A Fusion-Denoising Attack on InstaHide with Data Augmentation

Xinjian Luo, Xiaokui Xiao, Yuncheng Wu, Juncheng Liu, Beng Chin Ooi

National University of Singapore
{xinjluo, xiaoxk, wuyc, juncheng, ooibc}@comp.nus.edu.sg

Abstract

InstaHide is a state-of-the-art mechanism for protecting private training images, by mixing multiple private images and modifying them such that their visual features are indistinguishable to the naked eye. In recent work, however, Carlini et al. show that it is possible to reconstruct private images from the encrypted dataset generated by InstaHide. Nevertheless, we demonstrate that Carlini et al.'s attack can be easily defeated by incorporating data augmentation into InstaHide. This leads to a natural question: is InstaHide with data augmentation secure? In this paper, we provide a negative answer to this question, by devising an attack for recovering private images from the outputs of InstaHide even when data augmentation is present. The basic idea is to use a comparative network to identify encrypted images that are likely to correspond to the same private image, and then employ a fusion-denoising network for restoring the private image from the encrypted ones, taking into account the effects of data augmentation. Extensive experiments demonstrate the effectiveness of the proposed attack in comparison to Carlini et al.'s attack.

Introduction

Collaborative learning (Yang et al. 2019; Li et al. 2020; Wu et al. 2020) is an increasingly popular learning paradigm as it enables multiple data providers to jointly train models without disclosing their private data. However, recent studies on model inversion attacks (Fredrikson, Jha, and Ristenpart 2015; Hitaj, Ateniese, and Perez-Cruz 2017; Zhu and Han 2020; Luo et al. 2021) demonstrate that the training data can be precisely recovered based on the gradients or model parameters shared during collaborative learning. This leads to concerns on the security of existing collaborative learning methods (Huang et al. 2020; Kairouz et al. 2019). To mitigate the above concerns, (Huang et al. 2020) propose a practical scheme, InstaHide, which generates the training datasets by mixing multiple private images into one image (Zhang et al. 2017). The training images produced by InstaHide are called *encryptions* in (Huang et al. 2020). Intuitively, InstaHide aims to corrupt the visual features of the private images (as shown in Fig. 1) such that the encrypted training images fed into the models are hardly distinguishable by the naked eye, thus eliminating the threats caused by inversion attacks (Hitaj, Ateniese, and Perez-Cruz 2017; Zhu and Han 2020).

Recently, however, (Carlini et al. 2020) propose an attack that can approximately recover the private images encrypted by InstaHide. The main idea of (Carlini et al. 2020) is to first cluster the encrypted images based on a similarity metric, and then restore one private image from one cluster of encryptions by factoring out the useless components. Although this attack works well against the InstaHide Challenge dataset (ins 2020a), there are three main limitations. *First*, (Carlini et al. 2020) is specially designed for the InstaHide Challenge, where each private image is directly mixed into $T = 50$ encryptions. But in applications that T is set to a small number (e.g., 10), the performance of (Carlini et al. 2020) is greatly degraded (as pointed out by one author of InstaHide (Arora 2020)). *Second*, the private images could be pre-processed by data augmentation before mixing with other images (this case is included in the source code of InstaHide (ins 2020b) instead of the challenge dataset (ins 2020a)), and (Carlini et al. 2020) can barely restore distinguishable images. *Third*, (Carlini et al. 2020) can not precisely restore the original color profiles of the private images, which would degrade the visual features of the restored images and lead to indistinguishable results. In this paper, we investigate a more restricted but more practical problem: *how to precisely restore the visual structures and color profiles of a private image from a small number of encryptions generated by InstaHide with data augmentation*?

To address this problem, the general idea is first to determine a set of encryptions that contain the same private image (called *homogeneous encryptions*), then restore the private image based on these homogeneous encryptions. In particular, we view the component produced by irrelevant mixed images in an encryption as noise. Although the noise pattern is hard to be mathematically formulated because of the randomly nonlinear variations on the mixed pixels introduced by InstaHide, it can be learned effectively by a deep neural network. In this way, we can use a trained network to remove the noise component and accurately restore the color profiles and structures of the private image from a small number ($\ll 50$) of encryptions.

Implementing such a neural network is not trivial. Without careful design, the restored images could be meaningless, as shown in (Huang et al. 2020). The main difficulty is that

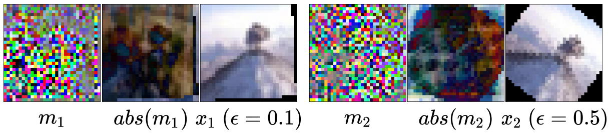


Figure 1: Examples generated by InstaHide with data augmentation. m_i: the original encryption; $abs(m_i)$: the image after removing all pixel signs of m_i; x_i: the original private image. ϵ is defined in the Preliminary section.

the private image could be randomly transformed by geometrical data augmentation before being mixed into multiple encryptions. Since the salient structures of the private image are severely corrupted after being processed by InstaHide (Fig. 1), the widely used image registration methods (Zitova and Flusser 2003; Ma, Ma, and Li 2019) that rely on visual features to geometrically align the structures of multiple images are hardly useful. Therefore, we have to design an image registration method from scratch to align the salient structures. In addition, we need to handle the case that one image is mixed $\ll 50$ times, in which the pixel-wise optimization method used in (Carlini et al. 2020) can not work because the information provided by the corresponding encoded pixels ($\ll 50$) that are derived from the same private pixel p is not sufficient to recover p (as shown in Fig. 4a). We need to consider a patch-wise restoration method in which the neighboring information of p is used for its restoration.

To overcome these difficulties, we take the attack on InstaHide with data augmentation as an image fusion-denoising task whose inputs are not pre-registered and severely corrupted, and design a registration-fusion-denoising pipeline to handle this task. We first devise a network component called *image relaxing* to automatically align the severely corrupted private images. Image relaxing can also reduce the noises caused by the structures of other irrelevant mixed images. We further give an extensive analysis of the noise pattern introduced by InstaHide, which inspires us that the corruption levels of the private image can be reflected by the pixel variance. Accordingly, we propose a re-weighting method based on the image variance to pre-process the encryptions before feeding them into the neural network. Following these insights, we then design a novel **F**usion-**D**enoising **N**etwork (FDN) to fuse several homogeneous encryptions into a single encryption and denoise this encryption to recover the private image. To our knowledge, this is the first work that utilizes a registration-fusion-denoising pipeline to solve the image reconstruction tasks based on the inputs with not pre-registered and severely corrupted visual features. We conduct extensive experiments to evaluate the generalization and attack performance of FDN. The results demonstrate the superior performance of the proposed scheme to (Carlini et al. 2020).

