataset, $\mathbf{D}_{valid}$, which is composed of a set of inputs and their ground-truth labels, s.t. $\mathbf{D}_{valid} \cap \mathbf{D}_{train} = \emptyset$.

B. Image Steganography

Image steganography is a type of covert communication technique that makes use of the content redundancy in digital media to conceal secret images [37]. As shown in Fig. 2, after hiding information, it is hard for observers to detect the existence of hidden images from stego images. A great number of hidden methods have been proposed and widely used for secret data transmission, copyright protection, access control, an so on. In this work, we mainly investigate the traditional least significant bit (LSB) substitution [38] and the state-of-the-art neural network-based steganography (NNS) [39].

(1) **LSB.** The basic principle of LSB is to hide the secret information into the least significant bits of the cover image. The changes in the cover image are unobservable by naked eyes, since only the lowest bits of pixels are replaced. For color images, each pixel consists of R, G, and B channels, and each channel is encoded into 8 bits. The rightmost bit of the pixel is the least important, which can be replaced without humans noticing the changes. However, the embedding capacity of LSB is limited. As the number of changed bits exceeds 4, the difference between stegos and covers will be dramatic. To solve this problem, we just hide the most significant bits of the hidden image instead of a full-size image. Let ζ denote the number of the least significant bits available. The effect after LSB substitution is shown in Fig. 2.

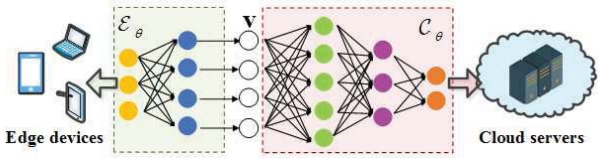


Fig. 3. The model of edge-cloud collaborative framework.

(2) **NNS.** The entire system consists of three networks: a prep-network, a hiding-network, and a reveal-network, which are trained at the same time. The prep-network takes a hidden image as input and transforms it into features that can be used by the hiding-network. The hiding-network takes a cover image and the features of hidden image as input, and creates a stego image that looks as similar as the cover image while containing enough information of the hidden image. The reveal-network takes the stego image as input and aims to recover the hidden image as realistically as possible.

Let C and C' represent the cover image and the stego image, and let H and H' represent the hidden image and the restored image, respectively. The whole system is trained through the following loss function: $\mathcal{L}_{NNS} = \mathbb{E}[|C - C'|] + \beta \mathbb{E}[|H - H'|]$, where β is the weight balancing the importance between the invisibility and restorability of hidden images. Here, $|C - C'|$ (resp. $|H - H'|$) measures the pixel difference between C and C' (resp. H and H'). The objective of the hiding-network is to minimize the average difference between cover images and stego images, and the reveal-network aims to minimize the average difference between hidden images and restored images. With such a loss function, the hiding-network learns where/how to hide images at the end of the training process. The reason why we investigate these two techniques is that LSB is very efficient due to its simplicity, and NNS has excellent capacity in hiding a large amount of information.

IV. Models

A. System Model

As shown in Fig. 3, with a cutting point $\bullet$, the DNN is partitioned into two parts: $f_{\boldsymbol{\theta}} = \mathcal{E}_{\boldsymbol{\theta}} \bullet \mathcal{C}_{\boldsymbol{\theta}}$, where $\mathcal{E}_{\boldsymbol{\theta}}$ is deployed on edge devices and $\mathcal{C}_{\boldsymbol{\theta}}$ is hosted by cloud servers. Given an inference instance, the edge-side network $\mathcal{E}_{\boldsymbol{\theta}}$ extracts features embedded and sends the intermediate value v to the cloud-side network $\mathcal{C}_{\boldsymbol{\theta}}$, which calculates the final output and gets it back.

DNN partitioning is usually formulated as an optimization problem. Inspired by the previous work [23], [26], we let edge devices store a small number of convolutional layers for feature extraction, while offloading the most layers (including all fully-connected layers) to the cloud. This edge-cloud collaborative framework achieves a better balance between computation and communication. On the one hand, it consumes less time and energy for edge devices to process a shallow-layer DNN compared with the whole network. On the other hand, the pooling/ReLU layers reduce the data elements, and the size of intermediate values to be transmitted is smaller than original inputs. It is worth noticing that, this partitioning is also

conducive to privacy protection. As demonstrated in [44], the deeper the partitioning point, the higher the privacy level.

B. Threat Model

The threat model assumes that the edge devices are fully trusted, and the cloud is a potential attacker with the knowledge of the entire DNN. For data privacy, the edge devices locally process the original data and extract general features before interacting with the cloud. The cloud is assumed to be interested in inferring useful information from the received intermediate value other than the inference result. The cloud would intercept the transferred features and perform *feature inversion attacks in a white-box setting* [44] and *denoising* [46] to recover the original input from the intermediate value.

To protect data privacy in the inference phase, this work aims to achieve the following privacy guarantee:

Invisibility. When the cloud tries to reconstruct the original input from the intermediate value, it can only synthesize inputs revealing no information about the sensitive images. In other words, the sensitive image is invisible and hidden in the synthesized input. We adopt two perceptual metrics, Peak Signal to Noise Ratio (PSNR) and Structural Similarity (SSIM) [53], to quantify the invisibility of hidden images.

(1) PSNR is a common tool used to measure the difference of the pixel points between two images. Given two images X and Y, the mean square error (MSE) between them is calculated as: $MSE = \frac{1}{h \times w} \sum_{i=1}^{h} \sum_{j=1}^{w} (X(i,j) - Y(i,j))^2$, where h and w is the height and width, respectively. Let MAX^2 denote the maximum pixel value. The PSNR between images X and Y is computed by: $PSNR = 10 \times \log_{10}\left(\frac{MAX^2}{MSE}\right)$, which represents the ratio between the maximum power of an image and the power of noise. The larger the value of PSNR, the more similar the two images will be.

(2) SSIM is an index to measure the similarity between images. Given two images, X and Y, the luminance, contrast and structure similarities between two images are measured by $L_{X,Y} = \left[\frac{2\mu_X\mu_Y + C_1}{\mu_X^2 + \mu_Y^2 + C_1}\right]$, $C_{X,Y} = \left[\frac{2\sigma_X\sigma_Y + C_2}{\sigma_X^2 + \sigma_Y^2 + C_2}\right]$, and $S_{X,Y} = \left[\frac{\sigma_{XY} + C_3}{\sigma_X\sigma_Y + C_3}\right]$, respectively. Here, μ_i and σ_i are weighted mean and variance of image i for $i \in \{X, Y\}$, and σ_{XY} is covariance of images X and Y. In addition, C_1, C_2 and C_3 are constants included to avoid instability when denominators are close to zero. The SSIM between two images is calculated by $SSIM(X,Y) = L_{X,Y}^a \times C_{X,Y}^b \times S_{X,Y}^c$, where a, b, c are weights used to adjust the relative importance of three components. As the value of SSIM gets closer to 1, the more similar the two images will be.

