Image Disguising for Protecting Data and Model Confidentiality in Outsourced Deep Learning

Sagar Sharma
HP Inc.
Vancouver, WA
sagar.sharma@hp.com

A K M Mubashwir Alam and Keke Chen
Trustworthy and Intelligent Computing Lab
Marquette University, Milwaukee WI
{mubashwir.alam, keke.chen}@marquette.edu

2021 IEEE 14th International Conference on Cloud Computing (CLOUD) | 978-1-6654-0060-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/CLOUD53861.2021.00020

Abstract

Large training data and expensive model tweaking are common features of deep learning development for images. As a result, data owners often utilize cloud resources or machine learning service providers for developing large-scale complex models. This practice, however, raises serious privacy concerns. Existing solutions are either too expensive to be practical, or do not sufficiently protect the confidentiality of data and model. In this paper, we aim to achieve a better trade-off among the level of protection for outsourced DNN model training, the expenses, and the utility of data, using novel image disguising mechanisms. We design a suite of image disguising methods that are efficient to implement and then analyze them to understand multiple levels of trade-offs between data utility and protection of confidentiality. The experimental evaluation shows the surprising ability of DNN modeling methods in discovering patterns in disguised images and the flexibility of these image disguising mechanisms in achieving different levels of resilience to attacks.

I. Introduction

Deep Neural Networks (DNN) have shown impressive performance across diverse domains such as image classification, natural language processing, speech recognition, and recommendation systems. However, DNN training is resource-intensive and time-consuming, requiring large training data, careful model architecture selection, and exhaustive model parameter tweaking. As a result, data owners or model developers often utilize multiple cloud GPUs or online model training services to lower their costs.

Despite its popularity, outsourcing DNN learning to the cloud raises privacy and security concerns about the sensitive training data and trained models [18], [4]. On one hand, cloud users cannot verifiably prevent the cloud provider from getting access to their data. In practice, using public clouds often means you have to fully trust your cloud provider. On the other hand, public cloud providers are not immune to security attacks, which may lead to breaches of sensitive data through both the insider [3], [4] and external attacks [14], [21]. Additionally, membership inference attacks [20], model inversion attacks [7], and adversarial example exploration [2], [16] can be applied to models directly to explore the training examples in DNN learning.

One possible solution to data and model confidentiality is to partition the entire data and learning task into sensitive and non-sensitive partitions and use cloud-client federated learning [10]. The non-sensitive portion, assuming it is much larger than the sensitive one, is exported to and processed by the cloud. Correspondingly, the learning process is also partitioned and distributed in between the cloud and the client, ensuring that the intermediate information exchanged between the two parties does not breach privacy. The collaborative deep learning framework proposed by Shokri et al. [19] may be tweaked into such a partition-based scenario. However, Hitaj et al. [9] show that an adversarial collaborator (e.g., a compromised cloud in the cloud-client scenario) is able to generate images resembling the sensitive classes owned by the victim parties (the trusted data owner), with a Generative Adversarial Network (GAN)-based attack.

Another approach is to train encrypted DNN models over encrypted data. However, due to the large training data and expensive training process in deep learning, cryptographic model training approaches are too expensive to be practical. A recent study on training small scale neural networks [15] (e.g., just two layers with a maximum of 128 neurons per layer) has shown astonishingly high communication, computation, and storage costs. As a result, cryptographic approaches are often limited to training small models [15], [17], or securely applying trained DNNs for prediction [23].

Researchers have also explored the application of differential privacy (DP) [5] in distributed (federated) learning scenarios [19], or trusted central training server [1]. However, DP works for the setting of sharing data and models without breaching individual training example's privacy. It does not fit the setting where data and model confidentiality has to be protected.

In summary, there is no satisfactory solution for protecting both data and model confidentiality in the outsourced model training scenario while also achieving ideal trade-off amongst confidentiality, model utility, and expenses. More importantly, none of the solutions successfully break the tie between original training space and trained models that is essential to model-targeted attacks.

Scope and contributions. We propose an image disguising approach to addressing the data and model confidentiality issues in training models in the cloud. The proposed image disguising mechanism protect both the training data and the learned models by casting training data into a confidential transformed space where powerful DNN models can still learn features and patterns distinguishing image classes. The intuition is twofold: (1) applying appropriate transformations and data protection mechanisms so that the disguised images

2159-6190/21/$31.00 ©2021 IEEE
DOI 10.1109/CLOUD53861.2021.00020

cannot be effectively reconstruct and re-link to original images; (2) meanwhile, powerful deep learning techniques can still pick up the unique topological/geometric features preserved in the transformed space to distinguish the originally defined classes of images in the transformed space. By doing so, the tie between the original training data and the learned model in the transformed space is broken, which also disables any model-based exploration [2], [16], [20], [7]. In the end, we can approximately preserve the *image distinguishability* for the target classification task, while destroying the *recoverability* of individual images.