Related Work

Image fusion is used to integrate the complementary features of multiple images into one informative image (Zhang et al. 2020a). Before a fusion, the images capturing the same scene but in different perspectives should be geometrically aligned, which is known as *image registration*. The traditional registration studies (Ma, Ma, and Li 2019), which mainly focus on extracting and aligning salient structures, such as edges and corners, are barely useful if the image structures are severely corrupted. Most fusion studies (Ma, Ma, and Li 2019; Zhang, Bai, and Wang 2017; Liu, Liu, and Wang 2015; Zhang et al. 2020a) assume that the images input to the fusion algorithms are pre-registered, and few of them consider the impact of image noise. Although a few studies consider joint image fusion and denoising (Li et al. 2018; Liu, Xu, and Fang 2020; Mei, Dong, and Huang 2019), they assume that the visual features of the input images are pre-aligned and not corrupted by the noise, which is not applicable for attacking the InstaHide with data augmentation.

Mixup is proposed as a regularization method for neural network training (Zhang et al. 2017). Since Mixup can obfuscate the visual features of images, some recent studies (Fu et al. 2019; Raynal, Achanta, and Humbert 2020; Zhang and Luo 2021) employ it to pre-process the raw training data for privacy-preserving. However, (Huang et al. 2020) demonstrate that one private image could be simply restored by averaging those mixup images containing it. Accordingly, (Huang et al. 2020) propose InstaHide to enhance the security of Mixup. But (Carlini et al. 2020) devise an attack that can restore distinguishable images from the InstaHide Challenge dataset (ins 2020a) by minimizing the norm of the noise component. Nevertheless, (Carlini et al. 2020) is specifically designed for the challenge dataset, which is not general and can be easily defeated by incorporating data augmentation into InstaHide. On the contrary, our registration-fusion-denoising pipeline has better generalization and can be easily extended to the related image restoration tasks without major modifications.

Preliminary

InstaHide. Given two private images x_1, x_2 and their corresponding one-hot labels y_1, y_2, InstaHide mixes x_1 and x_2 with $k-2$ public images to get a mixup image, and randomly flips the pixel signs of this mixup image to obtain the final encryption m, i.e.,

$$m = \sigma \circ (\lambda_1 x_1 + \lambda_2 x_2 + \sum_{i=3}^{k} \lambda_i u_i), \qquad (1)$$

where λ_i ($i \in \{1, \cdots, k\}$) is randomly sampled from $[0, 1]$ such that $\sum_{i=1}^{k} \lambda_i = 1$, and all the images are normalized into $[-1, 1]$ beforehand. σ is a one-time pad mask uniformly sampled from $\{+1, -1\}$, and $\circ$ denotes the element-wise multiplication. Accordingly, the label of m becomes $y_m = \lambda_1 y_1 + \lambda_2 y_2$. The mixup pair (m, y_m) is used to train the desired deep neural networks. Notice that the $k-2$ public images u_i ($i \in \{3, \cdots, k\}$), randomly sampled from a public dataset (e.g., ImageNet (Deng et al. 2009)), are mainly used to corrupt the visual features of x_1 and x_2, such that another party who obtains m can not discern the original private images. As the public images are useless to the downstream classification tasks, we call the $\sum_{i=3}^{k} \lambda_i u_i$ term in Eq. 1 the *noise component*.

Carlini et al.'s Attack on InstaHide. Carlini et al. (Carlini et al. 2020) propose an attack to restore the private images $\mathcal{X}$

contained in the InstaHide Challenge dataset $\mathcal{M}$ (ins 2020a). The main idea is to first cluster the Challenge dataset, such that the encryptions in the same cluster contain the information of the same private image $x \in \mathcal{X}$. After that, for each cluster of encryptions, a gradient optimization method is employed to recover the private images by minimizing the ℓ_2 norm of the noise component corresponding to the public images. Specifically, let a private image x be a d-dimensional vector, A be a $|\mathcal{X}| \times d$ private image matrix, B be a $|\mathcal{M}| \times d$ encryption matrix and C be a $|\mathcal{M}| \times |\mathcal{X}|$ coefficient matrix, i.e., each row of A denotes a private image x, and each row of B denotes an encryption image. Therefore, Eq. 1 can be rewritten as $\sigma \circ (C \cdot A + \delta) = B$, where δ denotes the noise component. By preserving only the absolute pixel values, the randomness of pixel signs caused by σ can be removed: $abs(C \cdot A + \delta) = abs(B)$. Note that it is difficult to directly solve A from this equation given $abs(B)$ and C as the unknown noise component δ can significantly change the distribution of $C \cdot A$. Instead, (Carlini et al. 2020) proposes to solve a modified minimization problem:

$$\underset{A' \in [-1,1]^{|\mathcal{X}| \times d}}{\arg\min} ||\delta||_2^2 \quad \text{s.t. } C \cdot abs(A') + \delta = abs(B). \tag{2}$$

However, there is a main defect in Eq. 2 that $abs(C \cdot A) \neq C \cdot abs(A)$, for example, $abs\left([0.5 \quad 0.3] \cdot [-0.8 \quad 1]^{\intercal}\right) = 0.1$ whereas $[0.5 \quad 0.3] \cdot abs\left([-0.8 \quad 1]^{\intercal}\right) = 0.7$. Consequently, it will produce images with obvious color shifts, even leading to indistinguishable images (Fig. 4).