V. The Intrusive Solution GHOST

The overview of GHOST is presented in Fig. 4, where the DNN $f_{\boldsymbol{\theta}}$ is divided into the edge-side network $\mathcal{E}_{\boldsymbol{\theta}}$ and the cloud-side network $\mathcal{C}_{\boldsymbol{\theta}}$. Inspired by the work of [23], the edge-side network $\mathcal{E}_{\boldsymbol{\theta}}$ is derived from the shallow layers of the original DNN $f_{\boldsymbol{\theta}}$, and its structure and weights are frozen. The cloud-side network $\mathcal{C}_{\boldsymbol{\theta}}$ is retrained in the training phase. In GHOST, the cloud is in charge of training and inference computation, and edge devices are responsible for hiding images and extracting features. For all data transmitted, it contains only abstract representations of stego images, protecting sensitive data in the dark. The key components of GHOST include *covert retraining* and *covert inference*.

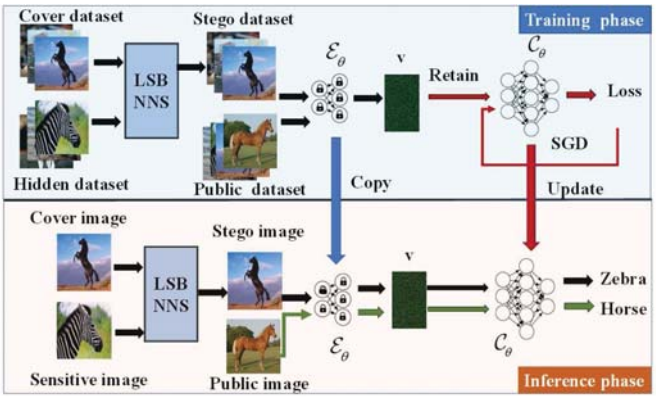


Fig. 4. The framework overview of GHOST.

A. Covert Retraining

To preserve data privacy, sensitive images are first hidden into public images before feature extraction. As shown in Section III-B, we can either hide the most significant bits of the sensitive image by using LSB or adopt NNS to hide the full-size image. However, this incurs the accuracy sacrifice in the inference phase. To improve the robustness of the DNN, we retrain the cloud-side DNN to learn the hidden features of sensitive information from the representations of stego images. The retraining process is summarized in Algorithm 1.

In Algorithm 1, the cloud-side network $\mathcal{C}_{\boldsymbol{\theta}}$ is retrained on the representations of a public dataset $\mathbf{D}_p$ and a generative dataset $\mathbf{D}_g$, where $\mathbf{D}_p$ contains raw training data and $\mathbf{D}_g$ is composed of stego data. Given a cover dataset $\mathbf{D}_c$ that contains public data, and a hidden dataset $\mathbf{D}_h$ that contains images of the same type as the sensitive images, the generative dataset $\mathbf{D}_g$ is constructed as follows: For each sample $(H_i, L_{H_i}) \in \mathbf{D}_h$, a stego image $\widetilde{C}_j$ is generated by hiding H_i into the cover image $C_j \in \mathbf{D}_c$, and then a sample (C_j, L_{H_i}) is put into $\mathbf{D}_g$. In this way, each sample in $\mathbf{D}_g$ looks similar as the cover image in $\mathbf{D}_c$, but has the same label as the hidden image in $\mathbf{D}_h$. The training loss is defined as follows:

$$\mathcal{L}(\mathcal{C}; \mathbf{x}^r, \tilde{\mathbf{c}}^r) = \mathcal{L}(\mathcal{C}; \mathbf{x}^r) + \lambda \mathcal{L}(\mathcal{C}; \tilde{\mathbf{c}}^r), \tag{3}$$

where $\mathbf{x}^r, \tilde{\mathbf{c}}^r$ are representations of samples from $\mathbf{D}_p$ and $\mathbf{D}_g$, respectively, and λ controls the importance between the losses of raw training data and generative training data.

B. Covert Inference

Once the training process is finished, the robustness of the cloud-side DNN against the stego representations is improved. To obtain the labels of insensitive images, the edge device directly extracts features from original images, without information hiding. As for a sensitive image, the edge device first picks a public image as the cover image, then hides the sensitive image by using steganography techniques, and finally extracts and uploads the features of the stego image.

Algorithm 1 Retraining the cloud-side network

Input: A public dataset $\mathbf{D}_p$, a cover dataset $\mathbf{D}_c$ a hidden dataset $\mathbf{D}_h$, an edge-side network $\mathcal{E}_{\boldsymbol{\theta}}$, and a cloud-side network $\mathcal{C}_{\boldsymbol{\theta}}$

OutPut: A retrained cloud-side network $\mathcal{C}_{\boldsymbol{\theta}'}$

```
Set D_g to an empty set
for each sample (H_i, l_{H_i}) in D_h do
    for each sample (C_j, l_{C_j}) in D_c do
        Obtain the stego image C̃_j by steganography
        Add (C̃_j, l_{H_i}) into D_g
Use E_θ to extract features from samples in D_g and D_p
Obtain C_θ' by training C_θ with the loss function of Eq. (3)
```

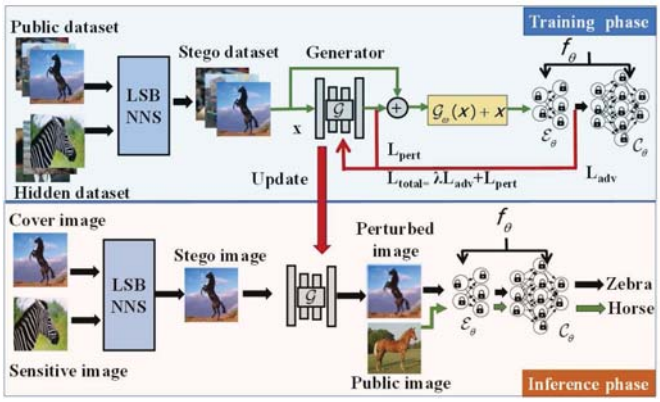


Fig. 5. The framework overview of GHOST$^+$.