There are several unique contributions. (1) We have designed two image disguising mechanisms: AES-based and random-projection-based for image-based DNN learning to preserve training data and model confidentiality in outsourced training. The goal is to study and achieve a good balance between the utility of disguised images and the level of confidentiality protection. (2) We have carefully analyzed the potential attacks under the outsourced deep learning settings and the resilience of different disguising mechanisms to attacks on data and model confidentiality. (3) We have conducted extensive experimental evaluations on several public datasets to show the trade-offs of different disguising schemes and related parameter settings between data utility and their resilience to attacks.

II. Preliminary

Threat Modeling. We are concerned with the confidentiality of both the sensitive training image data and the DNN models in the *outsourced training phase*. Here, we make some relevant security assumptions for our disguising mechanisms. 1) We consider the cloud provider to be an honest-but-curious adversary, which implies that a curious provider will still honestly deliver desired results to the data owner. 2) The adversary can observe the training data, the training process, and the trained models, including the structure of the DNN architecture and parameter settings for training. Thus, they can probe the observed items with any methods, such as any image reconstruction, re-identification, and membership attacks. 3) The adversary may have different levels of prior knowledge. (a) They may know what the model is used for (e.g., for face recognition), but do not know the disguising parameters. (b) Additionally, they may know a few pairs of images and their disguised versions. (4) We do not address evasive and poisoning attacks [2], [16], where adversaries will tamper the training data, which can be guarded with training data integrity checking. (5) The client infrastructure and communication channels are secure.

Potential Attacks. Recent studies have shown that attackers can explore training/testing examples and models, for example, to find adversarial examples misleading the prediction of deep neural networks [2], [16]. Such attacks depend on adversaries' clear understanding of the original image data and the ability of extensively experimenting with the developed models. Outsourced learning without protection makes these attacks much easier. With a protection mechanism on data and models, we consider a fundamental attack: training image re-identification that aims at linking the protected images to identifiable original images. Since the image disguising mechanisms break the link between the original training data and the learned models (in the transformed space), without successfully breaking the disguising mechanism the existing model-oriented attacks do not work anymore.

A few methods have been proposed to protect image training data in the outsourcing context, such as noise addition [6] and image blurring [13], and morphing [12]. They are clearly very weak as the visual characteristic content of the images is still perceivable and understandable (Figure 1), i.e., it is easy to visually re-identify the transformed images. Therefore, we investigate the basic requirement for protecting the confidentiality of image data: adversaries cannot *effectively* link the protected images to their original counter-parts, or reconstruct the original counter-parts with some prior knowledge. No known mechanism works for our purpose that preserves both confidentiality and utility of training data and models for deep learning.

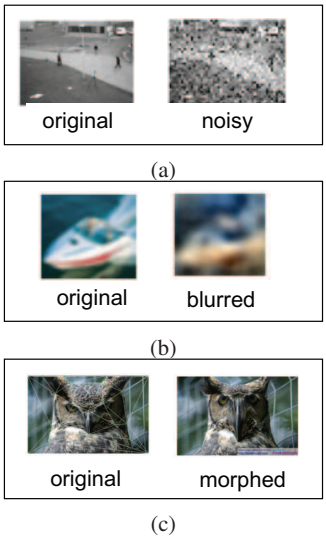

Fig. 1. (a) Differentially private noise addition to images [6]; (b) The reconstructed blurred images in PrivyNet [13]; (c) The morphed images [12]

III. DisguisedNets – a Novel Image Disguising Mechanism for Outsourced Deep Learning

In the following, we will introduce an image disguising framework that incorporates pixel-block partitioning, random block permutation, and block-wise transformations of images along with noise additions. The premise is that after the dramatic transformation, it is difficult to link the disguised images to the original images, while, unlike pure encryption schemes, it still preserves some essential patterns for distinguishing between classes of images that allow DNN learning methods to capture. This amalgam of multiple transformations provides a sufficiently large parameter space so that the attacks are computationally intractable (Section IV), which enables us to break the reconstruction based attacks and confidently hide the *visually identifiable* features of the images.

Figure 2 depicts the Disguised-Nets framework. A data owner disguises her private images before outsourcing them to the cloud for DNN learning. She can either fully outsource the entirety of the images and learning to the cloud or selectively retain sensitive images in the cloud-client partitioning setting.

She transforms all of her images using one secure transformation key secret to her. Note that this transformation should be at a reasonable cost, practical for a client's infrastructure to process.