InstaHide with Data Augmentation. Similar to MixMatch (Berthelot et al. 2019), let $\mathcal{X} = \{x_1, \cdots, x_N\}$ be a private image set, where N is the number of private images. Before employing InstaHide, we conduct data augmentation on each image x_i to generate a transformed dataset $\hat{\mathcal{X}} = \{\hat{x}_1, \cdots, \hat{x}_{N \times K}\}$. Specifically, we generate $K-1$ augmentations for each image x_i: $\hat{x}_{i,j} = \text{Augment}(x_i), j \in \{1, \cdots, K-1\}$. Meanwhile, we denote x_i by $\hat{x}_{i,0}$. All $\hat{x}_{i,j} (i \in \{1, \cdots, N\} \wedge j \in \{0, \cdots, K-1\})$ are flattened into $\hat{\mathcal{X}}$. After that, we shuffle $\hat{\mathcal{X}}$ to get $\hat{\mathcal{S}} = \{\hat{s}_1, \cdots, \hat{s}_{N \times K}\}$ and apply InstaHide on $\hat{\mathcal{X}}$ and $\hat{\mathcal{S}}$ to generate the encryption dataset $\mathcal{M} = \{m_1, \cdots, m_{N \times K}\}$ with

$$m_i = \text{InstaHide}(\hat{x}_i, \hat{s}_i, u_{i,3}, \cdots, u_{i,k}), \forall i \in \{1, \cdots, N \times K\}, \tag{3}$$

where $\{u_{i,3}, \cdots, u_{i,k}\}$ are $k-2$ random public images, and k is the mixup parameter in InstaHide. Accordingly, the labels $\mathcal{Y}_{\mathcal{M}} = \{y_1, \cdots, y_{N \times K}\}$ of $\mathcal{M}$ can be obtained: $y_i = \lambda_1 y_{\hat{x}_i} + \lambda_2 y_{\hat{s}_i}$, where $y_{\hat{x}_i}$ and $y_{\hat{s}_i}$ denote the one-hot labels of $\hat{x}_i$ and $\hat{s}_i$; λ_1 and λ_2 are the corresponding random coefficients. Consequently, $\mathcal{M}$ and $\mathcal{Y}_{\mathcal{M}}$ are used as the training dataset for classification tasks.

For data augmentation, we consider the *geometric transformations*, e.g., random cropping, rotation, and translation, instead of noise injection and color transformation. Specifically, the former method is widely adopted in deep learning (Ooi et al. 2015; Shorten and Khoshgoftaar 2019) (also included in the code of InstaHide (ins 2020b)), and it can change the structures of images, bringing more difficulty to the restoration work since we have to align the structures of multiple transformed images before restoring the original one. In contrast, the effects of the latter methods are trivial and can be generally covered by the mixup noise introduced by InstaHide. Note that the GAN-based augmentation methods, which typically synthesize new images that are not included in the original private dataset, can be regarded as an upstream task of geometric transformations (Shorten and Khoshgoftaar 2019). To better investigate the impact of geometric transformations on the security of InstaHide, we formally define the augmentation level ϵ based on the pixel displacement:

Definition 1 (ϵ-augmentation)**.** Given an image x with size $W \times H$ and its augmented version $\hat{x}$. Assume a pixel p^x in x has coordinate $C_{p^x} = (w_{p^x}, h_{p^x})$; and a pixel $p^{\hat{x}}$ in $\hat{x}$ has coordinate $C_{p^{\hat{x}}} = (w_{p^{\hat{x}}}, h_{p^{\hat{x}}})$. Then $\hat{x}$ is an ϵ-augmentation of x if for any possible pixel pairs $(p^{\hat{x}}, p^x)$ that $p^{\hat{x}}$ is transformed from p^x, $w_{p^{\hat{x}}} \in [w_{p^x} - \frac{\epsilon}{2}W, w_{p^x} + \frac{\epsilon}{2}W]$ and $h_{p^{\hat{x}}} \in [h_{p^x} - \frac{\epsilon}{2}H, h_{p^x} + \frac{\epsilon}{2}H]$ hold.

Fig. 1 shows an example of images with different data augmentation levels. Generally, the higher an ϵ is, the larger degree an image will be transformed with. For example, $\epsilon = 0.5$ corresponds to shifting x left for $W/4$, or cropping x to $3W/4 \times 3H/4$.

The Proposed Attack on InstaHide with Data Augmentation

We consider a threat model in which the attacker aims to restore the data owner's private images $\mathcal{X}$ based on the published encryptions $\mathcal{M}$ and labels $\mathcal{Y}_{\mathcal{M}}$. We assume that the $(\mathcal{M}, \mathcal{Y}_{\mathcal{M}})$ is accessible to the attacker since (Huang et al. 2020) claims that a data owner can directly send these data to another party for training desired models. Based on $\mathcal{M}$ and $\mathcal{Y}_{\mathcal{M}}$, our attack consists of three steps (Fig. 2). In the *absolute pre-processing* step, we remove the mask σ (see Eq. 1) by conducting $abs(m_i), \forall m_i \in \mathcal{M}$. σ renders the signs of mixup pixels to useless noise, yet failing to change the absolute pixel values. These absolute values can be utilized by the restoring algorithms. In the *encryptions clustering* step, we find a candidate set of encryptions M from $\mathcal{M}$ containing the information of the same private image x_i. In the *image restoring* step, we use a fusion-denoising network (FDN) to restore x_i from the homogeneous encryption set M.

Pixel-Level Noise Pattern

InstaHide is a pixel-wise mixup scheme: the pixels located in the same position of different images are linearly combined into a mixup pixel, and the sign of this mixup pixel is randomly flipped. Formally, a pixel $p^{m_{i,l}}$ of an encrypted image $m_{i,l}$ is computed by: $p^{m_{i,l}} = \sigma \circ (\lambda_1 p^{\hat{x}_i} + \lambda_2 p^{\hat{s}_i} + p^{\delta})$, where $p^{\hat{x}_i}$ and $p^{\hat{s}_i}$ are pixels from private images $\hat{x}_i$ and $\hat{s}_i$. Without loss of generality, we assume $\hat{x}_i$ as the *target image* that are commonly shared among a homogeneous encryption set, and the other image $\hat{s}_i$ as another source of noise. After conducting $abs(m_{i,l})$, the above equation can be rewritten as: $abs(p^{m_{i,l}}) = abs(\lambda_1 p^{\hat{x}_i} + p^{\delta})$. Now the task becomes given $p^{m_{i,l}}$ and λ_1 (can be inferred from the one-hot labels), the adversary aims to restore the value of $p^{\hat{x}_i}$. This task is difficult if given only one encryption because p^{δ} and the