On receiving the edge device's request, the cloud performs inference computation as normal and returns the final classification labels. Given the abstraction representations of stegos, the cloud may launch feature inversion attacks to recover the original input. However, as shown in Table III, even if the cloud is able to completely restore the stego image looking like the public image, the sensitive image is hidden and invisible.

Comparison with the state-of-the-art solution. Both GHOST and ARDEN [23] intrusively modify the cloud-side network $\mathcal{C}_{\boldsymbol{\theta}}$. The basic idea of GHOST is training a poisoned network $\mathcal{C}_{\boldsymbol{\theta}'}$, which outputs correct labels for clean instances, but implements the misclassification whenever a sensitive image is hidden in the input. On the contrary, ARDEN trains $\mathcal{C}_{\boldsymbol{\theta}}$ with noise that conforms to Laplace distribution, to enhance the network's robustness. The main difference is that the privacy level achieved by ARDEN is determined by the amount of noise injected, but the privacy of GHOST is protected by the steganography technique adopted. As shown in [23], when the amount of noise is minor, sensitive images will be successfully restored using the convolutional denoising autoencoder (CDA) [46]. For GHOST, even though only LSB and NNS are considered, steganography has been well studied and an abundant powerful methods are available. Moreover, ARDEN adds noise after feature extraction, and thus can be combined with GHOST to further enhance privacy.

VI. The Non-Intrusive Solution GHOST$^+$

Once the cloud-side network is retrained, GHOST allows users to conceal sensitive images while obtaining desired results. However, the pre-trained DNN is usually proprietary and not allowed to be changed. To improve the practicability, we propose the advanced solution, GHOST$^+$, which can effectively protect data privacy while keeping the DNN intact.

The overall architecture of GHOST$^+$ is illustrated in Fig. 5, where the DNN is partitioned across the edge device and the cloud as GHOST. The main difference is that GHOST$^+$ locally trains a GAN to produce adversarial perturbations instead of retraining the cloud-side network. Our basic idea is to leverage the super learning ability of GAN to add deliberate perturbations into stego images, making the DNN "misclassify" the perturbed stego images into our expected classes. It is motivated by the fact that most DNNs consistently misclassify adversarial samples that are formed by applying small perturbations to original inputs. GHOST$^+$ is a non-intrusive solution that is mainly consisted of the *adversarial perturbation training* and *adversarial inference* steps.

A. Adversarial Perturbation Training

A GAN consists of a generator and a discriminator, which learns unknown data distributions through a minmax two-player game. The objective of the generator is to fool the discriminator with synthesized images, while the discriminator aims to distinguish real images from the synthesized one. The generator implicitly learns the distribution when the process of the confrontation reaches a dynamic balance.

In GHOST$^+$, we design a GAN that can produce perturbations for stego images such that the pre-trained DNN outputs the label of the hidden image for the perturbed stego image. As shown in Fig. 5, the GAN consists of a generator $\mathcal{G}_{\boldsymbol{\omega}}$ and a pre-trained network $f_{\boldsymbol{\theta}}$. The generator $\mathcal{G}_{\boldsymbol{\omega}}$ is based on similar architecture of image-to-image translation [41], [54], which takes a stego image $\widetilde{C}$ as input and generates a perturbed image $\mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C})+\widetilde{C}$. The pre-trained network $f_{\boldsymbol{\theta}}$ is considered as the discriminator, but will not be updated in the training process. Given a stego image $\widetilde{C}$ that conceals a hidden image H, the goal of the generator $\mathcal{G}_{\boldsymbol{\omega}}$ is to produce the adversarial perturbation $\mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C})$, such that (1) $f_{\boldsymbol{\theta}}(\mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C})+\widetilde{C})=l_H$, where l_H is the label of the hidden image H; (2) $\mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C})$ should be small enough. To achieve this goal, we first design a loss function $\mathcal{L}_{adv}$ for misleading the pre-trained model $f_{\boldsymbol{\theta}}$:

$$\mathcal{L}_{adv}=\mathbb{E}[\mathcal{L}(f_{\boldsymbol{\theta}}(\mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C})+\widetilde{C}),l_H)]. \tag{4}$$

Here, $\mathcal{L}_{adv}$ represents the average distance between the prediction and the expected class. As the work in [41], we also use the $L2$ norm to bound the magnitude of the perturbation:

$$\mathcal{L}_{pert}=\mathbb{E}[||\mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C})||_2]. \tag{5}$$

Here, $\mathcal{L}_{pert}$ encourages the perturbed data to appear similar to the original input. Finally, the total loss is defined as:

$$\mathcal{L}_{total}=\lambda\mathcal{L}_{adv}+\mathcal{L}_{pert}, \tag{6}$$

where λ controls the relative importance of the accuracy of adversary inferences and the magnitude of perturbation. As the work of [55], we dynamically adjust parameter λ according to

Algorithm 2 Training the GAN in each batch

Input: A generative dataset $\mathbf{D}_g$, a pre-trained DNN $f_{\boldsymbol{\theta}}$, and a generator $\mathcal{G}_{\boldsymbol{\omega}}$

OutPut: A trained generator $\mathcal{G}_{\boldsymbol{\omega}'}$

{Parameters: Batch size $|B|$, weight λ, learning rate α}

$\mathbf{d} \leftarrow \mathbf{0}$
for each $(\widetilde{C}_i, l_{H_i})$ in a batch B **do**
  generate a perturbed image by $\widetilde{C}_i' \leftarrow \mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C}_i) + \widetilde{C}_i$
  $\mathcal{L}_{adv} \leftarrow \mathcal{L}(f_{\boldsymbol{\theta}}(\widetilde{C}_i'), l_{H_i})$
  $\mathcal{L}_{pert} \leftarrow ||\mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C}_i)||_2$
  Dynamic adjust weight λ with Eq. (7)
  $\mathcal{L}_{total} \leftarrow \lambda\mathcal{L}_{adv} + \mathcal{L}_{pert}$
  $\mathbf{d} \leftarrow \mathbf{d} + \nabla_{\boldsymbol{\omega}}\mathcal{L}_{total}$
$\boldsymbol{\omega}' \leftarrow \boldsymbol{\omega} - \alpha\frac{\mathbf{d}}{|B|}$

$\mathcal{L}_{adv}$ and $\mathcal{L}_{pert}$ to strike a better balance between two losses. In our experiments, λ is initialized to 2 and updated by:

$$\lambda = \begin{cases} 1/2, & if \quad \mathcal{L}_{pert} > \mathcal{L}_{adv} \\ 2, & if \quad \mathcal{L}_{pert} \leq \mathcal{L}_{adv} \end{cases} \tag{7}$$

Algorithm 2 outlines the process of adversarial perturbation training. In a similar way as GHOST, the edge device first constructs a generative dataset $\mathbf{D}_g$ which is fed to the generative network for training. Algorithm 2 is designed based on the SGD algorithm. In each batch, this algorithm generates perturbed images and feeds them into the pre-trained network $f_{\boldsymbol{\theta}}$ to obtain the loss on perturbed data $\mathcal{L}_{adv}$. This training process aims to minimize $\mathcal{L}_{adv}$ and the amount of perturbation $\mathcal{L}_{pert}$, and finally calculates the partial derivative of the joint loss with respect to each parameter in $\boldsymbol{\omega}$. The parameters are updated by the mean value of derivatives in the batch.