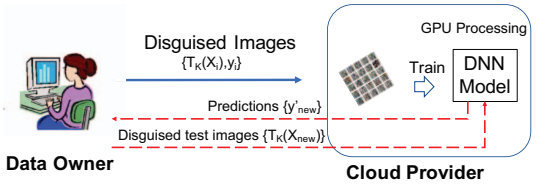

Fig. 2. Disguised-Nets: Image disguising framework for DNN learning.

Specifically, assume the data owner owns a set of images for training, notated as pairs $\{(X_i, y_i)\}$, where X_i is the image pixel matrix ($l \times m$ and $3 \times l \times m$ for grayscale and RGB images respectively) and y_i the corresponding label. We formally define the disguising process as follows. Let the disguising mechanism be a transformation T_K, where K is the secret key that depends on the selected perturbation techniques. By applying image disguising, the training data is transformed to $\{(T(X_i), c_i)\}$ with c_i mapped to $0, 1, \ldots$ randomly representing the classes y_i. The transformed images and classes are used to train a DNN, denoted as a function $\mathbf{D}_T$, that takes disguised images $T(X)$ and outputs a predicted label $\hat{c}$. The models trained on images transformed with the image disguising mechanisms only work on transformed images thus cannot be exploited with other image data as long as K is secure. For any new data X_{new}, the model application is defined as $\mathbf{D}_T(T(X_{new}))$, the new data transformed with the same key.

A. Pixel-Block Partitioning and Block-based Random Permutation

An image $X_{l \times m}$ is first partitioned into t blocks of uniform size $r \times s$. If we label the blocks sequentially as $v =< 1, 2, 3, 4, \ldots, t >$, a pseudorandom permutation of the image, $T_\pi(X)$, shuffles the blocks and reassemble the corresponding image accordingly. Block-based permutation preserves the in-block information and the relative positions of related blocks. Thus, we understand it preserves a great amount of information for effective modeling. However, while the permutation may break the global patterns of the images and achieve good visual privacy already, the between-block characteristics such as boundaries, color, content shape, and texture of the original neighboring blocks may provide clues for adversaries to recover the original image - imagine the jigsaw puzzle! For large t, such attacks can be time-consuming due to the vague similarity between block boundaries. However, with the prior knowledge: a pair of original image and its block-permuted image, it's not impossible. Thus, we use this as an auxiliary step in the disguising framework.

B. Pixel-Block Transformations

Next, we establish pixel-block-level protection mechanisms that aim to preserve the data utility for classification and further increase the resilience to attacks. We consider several candidate mechanisms, including encryption and random projection schemes, and discuss their characteristics. Note that the block-level parameter (e.g., the encryption key or transformation matrix) will be shared by different images for the same block position. Specifically, when an image is partitioned into t pixel blocks for random permutation, we get a list of t parameters $\{K_i, i = 1 \ldots t\}$, one for the pixel-block at the same position across the whole dataset. We name the specific position of the pixel block in the image *the pixel-block position*. The list of parameters acts as a secret key and will be shared, together with the permutation key, by each image in the dataset. The purpose of this setting is to maximize the preservation of distinguishable patterns between image classes - i.e., a pair of similar image patterns (blocks) can still be transformed to another similar pair after applying the disguising mechanism.

AES Encryption. The existing AES encryption schemes typically use 128-bit encryption keys, which encode every 16-byte data block sequentially. If we use AES for pixel-block encryption, assuming each pixel is stored in one byte, 16 original pixels are mapped to 16 encrypted bytes (pixels), and a whole pixel block is encoded to 16-byte units. Putting all encrypted pixel blocks together, we get a disguised image. For clear presentation, when we talk about AES encryption block, i.e., 16 bytes for a 128-bit encryption key, we use the 16-byte "encryption unit", which are different from "pixel blocks" we have been using previously in our image disguising framework. Figure 3 shows some example AES transformations on images.

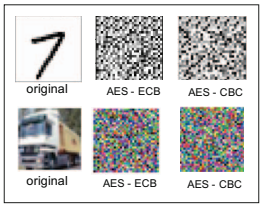

Fig. 3. Pixel-block based AES encryption of MNIST and CIFAR-10 images.

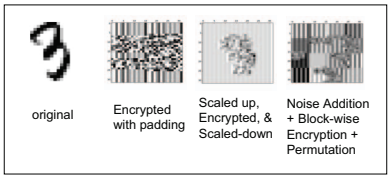

Fig. 4. AES-ECB encryption of MNIST image with different strategy.