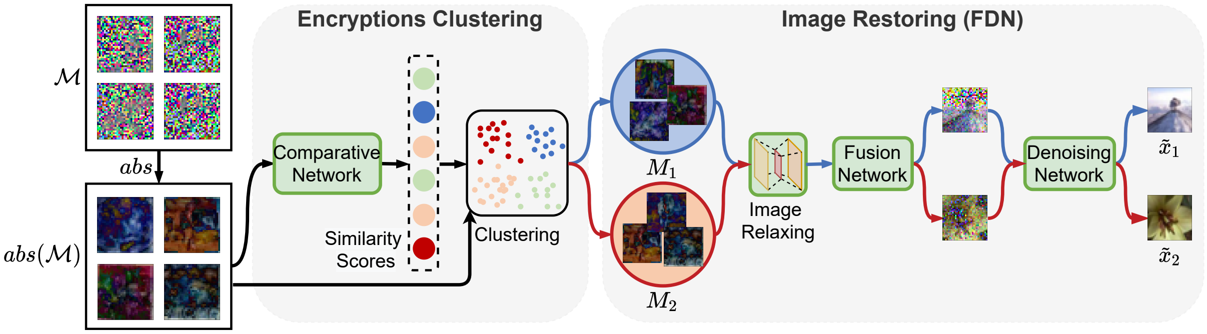


Figure 2: Overview of the proposed attack.

sign of $(\lambda_1 p^{\hat{x}_i} + p^\delta)$ are both unknown. We assume that the noise component p^δ follows a type of isotropic Gaussian distribution, which is reasonable since p^δ is initially a linear combination of $k-1$ images. Consequently, based on multiple $abs(p^{m_{i,l}})$ derived from a same $p^{\hat{x}_i}$, we can first roughly infer the signs of the corresponding $(\lambda_1 p^{\hat{x}_i} + p^\delta)$ by a neural network and then factor out the noise p^δ by averaging these $abs(p^{m_{i,l}})$ for restoring the original $p^{\hat{x}_i}$.

The problem is how to find those $abs(p^{m_{i,l}})$ derived from a same $p^{\hat{x}_i}$. This is rather challenging in InstaHide with data augmentation, because the locations of corresponding $abs(p^{m_{i,l}})$ are mostly different from the original location of $p^{\hat{x}_i}$ after geometric transformations, and the visual features of the encryptions are barely useful for determining these transformations. We observe that the neighboring pixels in an image patch typically change smoothly, i.e., their values are roughly the same, which means that the neighboring pixels of $\hat{x}_i$ can be used to align and recover $\hat{x}_i$. We therefore design an image-relaxing structure in the fusion phase to automatically diffuse the information of neighboring pixels into overlapping patches (information alignment), then use a window-based loss function in the denoising phase to patch-wisely restore the original image.

Clustering Mixup Images

To restore a private image x_i, we need to find a possible encryption set $M = \{m_{i,l}, l \in \{0, \cdots, a\}\}$ containing x_i or its transformed versions $\hat{x}_{i,j}$ where $j \in \{1, \cdots, K-1\}$. In this phase, we follow the clustering step in (Carlini et al. 2020), i.e., splitting the encryptions $\mathcal{M}$ into multiple clusters such that the encryptions in each cluster contain the information of the same image. Note that the idea, i.e., first clustering the encryptions then recovering the corresponding private images, is inescapable for attacking InstaHide, because it is impossible to recover the original private image from only a single encryption given the random and severe corruptness (Huang et al. 2020; Carlini et al. 2020).

(Carlini et al. 2020) uses a ResNet-28 to compute the similarity score for each pair of encryptions, which performs poorly in InstaHide with data augmentation. Because the periphery pixels produced by data augmentation are mostly useless for similarity comparison (Fig. 1), yet severely degrading the comparison performance of ResNet-28 since it tries to remember all the peripheral patterns of training data. Therefore, we design a new comparative network for computing the similarity scores (Fig. 5). Specifically, the multi-resolution information, which has been demonstrated beneficial in image comparison tasks (Zagoruyko and Komodakis 2015), is used to help the network pay more attention to the central pixels than the periphery pixels. For a 32×32 image, we generate two 16×16 images with different resolutions: the first image is generated by cropping the central part of the original image (high resolution), and the second image is generated by downsampling at half the original image (low resolution). For each pair of encryptions, we first generate a high-resolution pair and a low-resolution pair, then feed them into residual blocks (He et al. 2016). The results are concatenated and fed into a dense layer for computing the final similarity score. As a result, in our experiments, the testing accuracy of the proposed network can reach 92% under $\epsilon = 0.2$, whereas the accuracy of the original ResNet-28 reaches up to 71%.

Additional Filtering. After clustering, we obtain $|\mathcal{X}|$ clusters and each cluster consists of $|M|$ homogeneous encryptions, where $M = \{m_{i,l}, l \in \{0, \cdots, a\}\}$. In the experiments, we find that the encryptions with a large ϵ (e.g., rotated for 90 degrees), contribute little to or even degrade the restoration performance. This is because the structures of private images in these encryptions are difficult to be aligned with structures of other private images. Thus, we propose an additional filtering step to retain the neighboring encryptions in M such that the ϵ difference between any two neighboring encryptions is less than a threshold t_ϵ with a high probability. Specifically, we train a filtering model based on an encryption dataset, where the encryption pairs with ϵ difference less than t_ϵ are labeled with 1, and otherwise are labeled with -1. For each cluster M, we first use this filtering model to find all neighboring encryptions for each $m_{i,l} \in M$, then only keep the encryption m with most neighbors (together with its neighbors) in M. As a result, we can guarantee that the ϵ difference of any two encryptions in M is less than $2t_\epsilon$ as each encryption differs from m by at most t_ϵ. We conduct experiments with different t_ϵ and find the filter with $t_\epsilon = 0.2$ achieves a good trade-off between more homogeneous encryptions and less transformation after filtering M.

Restoring Private Images

After obtaining a homogeneous encryption set M, we first use a re-weighting method to pre-process each encryption $m_{i,l} \in M$, then feed them into a fusion-denoising network to recover the target $\hat{x}_i$.