B. Adversarial Inference

Once the GAN is trained, it can generate adversarial perturbations efficiently for any inference instance. As shown in Fig. 5, for insensitive data, the edge device directly extracts features from original images and sends them to the cloud for further inference. To obtain the label of a sensitive image, the edge device first generates a stego image, then perturbs the stego image with the well-trained GAN, and finally extracts the general features of the perturbed stego image.

The extracted features are fed to the cloud-side network, and the final labels are achieved and sent back. In GHOST$^+$, the whole DNN is kept intact, and thus the cloud is completely oblivious to the addition of perturbations. Given the features, the cloud may launch feature inversion attacks to recover the original input, and then remove noise by using the CDA. However, as shown in Fig. 9, the restored images look similar to the cover images, hiding the sensitive images in the dark.

Comparison with the state-of-the-art solution. Both GHOST$^+$ and SHREDDER [26] add deliberate noise into inference instances without altering the pre-trained DNN $f_{\boldsymbol{\theta}}$. The basic idea of GHOST$^+$ is training a GAN to generate adversarial perturbations, making $f_{\boldsymbol{\theta}}$ output the label of the sensitive image hidden in the perturbed image. In contrast, SHREDDER trains a network generating optimized noise, making $f_{\boldsymbol{\theta}}$ output the label of the perturbed data. Although inference instances are obfuscated in both solutions, the reasons for adding noise are different. As GHOST, the privacy of GHOST$^+$ is guaranteed by steganography, and our goal is to generate the minimal perturbation to render $f_{\boldsymbol{\theta}}$ misbehaving. The privacy of SHREDDER is measured by the amount of noise injected, and the goal is to find out the maximum perturbation for privacy while making $f_{\boldsymbol{\theta}}$ behave normally. As ARDEN, sensitive images in SHREDDER can be revealed when the amount of injected noise is minor.

VII. Privacy Measurement

Mutual information (MI) has been widely used to quantify information leakage in steganography systems [37] as well as to understand the behavior of DNNs [56]. Given two random variables, X and Y, with a joint distribution $p(x, y)$, the MI between X and Y is defined as follows:

$$\begin{aligned} I[X;Y] &= KL[p(x,y)||p(x)p(y)] \\ &= -\int_X\int_Y p(x,y)\log_2(\frac{p(x)p(y)}{p(x,y)})\mathrm{d}x\mathrm{d}y \\ &= \mathbb{H}[X] - \mathbb{H}[X|Y], \end{aligned} \tag{8}$$

where $KL[p||q]$ is the Kullback-Liebler divergence of two distributions p and q, and $\mathbb{H}[X]$ and $\mathbb{H}[X|Y]$ denote the entropy and conditional entropy of X and Y, respectively.

A. The Privacy of GHOST

In terms of the achieved privacy level, we follow the work of [26] employing MI to quantify the information leakage between the original input and the intermediate value sent to the cloud. Consider a K-partition DNN consisting of L layers $f_{\boldsymbol{\theta}} = F_1 \circ F_2 \circ \ldots \circ F_L$, where the first K layers $\mathcal{E}_{\boldsymbol{\theta}} = F_1 \circ \ldots \circ F_K$ are deployed on edges, and the remainder layers $\mathcal{C}_{\boldsymbol{\theta}} = F_{K+1} \circ \ldots \circ F_L$ are on the cloud. Let $\mathcal{ST}$ be the steganography method (LSB or NNS) and let X denote the original input. The privacy of GHOST is defined as:

$$\mathcal{P}_K = -I[X; \mathcal{E}_{\boldsymbol{\theta}}(\widetilde{C})], \tag{9}$$

where $\widetilde{C} = \mathcal{ST}(X)$ is the processed result after steganography. From Eq. (9), we know that the lower MI implies the higher level of privacy. Since $\mathcal{E}_{\boldsymbol{\theta}}$ is deterministic and frozen, Eq. (9) can be transformed into:

$$\begin{aligned} \mathcal{P}_K &= -\mathbb{H}[\mathcal{E}_{\boldsymbol{\theta}}(\widetilde{C})] + \mathbb{H}[\mathcal{E}_{\boldsymbol{\theta}}(\widetilde{C})|X] = -\mathbb{H}[\mathcal{E}_{\boldsymbol{\theta}}(\widetilde{C})] \\ &= -\mathbb{H}[F_K(F_{K-1}(\ldots(F_1(\mathcal{ST}(X))))))]. \end{aligned} \tag{10}$$

In the above equation, the term $\mathbb{H}[\mathcal{E}_{\boldsymbol{\theta}}(\widetilde{C})]$ controls the amount of information leaked to the cloud, and we want to minimize this information. This information can be minimized by the combined efforts of steganography and feature extraction. On the one hand, the steganography method generates stego images looking similar to the cover image while hiding the sensitive image stealthily. On the other hand, the neural network operations like pooling, ReLU and convolutions modify the input information, and thus the more the number of layers deployed locally, the higher privacy yield for a given accuracy.