We consider two AES modes in our design. First, we observe that with the AES Cipher Block Chaining (CBC) mode, any pixel-level change in the pixel block between two images will result in different encoding results for most 16-byte AES blocks in this pixel block position, making it not ideal for our purpose (we have observed accuracy lower than 10% across different datasets in our experiments).

Then, we turn to the AES Electronic Code Book (ECB) mode that can be considered as a fixed mapping function between 16-byte original data to 16-byte encrypted data. Different from CBC, the neighboring 16-byte blocks do not affect the encoding of the current block. This matches our requirement of data utility preservation, e.g., to preserve the block-level distinguishable patterns after the transformation. To preserve more information, based on the intuition of smaller blocks preserving more inter-pixel-block information, we can also use unit sizes smaller than the regular size (i.e., 16 pixels) for ECB. The method is to scale up the image first, e.g., from

32x32 to 256x256 (where each pixel is duplicated eight times), then encrypt it by 16-pixel units, and finally scale down to the size 32x32. Please refer to Figure 4 for the detailed example. It's equivalent to encrypt 2-pixel units in the original 32x32 images. We found by reducing the block size, the model quality can be significantly improved with the cost of less attack resilience.

The security guarantee of AES implies that it's computationally intractable to derive the encryption key. However, the ECB mode has a significant weakness due to the repetitive pixel-sequence patterns observed in different images, which can be used to design attacks without knowing the key. It's more significant for the images in the same class, where identical pixel-sequence patterns might be observed at the same pixel-block position. With the additional prior knowledge, e.g., knowing pairs of original images and encrypted images, the attacker can build a codebook for the mapping between the 16-byte original pixel to 16-byte encrypted pixel, which we will discuss more details in security analysis. To minimize the impact of the codebook attack, we introduce random noisy pixels, i.e., using the well-known "salt and pepper" noise addition method as a part of ECB-based method, to the original images before the block-level AES ECB encoding.

Multidimensional Projection. For an image represented as a pixel matrix X, a general linear transformation can be defined as $G(X) = RX$, where $R_{m\times m}$ is a random orthogonal matrix generated following the Haar distribution [8], or a random invertible matrix, e.g., a random projection matrix [22]. We call this method the randomized multidimensional transformation (RMT). When an image is partitioned into t blocks for random permutation, we prepare a list of random matrices $\{R_i, i = 1..t\}$, one for each image-block and share this list for each image. Figure 5 shows the effects of RMT on MNIST and CIFAR-10 datasets.

Such transformation is known to preserve (or approximately preserve by random projection) the Euclidean distance between columns of the matrix X. Without known pairs of original images and their transformed ones, it is not possible to recover the original data from the transformed data due to the large parameter space. However, with known pairs, they are vulnerable to regression attacks, which will be discussed in Section IV.

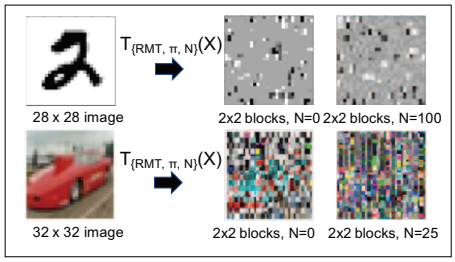

Fig. 5. Block-wise RMT+Noise on MNIST and CIFAR-10 images.

IV. Attack Analysis

The proposed disguising mechanisms can break the direct visual re-identifiability and successfully hide the visual attributes in the images recognizable to the human eyes, compared to other preliminary methods [13], [6]. Without additional prior knowledge, Disguised-Nets mechanisms provide strong confidentiality protection, as shown in the discussion of "brute-force attacks". More sophisticated attacks try to reconstruct the images with prior knowledge. Finally, the proposed methods will make the trained models in the transformed spaces immune to existing model-oriented attacks as the attackers have to break the transformation first in order to launch these attacks.

A. Brute-force Attacks

Without any prior knowledge, the brute-force attack on AES schemes is not possible and thus we focus on the schemes using multidimensional projection. Let us start with the block-level transformation for any block i with RMT. With $X'_i = X_i R_i$, the adversary knows only X'_i. To explore possible X_i in a brute-force manner as no other information is known, the overall attacking complexity is determined by the number of possible R_i matrices. We show that the number of possible R_i (even limited to orthogonal ones) can be exponentially large for given parameters.

Proposition 1: For values encoded in h-bit finite field, there are $O(2^{hm})$ candidate orthogonal matrices $R_{m\times m}$.

The proof is based on the theory of orthogonal matrix group, the detail of which is skipped here. With a typical setting in our experiments, e.g., $h = 32$ and $m = 7$ for the MNIST dataset, the overall complexity is $O(2^{224})$, which is more than sufficient to protect from any computationally-bounded attackers. Combined with the random permutation of blocks, the attack complexity is even higher.