Re-weighting. Notice that the coefficients λ_1 of $\hat{x}_i$ are different in different encryptions. The randomness of λ_1 may restrain the network from learning the correct pixel values of $\hat{x}_i$. To reduce the uncertainty introduced by λ_1, we rescale all encryptions by $1/\lambda_1$, i.e., for the pixels $p^{m_{i,l}}$ of $m_{i,l}$, we compute $abs(p^{m_{i,l}}/\lambda_1) = abs(p^{\hat{x}_i} + (p^{\delta}/\lambda_1))$. Note that after rescaling, the corruptness levels of $\hat{x}_i$ are different in different encryptions. For example, when $\lambda_1 = 0.4$ or 0.25, the noise p^{δ} will be enlarged by a factor of 2.5 or 4, respectively. We further observe that the noise level p^{δ}/λ_1 can be reflected by the variance of an encryption: assume $p^{\hat{x}_i}$ and p^{δ} are independent, then $\text{Var}(p^{\hat{x}_i} + \frac{p^{\delta}}{\lambda_1}) = \text{Var}(p^{\hat{x}_i}) + \frac{\text{Var}(p^{\delta})}{\lambda_1^2}$, which indicates that the larger λ_1 is, the smaller the variance of the encryption will be. Based on this observation, we further re-weight the encryptions based on their variances. Specifically, we compute the variances $\text{Var}(m_{i,l})$ for each $m_{i,l} \in M$, then re-weight $m_{i,l}$ by a factor of $\beta = \frac{\min(Var(m_{i,0}),\cdots,Var(m_{i,a}))}{Var(m_{i,l})}$. The factor β can ensure that the pixels of the encryption with the smallest variance (i.e., with the least corruptions) stay the same, while those with a larger variance are reduced since they contain more noise and provide less information of $\hat{x}_i$. The effects of this re-weighting method are evaluated in the ablation study.

Image Fusion and Denoising. To accurately restore $\hat{x}_i$, we need to utilize all the information provided by each $m_{i,l} \in M$, i.e., fusing the information of these encryptions. Recall that the $\hat{x}_i$ contained in an encryption $m_{i,l}$ could be either the original one or the transformed one. Before fusion, we need to geometrically align these encryptions based on their respective target images (Ma, Ma, and Li 2019). The traditional methods (Rublee et al. 2011; Ma, Ma, and Li 2019; Lowe 2004) are hardly useful since they rely on the visual features which are severely corrupted in this case (Fig. 1). Inspired by (Dosovitskiy and Brox 2016), we design an efficient network component, called *image relaxing*, to automatically align the information of target images. Suppose the size of an encryption is $W \times H \times 3$. Before fusing the encryptions, we feed them into a convolutional layer with a stride of 2, resulting in a feature map with size $\lceil W/2 \rceil \times \lceil H/2 \rceil \times c$ (downsampling, c is the number of filters). After that, we upsample this feature map to the full image size $W \times H \times c$ by a transposed convolutional layer with a stride of 2. The downsampling step can produce translation-invariant features, and the upsampling step can capture the high-level structures.

To help illustrate the effects of image relaxing, we first extract the features of two transformed images via a trained relaxing component and a 3×3 convolutional kernel, then simply fuse the corresponding features by averaging them (Fig. 3a). We observe that the features extracted by normal convolutions preserve more details, e.g., edges and corners, but the object structures are corrupted in the fused features because they are not properly aligned. In the features extracted by image relaxing, some details are lost, but the original structures experience less corruption after the fusion. Specifically, by downsampling and upsampling, image relaxing can transmit the information of a single pixel in the original image into a patch of neighboring pixels in the feature maps, which makes the information alignment easier, i.e., from point-wise alignment to patch-wise alignment. In addition, one encryption typically contains some structures irrelevant to the target private image. Image relaxing can lessen the impact of these irrelevant structures by downsampling, whereas normal convolutions preserve these structures and cause many artifacts on the fused images.

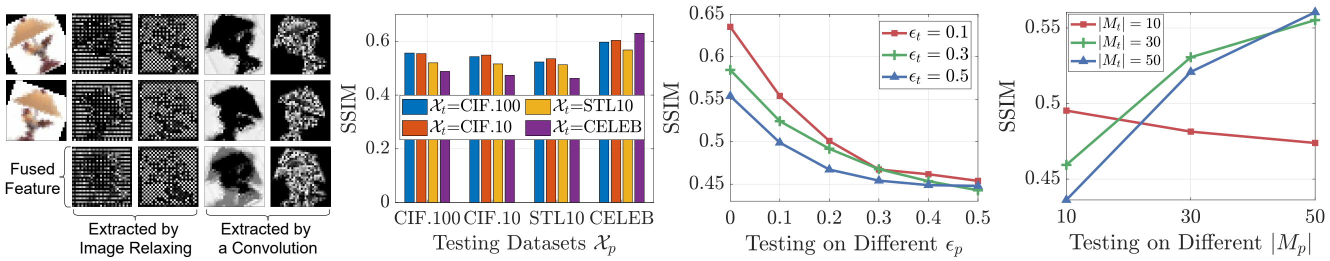
After extracting target features from multiple encryptions, we need to fuse these features based on a fusion rule. The two widely used rules are choose-max and average (Ma, Ma, and Li 2019). In the experiments, we find that the choose-max rule performs better when $|M| \leq 10$; while the average rule achieves better results when $|M| > 10$. The reason is that the average rule could hardly factor out the noise p^{δ} based on a limited number (e.g., less than 10) of encryptions, if some outliers, i.e., severely corrupted encryptions with large p^{δ}, exist. While the choose-max rule mainly focuses on restoring the least corrupted $\hat{x}_i$ contained in the encryptions with larger pixel values (not reduced in the re-weighting phase), mitigating the impact of outliers. Since the possible number $|M|$ of homogeneous encryptions input to FDN could not be determined beforehand, we design a multiple-channel fusion architecture (Fig. 6) to accept a variable number of encryptions as the input (all branches share the same set of parameters). After fusing multiple encryptions, we use a denoising network to restore the original private image. Among multiple denoising networks (such as (Mao, Shen, and Yang 2016; Zhang et al. 2020b, 2019)), we find RNAN (Zhang et al. 2019) performs best in this task since it can capture the long-range dependencies between channels and pixels in the whole image, which is important for FDN to learn the overall noise distribution and restore private images with accurate color profiles.