TABLE II
THE MODEL STRUCTURE AND PARAMETER SETTINGS

Dataset	# of images	# of classes	Input size	Model architecture	Accuracy	$[\alpha, \lvert B\rvert, \lvert E\rvert]$
MNIST	70,000	10	$28 \times 28 \times 1$	2Conv+2Pooling+2Dense	98.25	$[0.001, 256, 20]$
CIFAR-10	60,000	10	$32 \times 32 \times 3$	4Conv+2Pooling+4BN+4Dropout+3Dense	87.13	$[0.001, 128, 100]$
GTSRB	51,839	43	$32 \times 32 \times 3$	6Conv + 3Pooling +4Dropout+2Dense	96.21	$[0.001, 128, 50]$
SVHN	99,289	10	$32 \times 32 \times 3$	AlexNet [57]	91.79	$[5e^{-4}, 128, 50]$

Let $\mathcal{P}_{st} = -I[X, \widetilde{C}]$ denote the privacy level achieved by steganogpraphy. According to data processing inequality (DPI) theorem [56], a lower bound on privacy can be derived by:

$$\mathcal{P}_K \geq \mathcal{P}_{K-1} \geq \ldots \geq \mathcal{P}_1 \geq \mathcal{P}_{st}, \tag{11}$$

where $\mathcal{P}_i$ denotes the privacy provided by the i-th layer for $1 \leq i \leq K$. In other words, the edge-cloud collaborative framework achieves better privacy protection compared with steganography alone. At the same time, the deeper the partitioning point, the higher the privacy level. The task of boosting $\mathcal{P}_{st}$ can be achieved by uniform embedding distribution [37] and is out of the scope of this paper. To improve the privacy level of GHOST, our approach is to increase the number of layers K deployed on edge devices. As shown in Fig. 10, the attacker's reconstructive ability decreases as K increases.

B. The Privacy of GHOST$^+$

Consider a K-partition DNN, $f_{\boldsymbol{\theta}} = \mathcal{E}_{\boldsymbol{\theta}} \bullet \mathcal{C}_{\boldsymbol{\theta}}$. Let X denote the original input, let $\widetilde{C} = \mathcal{ST}(X)$ denote the processed result after steganography, and let $\mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C})$ denote the generated perturbation. The privacy of GHOST$^+$ is defined as:

$$\mathcal{P}_K^+ = -I[X; \mathcal{E}_{\boldsymbol{\theta}}(\widetilde{C} + \mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C}))]. \tag{12}$$

As $\mathcal{E}_{\boldsymbol{\theta}}$ is a deterministic function, we have:

$$\begin{aligned} \mathcal{P}_K^+ &= -\mathbb{H}[\mathcal{E}_{\boldsymbol{\theta}}(\widetilde{C} + \mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C}))] + \mathbb{H}[\mathcal{E}_{\boldsymbol{\theta}}(\widetilde{C} + \mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C}))|X] \\ &= -\mathbb{H}[\mathcal{E}_{\boldsymbol{\theta}}(\widetilde{C} + \mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C}))] \\ &= -\mathbb{H}[F_K(F_{K-1}(\ldots(F_1(\mathcal{G}(\mathcal{ST}(X)))))]. \end{aligned} \tag{13}$$

where $\mathcal{G}(Y) = Y + \mathcal{G}_{\boldsymbol{\omega}}(Y)$ denotes a function adding noise on input Y. In the above equation, the term $\mathbb{H}[\mathcal{E}_{\boldsymbol{\theta}}(\widetilde{C} + \mathcal{G}_{\boldsymbol{\omega}}(\widetilde{C}))]$ controls the amount of exposed information, and is supposed to be minimized. The loss function $\mathcal{L}_{pert}$ defined in Eq. (5) encourages injecting the minimal noise such that the perturbed stego image slightly differs from the cover image (concealing original input in the dark). Let $\mathcal{P}_{st} = -I[X, \widetilde{C}]$ and $\mathcal{P}_{pert} = -I[X, \mathcal{G}(\widetilde{C})]$ denote the privacy level achieved by steganogpraphy and perturbation, respectively. Again using the DPI theorem, a lower bound on privacy is derived by:

$$\mathcal{P}_K^+ \geq \mathcal{P}_{K-1}^+ \geq \ldots \geq \mathcal{P}_1^+ \geq \mathcal{P}_{pert} \geq \mathcal{P}_{st}. \tag{14}$$

This equation implies that by adding adversarial perturbations, GHOST$^+$ can reach the privacy level at least as much as steganography. To improve the privacy level, we adopt the same solution as GHOST by increasing the value of K.

VIII. EVALUATION

In this section, we implement the proposed solutions, GHOST and GHOST$^+$, using TensorFlow and evaluate their effectiveness in terms of *performance* and *privacy*. The metric of performance is tested by the inference accuracy for classification tasks, and the level of privacy is measured by the robustness against feature inversion attacks. Besides, the scalability of our solutions is tested by edge-side execution time. To validate our performance in practice, we compare GHOST with the intrusive solution, ARDEN [23], and compare GHOST$^+$ with the non-intrusive solution SHREDDER [26]. All four solutions adopt the edge-cloud collaborative framework for inference privacy. Unlike our solutions that employ steganography, both ARDEN and SHREDDER obfuscate inference instances with deliberate perturbations to protect data privacy.

A. Experimental Setup

We demonstrate the effectiveness of our solutions on four widely used image classification datasets: MNIST, CIFAR-10, GTSRB, and SVHN, where each image is first scaled to $[0, 1]$ range in a pre-processing stage. The experimental information for each dataset is shown in Table II. To train the DNN, the hyperparameters, including the learning rate α, the batch size $|B|$, and the number of epoches $|E|$ are configured according to different benchmarks. To train the generator network in GHOST$^+$, we use the Adam solver [58] with $\alpha = 0.0001$.

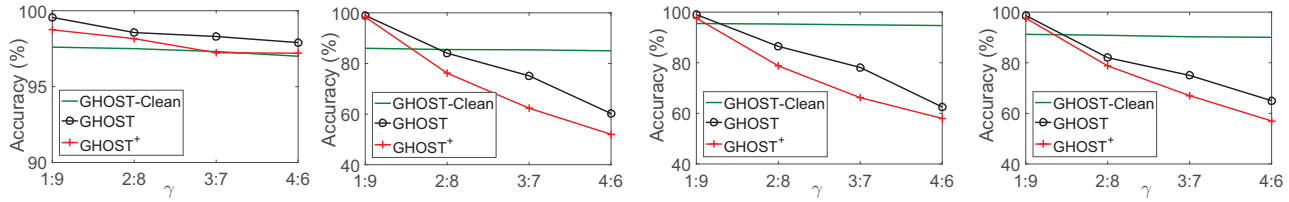
In terms of the steganography methods, LSB sets the number of least significant bits to $\zeta = 3$, and NNS is employed to hide full-size images. While training the NNS system, we set $\alpha = 0.001$ and $|B| = 32$ for all datasets, and set $|E| = 20$ for MNIST, and $|E| = 200$ for the remaining datasets.