B. Knowledge-Enhanced Attacks

A powerful adversary may manage to gain additional knowledge about the disguising mechanism: at the minimum level, the adversary might be able to get pairs of original image and the transformed one. Below, we focus on the *codebook attack* on the AES-ECB based disguising mechanism and the regression-based attack on the RMT mechanism.

Codebook Attack. The assumption is that the adversary is knowledgeable of the encryption procedure described above but not the encryption/decryption key. Due to the fact that we are not using fully randomized method to preserve data utility, the basic idea is to build a mapping (i.e., the codebook) between the plaintext unit (e.g., the 16-byte AES block) and its encrypted counterpart (e.g., the 16-byte AES cipher block). By processing the known image pairs, the adversary constructs a codebook as a dictionary of 16-byte pixel block mapped to encrypted 16-byte pixel block. Since different images, especially, those that are in the same class, might share some 16-byte pixel blocks, it's possible that some 16-byte encrypted blocks in the attacked images will show up in the codebook. For encrypted pixel pairs that are not present in the codebook, the adversary may map to a fixed pattern, e.g., all zero values, or set to most likely values. By repeating this procedure for each 16-byte block (we call it encryption unit later), the adversary can reconstruct a plausible image. Let the *hit rate* defined as the probability that an encrypted pixel block can find a match in the codebook.

This attack can become less effective if we add salt-and-pepper noises to the original images before encoding,

which reduce the hit rate significantly and make the mapping non-unique: the same 16-byte pixel block can be mapped to different ciphertexts. We evaluate the *success rate* of this attack using the accuracy that the DNN examiner trained with the original image space correctly classifies the re-constructed images.

Projection Matrix Estimation Attack. Note that the codebook attack does not work for the multidimensional projection method with noise addition, as the noise added pixel blocks have extremely low probability to be identical. However, there is a possibility that the transformation matrix might be estimated with linear regression, if the adversary has obtained an enough number of original and transformed image pairs. Specifically, a noise-added block-wise transformation is $Y_i = (X_i + \Delta_i)R_i$, where Δ_i is a random noise matrix, re-generated for each image block X_i, and drawn uniformly at random from $[0, N]$ where N is the tunable noise level. With known pairs of (X_i, Y_i), regression can be applied to estimate R_i. In general, the more known pairs, the more precise the estimation can be. We will examine the effectiveness of the regression-based attacks.

V. Experiments

The proposed image disguising mechanisms for outsourced deep learning involve intrinsic trade-offs between data utility and the resilience to attacks. In experiments, we show how different mechanisms and parameter setting will affect the model quality (data utility), then, evaluate resilience to attacks to find the best setting that maximize both data utility and attack resilience in Section IV.

Datasets. We use two prevalent DNN benchmarking datasets: MNIST and FASHION for experiments. MNIST (handwritten digits) and FASHION (fashion items) image-sets both consists of 60,000 training and 10,000 testing gray-scale 28×28 pixel-images with 10 classes. We used five folds of random sampling to estimate the standard deviation of modeling results, which are also used for later experiments. Table I summarizes the datasets, the techniques used to train the models, and their baseline model accuracy on the original image data when trained with AlexNet implemented in TensorFlow.

TABLE I. Datasets and baseline accuracy. Tr: Training, Te: Testing

Datasets	$Numberofimages$	Image size	Architecture	Baseline Accuracy
MNIST	(60000 Tr,10000 Te)	$\{28 \times 28\}$	AlexNet	$96.7 +/- 0.2\%$
FASHION	(60000 Tr,10000 Te),	$\{28 \times 28\}$	AlexNet	$88.7 +/- 0.3\%$

A. Evaluation on Utility Preservation

In this section, we focus on the utility preservation aspect of the disguising mechanisms and show how different parameter settings will affect data utility. Note that the cost of the proposed image disguising mechanisms are quite low, e.g., per image cost is less than 10 ms, and can be comfortably done by any PC or even mobile phones. The disguised images are used directly with the existing DNN training algorithms without any modification to the algorithm or data. Thus, we ignore the details of cost evaluation.

The first set of experiments is to understand the effect of block-size setting for the AES ECB based block protection. We use "pixel blocks" for partitioning and permutation, and "units" for AES encryption units. A pixel block typically contains more than one unit. A smaller block size results in a larger number of blocks after partitioning, as the image size is fixed. For clear presentation, we use the number of blocks as the x-axis of the following figures. If more than one block is generated in partitioning, we also apply a secret block-wise permutation to enhance the confidentiality. Recall that AES uses 16 bytes as the encryption unit if 128-bit encryption is used. Figure 6 shows different block size settings from 1 block (32x32 per block) to 64 blocks (4x4 pixels per block). Smaller block size seems to preserve more information. We also tested the effect of smaller unit sizes on the model quality. Figure 6 (right) shows that the model quality is further improved with only 2-3% worse than the baselines. However, this method is also relatively vulnerable to attacks, which will be discussed with the noise addition method in the next section.