Loss Function. The mean structural similarity index (MSSIM) (Wang et al. 2004) performs far better than ℓ_1 or ℓ_2 loss in our task, since MSSIM compares the local difference between two images over a sliding window, which facilitates the network to neglect the overall structure distortion caused by other mixed images and concentrate on restoring local structures; whereas the other two losses tend to average all possible color modes and restore blurry images. Since the ℓ_1 loss can facilitate the recovery of pixel intensities (Zhao et al. 2016), we compute the network loss by combining the ℓ_1 loss and MSSIM:

$$\mathcal{L} = \lambda_{\text{MSSIM}}\mathcal{L}_{\text{MSSIM}} + (1 - \lambda_{\text{MSSIM}})\mathcal{L}_{\ell_1}. \quad (4)$$

Experiments

Setup. We use CIFAR10 (Krizhevsky and Hinton 2009), CIFAR100 (Krizhevsky and Hinton 2009), STL10 (Coates, Ng, and Lee 2011) and CelebFaces (CELEBA) (Liu et al. 2015) as the training and testing datasets. The λ_{MSSIM} in Eq. 4 is empirically set to 0.7. The InstaHide parameterized with $k = 6$ is employed unless otherwise specified. We compare our scheme with two baselines: the Carlini et al.'s original attack (CA) (Carlini et al. 2020), and a modified CA with the ResNet-28 used in the clustering phase replaced by our comparative network (CA-CN). Besides, the MSSIM with window size 8 (SSIM for short) is used to measure the similarity between the restored images and the ground truth images.



(a) Image Relaxing (b) Private Datasets $\mathcal{X}$ (c) ϵ-augmentation (d) $|M|$

Figure 3: (a) a comparison between image relaxing and normal convolutions; (b-d) the generalization performance of FDN.

Note that the ℓ_1 and ℓ_2 (MSE) losses are not appropriate for the similarity evaluation in this paper, because they are pixel-wise metrics and a slight geometric transformation in the restored images could greatly change the results of them. More setting details are reported in the arXiv version.

Generalization

There are three hyper-parameters in FDN: the private image set $\mathcal{X}$ used for generating the training dataset of FDN, the number $|M|$ of homogeneous encryptions in each cluster (i.e., the number of inputs to FDN), and the data augmentation level ϵ. We now demonstrate the generalization of FDN with respect to the three hyper-parameters. Specifically, we investigate: given a set of encryptions $\mathcal{M}_p$ which is generated from a private dataset ($\mathcal{X}_p$) with an unknown level of augmentation (ϵ_p) and an unknown number of homogeneous encryptions derived from a same private image ($|M_p|$), whether the adversary can restore $\mathcal{X}_p$ via an FDN trained on another encryption set $\mathcal{M}_t$ generated from different $\mathcal{X}_t$, $|M_t|$, and ϵ_t.

Generalization w.r.t. different datasets $\mathcal{X}_p$. We fix $|M_p| = |M_t| = 10$, $\epsilon_p = \epsilon_t = 0.1$, and generate two encryption datasets $\mathcal{M}_p$ and $\mathcal{M}_t$ based on two private datasets $\mathcal{X}_p$ and $\mathcal{X}_t$. We first train an FDN based on $\mathcal{M}_t$ and then use it to restore the $\mathcal{X}_p$ from $\mathcal{M}_p$. From Fig. 3b, we observe that the FDN trained on CIFAR10 achieves the best restoration performance among the first three testing datasets, while the FDN trained on CELEBA performs worst. Because the image patterns in CIFAR10 are generally the most complicated among these datasets, which can help the network learn to restore images with complicated distributions; whereas the image patterns of CELEBA (human faces) are more simple and predictable, making the networks trained on it perform worse on restoring more complicated images. Note that when testing on the same dataset, the performance of the FDNs trained on different datasets remains roughly the same (suffering at most 6% degradation), demonstrating FDN's good generalization ability with respect to different datasets.

Generalization w.r.t. different data augmentation levels ϵ_p. We fix $|M_p| = |M_t| = 10$ and $\mathcal{X}_p = \mathcal{X}_t =$ CIFAR100 (80% for $\mathcal{X}_p$ and 20% for $\mathcal{X}_t$; this setting is used when $\mathcal{X}_p = \mathcal{X}_t$), then train and test an FDN based on two encryption datasets generated with two different augmentation levels ϵ_t and ϵ_p. From Fig. 3c, we see that with the increase of ϵ_p, the performance of FDN degrades. Because a larger ϵ_p represents a larger transformation to a private image, indicating that the structures of the private images contained in input encryptions are harder to be registered. Note that the FDN trained on $\epsilon_t = 0.1$ achieves better performance than other models. The reason is that when trained on a dataset with smaller transformations, the FDN learns to restore more image details instead of focusing on registering image structures. Nevertheless, when tested on a specific ϵ_p, different FDNs perform similarly, demonstrating the good generalization of FDN under different augmentation levels.

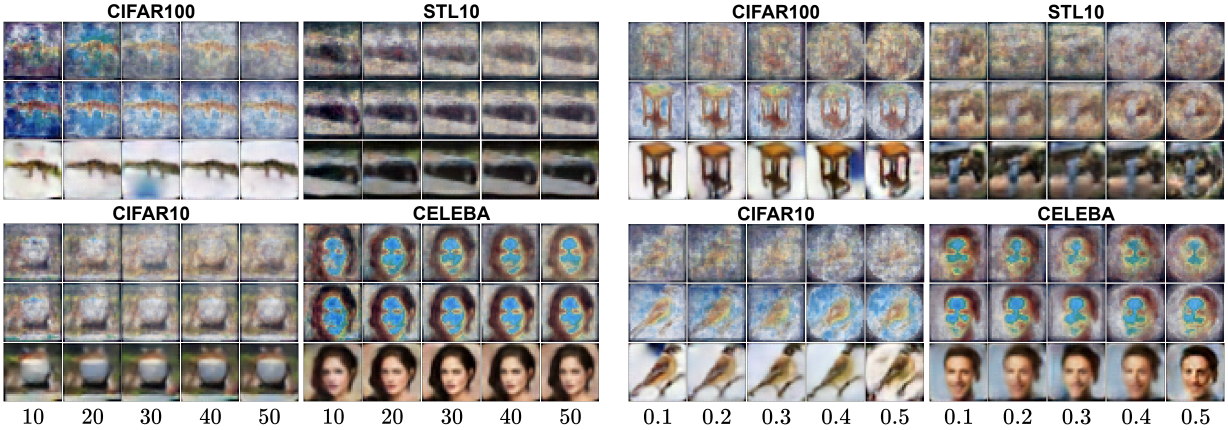
Generalization w.r.t. different number of inputs $|M_p|$. We first fix $\mathcal{X}_p = \mathcal{X}_t =$ CIFAR100 and $\epsilon_p = \epsilon_t = 0.2$, then train and test an FDN under different $|M_p|$ and $|M_t|$. From Fig. 3d, we see that with the increase of $|M_p|$, the testing performance of FDNs trained on $|M_t| = 30$ and 50 improves; while the performance of the FDN trained on $|M_t| = 10$ slightly degrades. This is because we use the choose-max and average rules to train FDNs with $|M_t| \leq 10$ and $|M_t| > 10$, respectively. The choose-max rule is more robust under severely corrupted encryptions than the average rule, producing images with better quality when $|M| \leq 10$. While for $|M| > 10$, the average rule can learn more details than the choose-max rule. But when tested on the same $|M_p|$, the performances of FDN trained on $|M_t| = 30$ or 50 are similar. This shows the flexibility of FDN, i.e., the adversary can train an FDN based on different $|M_t|$ without worrying about the quality degradation of restored images.