Edge device specifications. The operations relevant to hiding images, adversarial perturbation training/injection, and feature extraction are performed on a laptop, which is equipped with a NVIDIA GeForce MX250 off-the-shelf GPU running CUDA V10.2.141 on the MS Windows 10 operating system.

Cloud specifications. The training and inference computations of DNN are run on a server with an Intel(R) Xeon(R) Gold 5218 CPU of 2.3GHz and 128GB memory, alongside a NVIDIA GeForece RTX 3080 GPU running CUDA V11.4.56 on the MS Windows Server 2016 operating system.

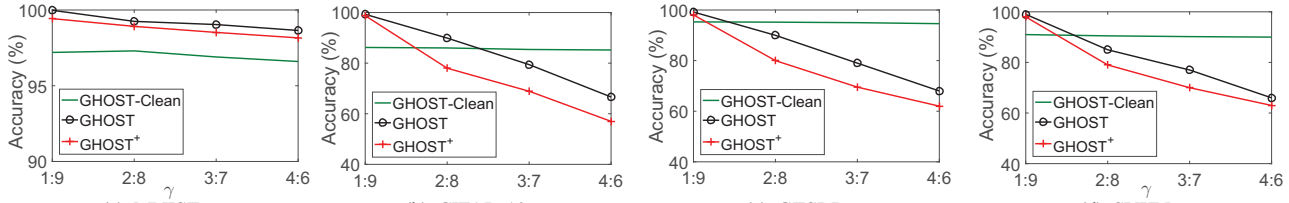
B. Performance

For each class, the related images are classified into two types: sensitive samples and public samples. For each sensitive image, both GHOST and GHOST$^+$ need to choose a public image as the cover image and hide it with the steganography methods (LSB or NNS). According to whether or not there are sensitive samples associated, the labels can be divided into sensitive labels and public labels. The number of sensitive labels n_s and the number of public labels n_c are two important parameters affecting the performance of our solutions. Let γ denote the ratio of n_s to n_c. To test the influence of these parameters, we fix the number of layers deployed on edge devices with $K = 3$, and evaluate the inference accuracy of our solutions under the setting of $\gamma = \{1:9, 2:8, 3:7, 4:6\}$



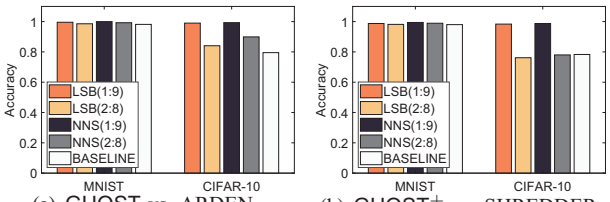
(a) MNIST. (b) CIFAR-10. (c) GTSRB. (d) SVHN.

Fig. 6. The results of GHOST and GHOST$^+$ for LSB. GHOST-Clean and GHOST denote the results for public and sensitive samples, respectively.



(a) MNIST. (b) CIFAR-10. (c) GTSRB. (d) SVHN.

Fig. 7. The results of GHOST and GHOST$^+$ for NNS. GHOST-Clean and GHOST denote the results for public and sensitive samples, respectively.



(a) GHOST vs. ARDEN. (b) GHOST$^+$ vs. SHREDDER.

Fig. 8. Comparison results of inference accuracy for sensitive samples.

Fig. 9. The visualized results of invisibility on MNIST and CIFAR-10.

GHOST. Since the DNN is retrained in GHOST, we test the inference accuracy on the fine-tuned DNN for both the public samples and the sensitive samples. Fig. 6-Fig. 7 show the accuracy with varying ratio γ and steganography methods for different benchmarks. Among all benchmarks, GHOST performs the best and incurs a minor accuracy loss with incremental γ in MNIST that is simple in format and structure. For the remaining benchmarks, the inference accuracy of sensitive samples decreases as the ratio γ increases. However, the ratio γ has minor impact on the inference accuracy of public samples. Furthermore, NNS supports hiding full-size images and lets the stego image contain superabundant information of the hidden image. Therefore, NNS performs better than LSB with limited embedding capacity. The main problem with NNS is that it consumes more edge-side execution time than LSB for constructing the generative dataset $\mathbf{D}_g$.

GHOST$^+$. Since the DNN is pre-trained, the accuracy of GHOST$^+$ for public samples is the same as shown in Table II. The results relevant to sensitive samples are also displayed in Fig. 6-Fig. 7. As GHOST, the ratio γ has a negative impact on the inference accuracy of GHOST$^+$, and NNS has higher accuracy than LSB. Compared with GHOST, GHOST$^+$ requires a little extra costs on edge devices for training the generator network $\mathcal{G}_{\boldsymbol{\omega}}$. For example, it costs about 5min and 20min to train $\mathcal{G}_{\boldsymbol{\omega}}$ (with $\gamma = 1:9$) for MNIST and CIFAR-10, respectively. However, once $\mathcal{G}_{\boldsymbol{\omega}}$ is trained, the execution time for producing perturbations is negligible. In general, GHOST$^+$ is more practical than GHOST since it incurs only minor accuracy loss while keeping the DNN intact.

Performance comparison. From the experiment results, we have a surprising observation that our inference accuracy is even higher than that of the pre-trained accuracy shown in Table II. The reason for such exciting results is that the accuracy in our solutions, in a sense, can be regarded as the ASR of adversarial attacks. The retraining process of GHOST resembles the process of training a poisoned network, and GHOST$^+$ simulates the generation of GAN-based adversarial samples. Our goal is to mislead the DNN to produce our expected result with a high ASR. Compared with the expensive cryptographic techniques [32], [33], our solutions incur much less overheads and have high scalability. To show their effectiveness, we compare GHOST (resp. GHOST$^+$) with ARDEN (resp. SHREDDER) on two datasets, MNIST and CIFAR-10, under different privacy budges. From Fig. 8, we know that both solutions outperform the state-of-the-art solutions as the ratio γ is small. However, as the value of γ increases, our solutions become less competitive. The main reason is that the greater value of γ means the larger amounts of hidden features. For GHOST, it requires a larger-scale neural network of better learning and discrimination power, and for GHOST$^+$, it requires training a stronger generator to generate adversarial perturbations for a variety of sensitive types. It is worth noting that an alternative solution for GHOST$^+$ is to train a generator for each sensitive type with a low ratio γ. However, the larger the scale of neural networks (resp. the more the number of generators), the more cost required for training and inference. Our future work will try to offer an attractive trade-off between performance and cost.