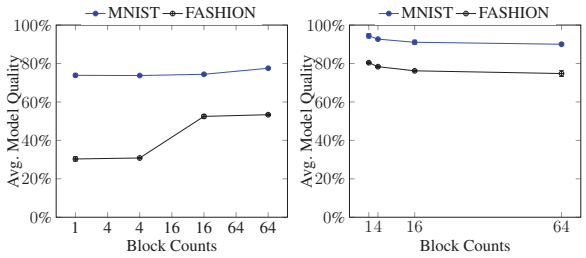

Fig. 6. Effect on Model Quality on AES-ECB-disguised Images: Varying block counts. Regular (left). Scale-up Scale-down (right)

Next we look at the effects of block size and different projection methods on models trained on images transformed with RMT methods. Figure 7 (right) shows that the model quality is slightly affected with smaller block sizes (more blocks per image). Overall, the model quality is well preserved, only 2-3% worse than the baseline. However, Figure 7 (left) shows an interesting result that orthogonal matrices preserve more data utility than random projection matrices. Surprisingly, adding noise to the images before transformation does not change the model quality much.

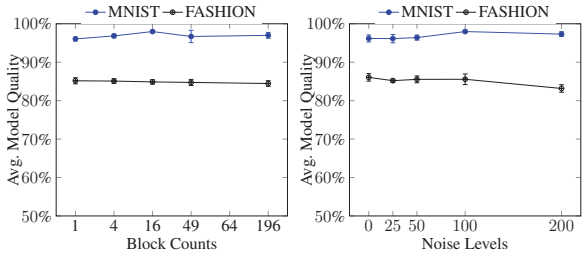

Fig. 7. Effects of block size and varying noise levels on model quality for RMT-disguised images: orthogonal vs. random invertible projection (left). Block size (right).

B. Evaluation on Attack Resilience

With the known additional knowledge, i.e., pairs of original and disguised images, the disguising mechanisms might be under the reconstruction attack, where attackers will visually check the reconstructed images. Some existing disguising

methods [12], [6], [13] are so weak that the disguised images can be identified without additional knowledge and the reconstruction step.

To effectively evaluate the re-identification step, we use a DNN trained on the original image data to simulate the attacker in the visual re-identification process. The intuition is that if any features in the disguised (or reconstructed) images can be detected visually by the adversary, it can be used to link the disguised (or reconstructed) images to the original images. Such linking is often probabilistic, and we can use the linking success rate (i.e., the accuracy of prediction) to gauge the threat level of attacks. As DNNs perform comparably well as human experts do in the image-based classification tasks [11], we believe such "DNN examiners" can satisfactorily simulate the attacker.

We train DNN examiners with the original training data using the same DNN architectures detailed in Table I. We then apply the DNN examiners to see whether the reconstructed images can be correctly classified to their original labels. To minimize the impact of DNN architecture and different baseline accuracy, we define the *attack success rate* as

$$\frac{\text{accuracy of DNN examiner on disguised/reconstructed images}}{\text{accuracy of DNN examiner on original images}} \times 100.$$

1) Resilience to Codebook Attack for the AES ECB method: Assume the attacker knows m pairs of original images and their ECB encrypted ones, and also other information such as their pixel-block sizes. The codebook attack uses the known pairs to construct a mapping between the known plaintext 16-byte pixels (or a reduced number of pixels if the scaling up/down method is used to preserve more utility) and the corresponding encrypted 16-byte pixels. The attacker might be able to use the codebook to partially recover the original pixel blocks of a disguised image (with random pixel patches for unrecognized blocks). We use the DNN examiner to examine the quality of reconstructed images.

Due to the limited space and clear presentation, we show only the evaluation on the MNIST data as the Fashion data has a similar pattern. Figure 8 compares the attack results on 16-pixel encryption units (left) and 2-pixel encryption units (by applying the scale-up/down method) (right). The attacker's known pairs are selected from the training data, while the attacked images are selected from the testing data. 16-pixel encryption unit gives a one-to-one mapping between the original pixel units and the encrypted ones. We observed hit rates are quite low (lower than 10%), but success rates are increasing steadily due to the increased codebook size. However, attackers will need a large number of pairs to achieve a good success rate. 2-pixel encryption unit actually creates a one-to-many mapping between original pixel units and the encrypted ones, because the scaling-down stage, for which we used the Python library function for image scaling, merges the nearby pixels. If we consider a hit is finding a one-one match in the codebook, we observed that hit rates initially increase to around 10% and then drop to 2-3%. The success rate reaches the plateau – around 50% with only 20 image pairs. Overall, the 16-pixel unit method is more resilient to attacks – both the hit rates and success rates grow slowly and knowing the whole training data does not help improve the success rates much. Therefore, we have seen the first significant trade-off: the 2-pixel encryption unit method preserves more utility, but it's also more vulnerable to known-pairs attacks.