Comparison with Carlini et al.'s Attack

Different numbers of input mixups. In this set of experiments, we fix $\epsilon_p = \epsilon_t = 0.1$, then train an FDN based on $\mathcal{X}_t =$ CIFAR10 and $|M_t| = 30$, and test it under different $|M_p|$ (i.e., number of input encryptions to FDN). We show the SSIM results in Tab. 1 and some restored images in Fig. 4a. From Tab. 1, we see that the modified attack CA-CN performs better than the original attack CA. The reason is that the ResNet used in CA can produce plenty of false positive images (i.e., not containing the target private image) in the same cluster, which brings considerable noise to the input of the restoration phase and renders the final images indistinguishable. Our comparative network in CA-CN can reduce the false positive cases and transmit more useful information to the restoration algorithm. In addition, with the increasing of $|M_p|$, both FDN and CA can restore the private images with increasing quality. This is expected since more input encryptions can provide more details of the private images. Also, FDN performs better than CA, which can be clearly demonstrated by the examples in Fig. 4a. The substantial

Dataset	Attack	Different $\lvert M\rvert$					Different ϵ				
		10	20	30	40	50	0.1	0.2	0.3	0.4	0.5
CIFAR100	CA	0.3433	0.3776	0.3824	0.3951	0.4051	0.3917	0.3304	0.3150	0.2986	0.2781
	CA-CN	0.4085	0.4580	0.4657	0.4827	0.4961	0.5062	0.4416	0.4041	0.3939	0.3552
	FDN	**0.5565**	**0.5936**	**0.6229**	**0.6327**	**0.6458**	**0.6618**	**0.6208**	**0.6009**	**0.5507**	**0.5487**
CIFAR10	CA	0.3831	0.4165	0.4168	0.4203	0.4335	0.4167	0.3503	0.3259	0.3133	0.2877
	CA-CN	0.4677	0.4941	0.5106	0.5206	0.5277	0.5073	0.4515	0.4219	0.4087	0.3674
	FDN	**0.5490**	**0.5844**	**0.6133**	**0.6289**	**0.6404**	**0.6384**	**0.6029**	**0.5795**	**0.5449**	**0.5320**
STL10	CA	0.4098	0.4382	0.4430	0.4495	0.4530	0.4430	0.3561	0.3378	0.3161	0.2849
	CA-CN	0.5057	0.5312	0.5449	0.5465	0.5636	0.5636	0.4902	0.4441	0.4303	0.3646
	FDN	**0.5130**	**0.5622**	**0.5923**	**0.6091**	**0.6307**	**0.6429**	**0.5872**	**0.5612**	**0.5111**	**0.4958**
CELEBA	CA	0.3775	0.3872	0.3897	0.3980	0.4039	0.3981	0.3290	0.3111	0.2997	0.2793
	CA-CN	0.4593	0.4843	0.4954	0.5018	0.5066	0.5132	0.4275	0.4069	0.3848	0.3404
	FDN	**0.6302**	**0.6613**	**0.6777**	**0.6895**	**0.7166**	**0.7166**	**0.7032**	**0.6832**	**0.6671**	**0.6264**

Table 1: The performance comparison (SSIM) *w.r.t.* different numbers of inputs $|M|$ and different ϵ-augmentation.



(a) Examples *w.r.t.* $|M|$

(b) Examples *w.r.t.* ϵ

Figure 4: The comparison of restored images *w.r.t.* (a) different $|M|$ and (b) different ϵ. The first row shows results of CA; the second row shows results of CA-CN; and the third row shows results of FDN. See the arXiv version for more examples.

difference between the images restored by FDN and images restored by CA is in the color profile. FDN can precisely restore the color profile and salient features, while CA loses considerable details and generates color shift areas in the restored images (as discussed in the preliminary section), which is most obvious in CELEBA.

Different augmentation levels. We first fix $|M_p| = 50$ and generate different testing encryption datasets from different $\mathcal{X}_p$ with different ϵ_p, then attack these datasets via an FDN trained on encryptions generated from $\mathcal{X}_t =$ CIFAR10, $|M_t| = 30$ and $\epsilon_t = 0.1$. Note that we use the *filtering* phase to process each set of homogeneous encryptions before inputting them to FDN. Tab. 1 and Fig. 4b show the restoration performance and some examples. From Tab. 1, we observe that the performance of FDN degrades with the increasing of ϵ_p. The main reason is that the filtering step reduces the number of homogeneous encryptions input to FDN. Specifically, the general numbers of encryptions input to FDN after filtering are 25, 19 and 16 corresponding to $\epsilon_p = 0.3$, 0.4, 0.5, respectively. Less number of encryptions contain less information for the restoration of target images, causing the performance degradation of FDN. Nevertheless, Fig. 4b shows that FDN can restore far better colors and structures than CA. Note that CA is a pixel-wise optimization method which is developed to restore the private images without any transformations. When recovering images pre-processed by data augmentation, the corresponding pixels from different encryptions could be unaligned, and are thus treated as noise and factored out by CA, leading to considerable detail loss.

In addition, the results of attacking InstaHide Challenge, the classification utility tests of InstaHide with different ϵ, and the *ablation studies* are reported in the arXiv version.