C. Privacy

To empirically evaluate the privacy guaranteed by our solutions, we first use PSNR and SSIM to measure the

TABLE III
THE INVISIBILITY OF HIDDEN IMAGES

Dataset	LSB (PNSR/SSIM)			NNS(PNSR/SSIM)		
	Ave	Max	Min	Ave	Max	Min
MNIST	39/0.99	44/0.99	37/0.99	36/0.99	39/0.99	32/0.99
CIFAR-10	41/0.99	49/0.99	33/0.98	36/0.99	39/0.99	31/0.95
GTSRB	37/0.99	40/0.99	35/0.93	33/0.98	36/0.99	30/0.95
SVHN	43/0.99	49/0.99	36/0.99	36/0.99	40/0.99	34/0.98

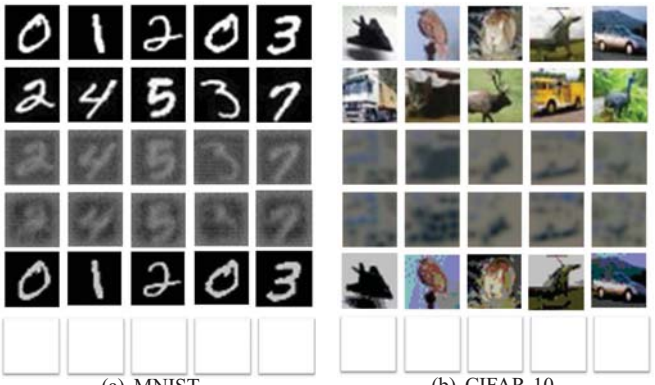
The difference between images is invisible to naked eyes when PNSR is larger than 30 and SSIM is close to 1.

invisibility of hidden images, and then show the robustness of our solutions against feature inversion attacks.

Invisibility. To exhibit the invisibility, we fixed the number of layers deployed on edge devices with $K = 3$. We first consider a powerful attacker that can completely reconstruct the perturbed stego images of GHOST$^+$ using write-box feature inversion attacks. Fig. 9 visualizes the restored images (after removing noise from perturbed stego images using the CDA) based on the MNIST and CIFAR-10 datasets. From this figure, we know that the restored image either looks like the cover image (hiding the sensitive image in secret) or looks very blurry and cannot reveal any useful information about the sensitive image. Then, we consider a stronger attacker that is powerful enough to recover the stego images, by first reconstructing the perturbed stego images using feature inversion attacks, and then restoring the stego images by the CDA. Table III shows the average (Ave), maximum (Max), and minimal (Min) PSNR/SSIM values between cover images and stego images based on four benchmarks. From the quantitative results, we know that it is hard for the observer to detect the difference between cover images and stego images. Therefore, our solutions can protect data privacy in the dark.

The effect of feature inversion attacks. Compared with the black-box setting, white-box feature inversion attacks assume that the attacker has the knowledge of the edge-side network $\mathcal{E}_{\boldsymbol{\theta}}$. Given the intermediate value v output by $\mathcal{E}_{\boldsymbol{\theta}}$, the attacker aims to recover an input x_0 that satisfies the following requirements: (1) $\mathcal{E}_{\boldsymbol{\theta}}(x_0)$ is similar to $\mathcal{E}_{\boldsymbol{\theta}}(x)$, where x is the original input; (2) x_0 follows the same distribution as other inference samples. The goal is equivalent to minimizing the Euclidean distance between $\mathcal{E}_{\boldsymbol{\theta}}(x_0)$ and $\mathcal{E}_{\boldsymbol{\theta}}(x)$, while minimizing the total variation of generated images. This can be formalized as an optimization problem, and can be solved by the regularized maximum likelihood estimation method.

To exhibit the impact of K, we consider two datasets MNIST and CIFAR-10, and launch feature inversion attacks on GHOST, where LSB is used to hide images. Fig. 10 visualizes images reconstructed by feature inversion attacks with varying K. From this figure, we know that the attacker's reconstructive ability degrades as the value of K increases. Furthermore, GHOST achieves better privacy than applying steganography alone. For example, for LSB, the sensitive images are visually observable by directly extracting the last $\zeta = 3$ bits from the stego images, but as K increases to 3, (i.e., the pool1 layers and conv12 layers are deployed locally for MNIST and CIFAR-10, respectively), the reconstructed images are very vague, from which it is hard to extract useful information of



(a) MNIST. (b) CIFAR-10.

Fig. 10. The effect of feature inversion attacks. The 1^{st} row shows the sensitive images, the 2^{nd} row shows the stego images, the 3^{rd} row shows the reconstructive images when $K = 3$, the 4^{th} row shows the reconstructive images when $K = 5$, the 5^{th} row shows the restored images by drawing the last $\zeta = 3$ bits from stego images, and the 6^{th} row shows the restored images by drawing the last $\zeta = 3$ bits from the reconstructive images ($K = 3$).

sensitive images. In GHOST$^+$, the attacker needs to eliminate noise in addition, and thus it is more difficult for the attacker to restore sensitive images. As for NNS, the reveal-network is secret and only known to legal receivers, and the difficulty of restoring sensitive images increases. Therefore, our solutions can effectively resist feature inversion attacks.

IX. CONCLUSION

This paper aims to explore the power of image steganography in protecting data privacy of deep learning. To this end, we propose two private inference solutions, GHOST and GHOST$^+$, both of which employ the traditional LSB and recent NNS techniques to hide sensitive images. To make the DNN output the label of the sensitive image hidden in a stego image, GHOST retrains a poisoned network and GHOST$^+$ generates adversarial inputs, both turning the vulnerability of DNNs against adversarial attacks into a breakthrough for protecting data privacy. Experiment results demonstrate that our solutions can preserve data privacy while guaranteeing a high inference accuracy by creatively integrating steganography and the nature of DNNs. As part of our future work, we will further explore the power of steganography in private deep learning and try to implement our solutions in other domains, such as voice and text, in addition to the vision domain.The authors have provided public access to their code at https://zenodo.org/record/5832391#.YdvYLIgzbid.

ACKNOWLEDGMENTS

This work was supported in part by NSFC grants 61632009, 61872133, and 61802076; NSF grants CNS 1824440, CNS 1828363, and CNS 1757533; the CERNET Innovation Project (NGII20190409); the Guangdong Provincial Natural Science Foundation (No. 2017A030308006), and the Hunan Provincial Natural Science Foundation of China (Grant No. 2020JJ3015).

REFERENCES

[1] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," In *Proc. of CVPR*, 2015.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," In *Proc. of CVPR*, 2016.