Hoping to achieve both good utility and attack resilience for the setting of the 2-pixel encryption unit, we found that it's possible to defend from the codebook attack by adding "salt-and-pepper" noises to the original images. The AES encrypted pixel block changes dramatically when any of the original pixel changes, which helps reduce the attack success rate. Figure 9 shows by adding a small amount of noise, e.g., 2-3%, the attack success rate drops by 10%, while the model quality is not significant damaged. However, adding noise should be carefully done to avoid destroying the data utility – if the noise level is too high, the model quality will drop significantly, too.

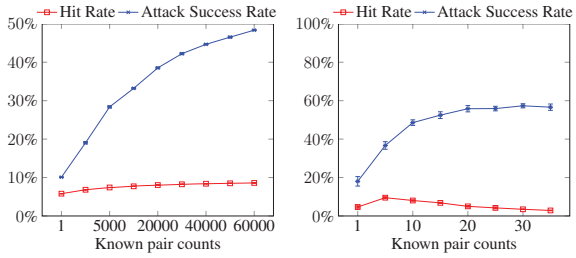

Fig. 8. Codebook attack on MNIST dataset with varying number of known pairs. Using 16-pixel encryption unit (left). Using 2-pixel encryption unit (right)

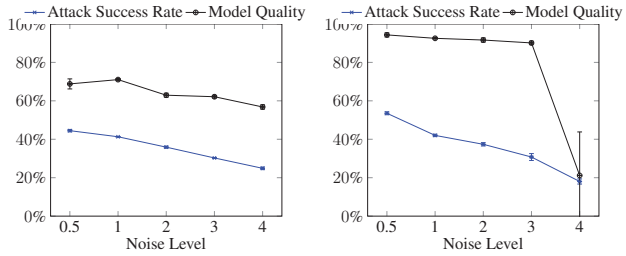

Fig. 9. Protecting AES-based disguising with noise addition (MNIST data). 16-pixel encryption units (left). 2-pixel encryption units (right)

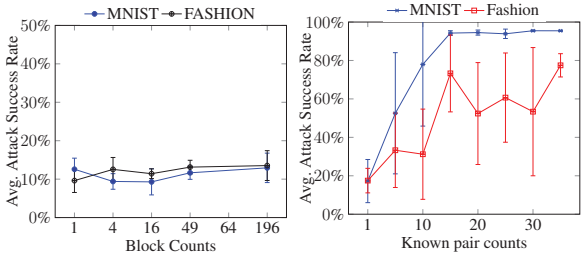

Fig. 10. Direct re-identification attack on RMT disguised images for different block size (left). Regression attack on the noise-added RMT disguising method: images with block-size 7×7 and noise level $u = 100$ (right).

2) Resilience to Attacks for the RMT scheme: We study how known pairs can be effectively used to attack the RMT method. Again, we assume the strong attack case: the attacker also knows the pixel-block size and the permutation pattern. By known only one pair of images, RMT without noise addition can be easily broken – the block-wise transformation parameters $\{R_i, i = 1..m\}$ can be straightforwardly recovered.

TABLE II. RESULTS OF APPLYING DIFFERENT IMAGE DISGUISING MECHANISMS.

	Model Accuracy (%)			Attack Success Rates (%)		
Datasets	RMT Disguise	AES ECB Disguise	Baseline (no disguise)	RMT Disguise	AES ECB Disguise	Baseline (no disguise)
MNIST	96.6 ± 0.4 %	91.6 ± 1.1%	96.7 ±0.2%	9.29 ± 3.3 %	37.3 ± 0.9 %	96.7 ±0.2%
FASHION	85.1 ± 0.6 %	68.9±1.4%	88.7 ±0.3%	11.42 ± 1.3 %	19.3 ± 0.8 %	88.7 ±0.3%

With noise addition, the only known attack method is to use linear regression to estimate the parameters $\{R_i\}$, the accuracy of which is affected by the noise intensity (i.e., the variance of noise). Different from the "salt-and-pepper" noise, we generate a noise value for each pixel and add it to the original pixel value before applying the projection, i.e., $Y_i = (X_i + \Delta_i)R_i$, where the noise Δ_i is drawn from the uniform distribution $U(0, u)$. Figure 10 shows that the regression attack is surprisingly effective on both the MNIST data and the Fashion data. With a small number of known image pairs, the attack can achieve surprisingly high success rates.