Conclusion

In this paper, we design a fusion-denoising attack (FDN) on the real-world variations of InstaHide. Although image relaxing could cause some detail loss and reduce the sharpness of the restored images, the experiments demonstrate that FDN can precisely restore the color profiles and structures, which issues an alert to the ML applications that seek to use a revised version of InstaHide to protect the private images. Nevertheless, the motivation of InstaHide, i.e., corrupting the visual features of private images, is promising in future studies. We believe more secure methods that incorporate data encryption into machine learning will play an important role in both the security community and AI systems.

Acknowledgments

This research is supported by Singapore Ministry of Education Academic Research Fund Tier 3 under MOEs official grant number MOE2017-T3-1-007.

References

2020a. A Challenge for InstaHide. https://github.com/Hazelsuko07/InstaHide_Challenge. Online; accessed 1-February-2021.

2020b. InstaHide Training. https://github.com/Hazelsuko07/InstaHide. Online; accessed 15-May-2021.

Arora, S. 2020. How to allow deep learning on your data without revealing the data. http://www.offconvex.org/2020/11/11/instahide/. Online; accessed 15-January-2021.

Berthelot, D.; Carlini, N.; Goodfellow, I.; Papernot, N.; Oliver, A.; and Raffel, C. A. 2019. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 5049–5059.

Carlini, N.; Deng, S.; Garg, S.; Jha, S.; Mahloujifar, S.; Mahmoody, M.; Song, S.; Thakurta, A.; and Tramer, F. 2020. An Attack on InstaHide: Is Private Learning Possible with Instance Encoding? *arXiv preprint arXiv:2011.05315*.

Coates, A.; Ng, A.; and Lee, H. 2011. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 215–223.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Dosovitskiy, A.; and Brox, T. 2016. Inverting visual representations with convolutional networks. In *CVPR*, 4829–4837.

Fredrikson, M.; Jha, S.; and Ristenpart, T. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*, 1322–1333.

Fu, Y.; Wang, H.; Xu, K.; Mi, H.; and Wang, Y. 2019. Mixup Based Privacy Preserving Mixed Collaboration Learning. In *SOSE*, 275–2755. IEEE.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hitaj, B.; Ateniese, G.; and Perez-Cruz, F. 2017. Deep models under the GAN: information leakage from collaborative deep learning. In *CCS*, 603–618.

Huang, Y.; Song, Z.; Li, K.; and Arora, S. 2020. Instahide: Instance-hiding schemes for private distributed learning. In *ICML*, 4507–4518.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Li, H.; He, X.; Tao, D.; Tang, Y.; and Wang, R. 2018. Joint medical image fusion, denoising and enhancement via discriminative low-rank sparse dictionaries learning. *Pattern Recognition*, 79: 130–146.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3): 50–60.

Liu, L.; Xu, L.; and Fang, H. 2020. Infrared and visible image fusion and denoising via ℓ_2 - ℓ_p norm minimization. *Signal Processing*, 172: 107546.

Liu, Y.; Liu, S.; and Wang, Z. 2015. A general framework for image fusion based on multi-scale transform and sparse representation. *Information fusion*, 24: 147–164.

Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep Learning Face Attributes in the Wild. In *ICCV*, 3730–3738.

Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2): 91–110.

Luo, X.; Wu, Y.; Xiao, X.; and Ooi, B. C. 2021. Feature inference attack on model predictions in vertical federated learning. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 181–192. IEEE.

Ma, J.; Ma, Y.; and Li, C. 2019. Infrared and visible image fusion methods and applications: A survey. *Information Fusion*, 45: 153–178.

Mao, X.-J.; Shen, C.; and Yang, Y.-B. 2016. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. *arXiv preprint arXiv:1603.09056*.

Mei, J.-J.; Dong, Y.; and Huang, T.-Z. 2019. Simultaneous image fusion and denoising by using fractional-order gradient information. *Journal of Computational and Applied Mathematics*, 351: 212–227.

Ooi, B. C.; Tan, K.; Wang, S.; Wang, W.; Cai, Q.; Chen, G.; Gao, J.; Luo, Z.; Tung, A. K. H.; Wang, Y.; Xie, Z.; Zhang, M.; and Zheng, K. 2015. SINGA: A Distributed Deep Learning Platform. In *Proceedings of the ACM International Conference on Multimedia*, 685–688.

Raynal, M.; Achanta, R.; and Humbert, M. 2020. Image Obfuscation for Privacy-Preserving Machine Learning. *arXiv preprint arXiv:2010.10139*.

Rublee, E.; Rabaud, V.; Konolige, K.; and Bradski, G. 2011. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, 2564–2571.

Shorten, C.; and Khoshgoftaar, T. M. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1): 1–48.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612.

Wu, Y.; Cai, S.; Xiao, X.; Chen, G.; and Ooi, B. C. 2020. Privacy Preserving Vertical Federated Learning for Tree-based Models. *Proc. VLDB Endow.*, 13(11): 2090–2103.

Yang, Q.; Liu, Y.; Chen, T.; and Tong, Y. 2019. Federated machine learning: Concept and applications. *TIST*, 10(2): 1–19.

Zagoruyko, S.; and Komodakis, N. 2015. Learning to compare image patches via convolutional neural networks. In *CVPR*, 4353–4361.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhang, X.; and Luo, X. 2021. Exploiting Defenses against GAN-Based Feature Inference Attacks in Federated Learning. arXiv:2004.12571.

Zhang, Y.; Bai, X.; and Wang, T. 2017. Boundary finding based multi-focus image fusion through multi-scale morphological focus-measure. *Information fusion*, 35: 81–101.

Zhang, Y.; Li, K.; Li, K.; Zhong, B.; and Fu, Y. 2019. Residual non-local attention networks for image restoration. *arXiv preprint arXiv:1903.10082*.

Zhang, Y.; Liu, Y.; Sun, P.; Yan, H.; Zhao, X.; and Zhang, L. 2020a. IFCNN: A general image fusion framework based on convolutional neural network. *Information Fusion*, 54: 99–118.

Zhang, Y.; Tian, Y.; Kong, Y.; Zhong, B.; and Fu, Y. 2020b. Residual dense network for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhao, H.; Gallo, O.; Frosio, I.; and Kautz, J. 2016. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, 3(1): 47–57.

Zhu, L.; and Han, S. 2020. Deep leakage from gradients. In *Federated Learning*, 17–31. Springer.

Zitova, B.; and Flusser, J. 2003. Image registration methods: a survey. *Image and vision computing*, 21(11): 977–1000.