[3] M. Ribeiro, K. Grolinger, and M. A. Capretz, "MLaaS: Machine learning as a service.," In *Proc. of ICMLA*, 2015.
[4] Caffe Model Zoo. https://github.com/BVLC/caffe/wiki/Model-Zoo.
[5] BigML. https://bigml.com.
[6] M. A. Rubaie and J. M. Chang, "Privacy-preserving machine learning: threats and solutions," *IEEE Security & Privacy*, 2019.
[7] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. of CCS*, 2015.
[8] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. of CCS*, 2016.
[9] NH. Phan, X. Wu, H. Hu, and D. Dou, "Adaptive laplace mechanism: Differential privacy preservation in deep learning," in *Proc. of ICDM*, 2017.
[10] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv:1712.07557*, 2018.
[11] L. T. Phong and T. T Phuong, "Privacy-preserving deep learning via weight transmission," *IEEE Transactions on Information Forensics and Security*, 2019.
[12] U. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized aggregatable privacy-preserving ordinal response," in *Proc. of CCS*, 2014.
[13] Differential Privacy Team, "Learning with privacy at scale," *Tech. Rep.*, https://machinelearning.apple.com/research/learning-with-privacy-at-scale, 2017.
[14] A. Triastcyn and B. Faltings, "Generating artificial data for private deep learning," *arXiv:1803.03148*, 2018.
[15] C. Orlandi, A. Piva, and M. Barni, "Oblivious neural network computing via homomorphic encryption," *EURASIP Journal on Information Security*, 2007.
[16] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser "Machine learning classification over encrypted data," in *Proc. of NDSS*, 2015.
[17] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. of ICML*, 2016.
[18] P. Mohassel, and P. Rindal, "ABY^3: A mixed protocol framework for machine learning," in *Proc. of CCS*, 2018.
[19] S. Wagh, D. Gupta, and N. Chandran, "SecureNN: 3-party secure computation for neural network training," in *Proc. of PETS*, 2019.
[20] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "Falcon: Honest-majority maliciously secure framework for private deep learning," in *Proc. of PETS*, 2021.
[21] R. Rachuri, and A. Suresh, " Trident: Efficient 4PC framework for privacy preserving machine learning," in *Proc. of NDSS*, 2020.
[22] J. Liu, M. Juuti, Y. Lu, and N. Asokan, " Oblivious neural network predictions via minionn transformations," in *Proc. of CCS*, 2017.
[23] J. Wang, J. Zhang, W. Bao, X. Zhu, and P. S. Yu, "Not just privacy: Improving performance of private deep learning in mobile cloud," in *Proc. of SIGKDD*, 2018.
[24] S. A. Osia, A. Taheri, A. S. Shamsabadi, K. Katevas, H. Haddadi, and H. R. Rabiee, "Deep private-feature extraction," *IEEE Transactions on Knowledge and Data Engineering*, 2018.
[25] S. Leroux, T. Verbelen, P. Simoens, and B. Dhoedt, "Privacy aware offloading of deep neural networks," *arXiv:1805.12024*, 2018.
[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] L. Xiang, H. Zhang, H. Ma, Y. Zhang, J. Ren, and Q. Zhang, "Interpretable complex-valued neural networks for privacy protection," in *Proc. of ICLR*, 2020.
[28] F. Tramer, and D. Boneh, " Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," in *Proc. of ICLR*, 2018.
[29] N. Hynes, R. Cheng, and D. Song, " Efficient deep learning on multi-source private data," *arXiv:1807.06689*, 2018.
[30] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, " Chiron: Privacy-preserving machine learning as a service," *arXiv:1803.05961*, 2018.
[31] H. Hashemi, Y. Wang, and M. Annavaram, " DarKnight: A Data Privacy Scheme for Training and Inference of Deep Neural Networks," *arXiv:2006.01300*, 2020.
[32] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. of STOC*, 2009.
[33] A. C. Yao, "How to generate and exchange secrets," in *Proc. of FOCS*, 1986.
[34] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. of TCC*, 2006.
[35] G. Cormode, S. Jha, T. Kulkarni, N. Li, D. Srivastava, and T. Wang, "Privacy at scale: Local differential privacy in practice," in *Proc. of CCS*, 2018.
[36] S. Wagh, X. He, A. Machanavajjhala, and P. Mittal, "DP-cryptography: Marrying differential privacy and cryptography in emerging applications," *Communications of the ACM*, 2021.
[37] 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*, 2019.
[38] C. K. Chan and L. M. Cheng, "Hiding data in images by simple LSB substitution," *Pattern Recognition*, 2004.
[39] S. Baluja, "Hiding images within images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
[40] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, and Y. Bengio, "Generative Adversarial Networks." In *arXiv preprint arXiv:1406.2661*, 2014.
[41] C. Xiao, B. Li, J. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," in *Proc. of IJCAI*, 2018.
[42] R. Pang, H. Shen, X. Zhang, S. Ji, Y. Vorobeychik, X. Luo, A. Liu, and T. Wang, "A tale of evil twins:adversarial inputs versus poisoned models," in *Proc. of CCS*, 2020.
[43] S. Li, M. Xue, B. Zhao, H. Zhu, X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *Transactions on Dependable and Secure Computing*, 2020.
[44] Z. He, T. Zhang, and R. B. Lee, "Attacking and protecting data privacy in edge-cloud collaborative inference systems," *IEEE Internet of Things Journal*, 2020.
[45] J. Chen and X. Ran "Deep learning with edge computing: A review," *Proceedings of the IEEE*, 2019.
[46] J. Masci, U. Meier, D. Ciresan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proc. of ICANN*, 2011.
[47] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural Networks*, 1995.
[48] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 2009.
[49] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, 2012.
[50] Y. Netzer, T. Wang, A. Coates, A. Bissacco, and B. W. Andrew Y. Ng, "Reading digits in natural images with unsupervised feature learning," In *Proc. of NIPS*, 2011.
[51] V. Costan, I. Lebedev, and S. Devadas, " Sanctum: Minimal hardware extensions for strong software isolation," in *Proc. of USENIX Security*, 2016.
[52] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," In *Proc. of ICML*, 2004.
[53] 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.
[54] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," In *Proc. of CVPR*, 2017.
[55] L. Zhu, R. Ning, C. Wang, C. Xin, and H. Wu, "GangSweep: Sweep out Neural Backdoors by Gan." In *Proc. of ACM MM*, 2020.
[56] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," empharXiv:1703.00810, 2017.
[57] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, 2012.
[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proc. of ICLR*, 2014.