C. Discussion

By summarizing the effect of parameter settings and the mechanisms improving attack resilience, we have the best results for each disguising mechanism. Without prior adversarial knowledge, the RMT mechanism preserves much better utility. With prior adversarial knowledge, AES ECB with small encryption unit and 2% noise addition satisfactorily protect from the codebook attack and also the best model quality. Table II summarizes the best results we have so far.

Currently, we only consider learning DNN models. It will be important to assess if other learning methods also share similar results. Furthermore, transfer learning is extremely useful for building powerful models with a small number of labeled data. It's interesting to investigate whether and how a disguised model can be effectively transferred to a related domain without revealing the disguising parameters.

VI. CONCLUSION

Outsourcing large image datasets to the cloud for deep learning has been an economical and popular option, while it also raises concerns on data and model confidentiality. The existing solutions are either too expensive to be practical, vulnerable to different model-based adversarial attacks, or simply ineffective in protecting the image content. By focusing on the training image reconstruction attacks, we propose several image disguising mechanisms that thwart the attacks efficiently while preserving satisfactory model quality. Our use of block-wise random permutation, block-wise AES encryption or multidimensional transformation does not require any changes to the existing DNN modeling architectures. Experimental results show that some settings of the proposed image disguising mechanisms can preserve the model quality and provide attack resilience surprisingly well. We also provide a comprehensive guideline for using different mechanisms to achieve variant trade-offs between data utility and data and model confidentiality.

REFERENCES

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. 2016.

[2] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay. Adversarial attacks and defences: A survey. *CoRR*, abs/1810.00069, 2018.

[3] A. Chen. Gcreep: Google engineer stalked teens, spied on chats. *Gawker, http://gawker.com/5637234/*, 2010.

[4] A. J. Duncan, S. Creese, and M. Goldsmith. Insider attacks in cloud computing. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, 2012.

[5] C. Dwork. Differential privacy. In *International Colloquium on Automata, Languages andProgramming*. Springer, 2006.

[6] L. Fan. Image pixelization with differential privacy. In *Data and Applications Security and Privacy XXXII - 32nd Annual IFIP WG 11.3 Conference, DBSec 2018, Bergamo, Italy, July 16-18, 2018, Proceedings*, pages 148–162, 2018.

[7] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd USENIX Security Symposium USENIX Security 14*, pages 17–32, San Diego, CA, 2014. USENIX Association.

[8] J. Gallier. *Geometric Methods and Applications for Computer Science and Engineering*. Springer-Verlag, New York, 2000.

[9] B. Hitaj, G. Ateniese, and F. Pérez-Cruz. Deep models under the GAN: information leakage from collaborative deep learning. *CoRR*, abs/1702.07464, 2017.

[10] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, and et al. Advances and open problems in federated learning. *CoRR*, abs/1912.04977, 2019.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[12] S. Lee, K. Chwa, J. Hahn, and S. Shin. Image morphing using deformation techniques. *JOURNAL OF VISUALIZATION AND COMPUTER ANIMATION*, 7(1):3 – 23, n.d.

[13] M. Li, L. Lai, N. Suda, V. Chandra, and D. Z. Pan. Privynet: A flexible framework for privacy-preserving deep neural network training with A fine-grained privacy control. *CoRR*, abs/1709.06161, 2017.

[14] S. Mansfield-Devine. The Ashley Madison affair. *Network Security*, 2015(9):8 – 16, 2015.

[15] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[16] E. Raff, J. Sylvester, S. Forsyth, and M. McLean. Barrage of random transforms for adversarially robust defense. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6521–6530, 2019.

[17] S. Sharma and K. Chen. Confidential boosting with random linear classifiers for outsourced user-generated data. In *Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part I*, pages 41–65, 2019.

[18] S. Sharma, K. Chen, and A. Sheth. Toward practical privacy-preserving analytics for iot and cloud-based healthcare systems. *IEEE Internet Computing*, 22(2):42–51, Mar./Apr. 2018.

[19] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[20] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, May 2017.

[21] L. Unger. Breaches to customer account data. *Computer and Internet Lawyer*, 32(2):14 – 20, 2015.

[22] S. S. Vempala. *The Random Projection Method*. American Mathematical Society, 2005.

[23] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. E. Lauter, and M. Naehrig. Crypto-nets: Neural networks over encrypted data. *CoRR*, abs/1412.6181, 2014.